i.MX RT1060 Processor Reference Manual

With annotations for Teensy

Updated 21-Jul-2025

Document Number: IMXRT1060RM
Rev. 3, 07/2021

Contents

Section number **Title** **Page**

Chapter 1
About this Document

1.1 Audience 21

1.2 Organization 21

1.3 Suggested Reading 21

1.4 Conventions 22

1.5 Register Access 24

1.6 Acronyms and Abbreviations 26

Chapter 2
Introduction

2.1 Introduction 29

2.2 Features 31

2.3 Target Applications 33

2.4 Endianness Support 33

Chapter 3
Memory Maps

3.1 Memory system overview 35

3.2 Arm Platform Memory Map 35

Chapter 4
Interrupts, DMA Events, and XBAR Assignments

4.1 Chip-specific Interrupt information 43

4.2 Overview 43

4.3 CM7 interrupts 43

4.4 DMA Mux 52

4.5 PIT Channel Assignments For Periodic DMA Triggering 60

4.6 XBAR Resource Assignments 61

Chapter 5
Direct Memory Access Multiplexer (DMAMUX)

5.1 Chip-specific DMAMUX information..... 75
5.2 Overview..... 75
5.3 Functional description..... 77
5.4 External signals..... 82
5.5 Initialization/application information..... 82
5.6 Memory map/register definition..... 85

Chapter 6
Enhanced Direct Memory Access (eDMA)

6.1 Chip-specific eDMA information..... 89
6.2 Overview..... 89
6.3 Functional description..... 93
6.4 Initialization/application information..... 99
6.5 Memory map/register definition..... 114

Chapter 7
System Security

7.1 Chapter overview..... 173
7.2 Feature summary..... 173
7.3 High-Assurance Boot (HAB)..... 175
7.4 Secure Non-Volatile Storage (SNVS) module..... 177
7.5 Data Co-Processor (DCP)..... 179
7.6 Standalone True Random Number Generator (TRNG)..... 179
7.7 On-Chip OTP Controller (OCOTP_CTRL)..... 179
7.8 Central Security Unit (CSU)..... 180
7.9 System JTAG Controller (SJC)..... 180
7.10 Bus Encryption Engine (BEE)..... 181

Chapter 8
System Debug

8.1 Overview..... 183
8.2 Chip and Arm Platform Debug Architecture..... 183
8.3 Miscellaneous..... 190

Section number Title Page

Chapter 9
System Boot

9.1 Chip-specific Boot Information.... 191
9.2 Overview.... 195
9.3 Boot modes.... 197
9.4 Device configuration.... 201
9.5 Device initialization.... 203
9.6 Boot devices.... 209
9.7 Program image.... 256
9.8 Plugin image.... 263
9.9 Serial Downloader.... 263
9.10 Recovery devices.... 274
9.11 SD/MMC manufacture mode.... 274
9.12 High-Assurance Boot (HAB).... 275
9.13 ROM APIs.... 276

Chapter 10
External Signals and Pin Multiplexing

10.1 Overview.... 289

Chapter 11
IOMUX Controller (IOMUXC)

11.1 Overview.... 317
11.2 Functional description.... 319
11.3 IOMUXC GPR Memory Map/Register Definition.... 323
11.4 IOMUXC SNVS Memory Map/Register Definition.... 376
11.5 IOMUXC SNVS GPR Memory Map/Register Definition.... 395
11.6 IOMUXC Memory Map/Register Definition.... 399

Chapter 12
General Purpose Input/Output (GPIO)

12.1 Chip-specific GPIO information.... 945
12.2 Overview.... 945

12.3 Functional Description....947
12.4 External Signals....956
12.5 Application Information....957
12.6 Memory Map/Register Definition....958

Chapter 13
Clock and Power Management

13.1 Introduction....975
13.2 Device Power Management Architecture Components....975
13.3 Clock Management....978
13.4 Power management....987
13.5 ONOFF (Button)....1001
13.6 WAKEUP Pin....1002

Chapter 14
Clock Controller Module (CCM)

14.1 Chip-specific CCM information....1005
14.2 Overview....1005
14.3 External Signals....1008
14.4 CCM Clock Tree....1008
14.5 System Clocks....1011
14.6 Functional Description....1025
14.7 CCM Memory Map/Register Definition....1038
14.8 CCM Analog Memory Map/Register Definition....1089

Chapter 15
Crystal Oscillator (XTALOSC)

15.1 Chip-specific XTALOSC information....1131
15.2 Overview....1131
15.3 External Signals....1132
15.4 Crystal Oscillator 24 MHz....1132
15.5 Crystal Oscillator 32 kHz....1134
15.6 XTALOSC 24MHz Memory Map/Register Definition....1136

Chapter 16
Power Management Unit (PMU)

16.1 Chip-specific PMU information..... 1149
16.2 Overview..... 1150
16.3 Analog LDO Regulators..... 1151
16.4 USB LDO Regulator..... 1153
16.5 SNVS Regulator..... 1153
16.6 PMU Memory Map/Register Definition..... 1154

Chapter 17
General Power Controller (GPC)

17.1 Chip-specific GPC information..... 1179
17.2 Overview..... 1179
17.3 Clocks..... 1180
17.4 Power Gating Control (PGC)..... 1180
17.5 GPC Interrupt Controller (INTC)..... 1183
17.6 GPC Memory Map/Register Definition..... 1184
17.7 PGC Memory Map/Register Definition..... 1190

Chapter 18
DCDC Converter (DCDC)

18.1 Chip-specific DCDC information..... 1199
18.2 Overview..... 1200
18.3 Functional description..... 1201
18.4 Application information..... 1202
18.5 Memory Map and register definition..... 1204

Chapter 19
Temperature Monitor (TEMPMON)

19.1 Chip-specific TEMPMON information..... 1215
19.2 Overview..... 1215
19.3 Software Usage Guidelines..... 1217
19.4 TEMPMON Memory Map/Register Definition..... 1218

Chapter 20
Secure Non-Volatile Storage (SNVS)

20.1 Chip-specific SNVS information....1225
20.2 Overview....1225
20.3 SNVS functional description....1227
20.4 External Signals....1232
20.5 Initialization of SNVS....1233
20.6 Memory Map and register definition....1234

Chapter 21
System Reset Controller (SRC)

21.1 Chip-specific SRC information....1261
21.2 Overview....1261
21.3 Functional Description....1262
21.4 External Signals....1271
21.5 Initialization....1272
21.6 SRC Memory Map/Register Definition....1273

Chapter 22
Fusemap

22.1 Boot Fusemap....1287
22.2 Lock Fusemap....1298
22.3 Fusemap Descriptions Table....1299

Chapter 23
On-Chip OTP Controller (OCOTP_CTRL)

23.1 Chip-specific OCOTP_CTRL information....1311
23.2 Overview....1311
23.3 Functional Description....1312
23.4 External Signals....1319
23.5 Fuse Map....1319
23.6 Memory Map/Register Definition....1319

Chapter 24

External Memory Controllers

24.1 Overview....1355
24.2 Smart External Memory Controller (SEMC) Overview....1355
24.3 eMMC/eSD/SDIO....1356
24.4 Quad Serial Peripheral Interface....1357

Chapter 25
Smart External Memory Controller (SEMC)

25.1 Chip-specific SEMC information....1359
25.2 Overview....1359
25.3 Functional Description....1362
25.4 External Signals....1402
25.5 Initialization....1405
25.6 Application Information....1405
25.7 Memory Map and Register Definition....1408

Chapter 26
Ultra Secured Digital Host Controller (uSDHC)

26.1 Chip-specific uSDHC information....1465
26.2 Overview....1465
26.3 Functional description....1468
26.4 External signals....1501
26.5 Application of uSDHC....1502
26.6 Software restrictions....1528
26.7 uSDHC memory map/register definition....1530

Chapter 27
FlexSPI Controller

27.1 Chip-specific FlexSPI information....1601
27.2 Overview....1602
27.3 Functional description....1603
27.4 External Signals....1653
27.5 Initialization....1655

27.6 Application information....1655
27.7 Memory Map and register definition....1666
27.8 AHB Memory Map definition....1753

Chapter 28
ARM Cortex M7 Platform

28.1 Chip-specific Arm Cortex M7 information....1755
28.2 Overview....1756
28.3 Functional Description....1757
28.4 External Signals....1758
28.5 Memory Map and register definition....1758

Chapter 29
Network Interconnect Bus System (NIC-301)

29.1 Chip-specific NIC-301 information....1763
29.2 Overview....1763
29.3 External Signals....1764
29.4 Memory Map and Register Definition....1764

Chapter 30
On-Chip RAM Memory Controller (OCRAM)

30.1 Chip-specific OCRAM information....1775
30.2 Overview....1775
30.3 Functional Description....1777
30.4 Programmable Registers....1779

Chapter 31
FlexRAM

31.1 Chip-specific FlexRAM information....1781
31.2 Overview....1782
31.3 Functional description....1783
31.4 Memory Map and register definition....1786

Chapter 32
AHB to IP Bridge (AIPSTZ)

32.1 Chip-specific AIPSTZ information....1793
32.2 Overview....1793
32.3 Clocks....1794
32.4 Functional Description....1794
32.5 Access Protections....1795
32.6 Access Support....1795
32.7 Initialization Information....1796
32.8 AIPSTZ Memory Map/Register Definition....1798

Chapter 33
Display and Camera Overview

33.1 Overview....1815

Chapter 34
CMOS Sensor Interface (CSI)

34.1 Chip-specific CSI information....1817
34.2 Overview....1817
34.3 Functional Description....1819
34.4 External Signals....1833
34.5 Initialization....1835
34.6 CSI Memory Map/Register Definition....1835

Chapter 35
Enhanced LCD Interface (eLCDIF)

35.1 Chip-specific eLCDIF information....1857
35.2 Overview....1857
35.3 Functional Description....1859
35.4 External Signals....1876
35.5 Initialization....1877
35.6 eLCDIF Memory Map/Register Definition....1877

Chapter 36
Pixel Pipeline (PXP)

36.1 Chip-specific PXP information....1905

36.2 Overview....1905
36.3 Functional Description....1906
36.4 External Signals....1941
36.5 Initialization....1941
36.6 PXP Memory Map/Register Definition....1941

Chapter 37
Audio Overview

37.1 Audio Overview....1979

Chapter 38
Synchronous Audio Interface (SAI)

38.1 Chip-specific SAI information....1985
38.2 Overview....1986
38.3 Functional description....1988
38.4 External signals....1999
38.5 Memory map and register definition....1999

Chapter 39
Medium Quality Sound (MQS)

39.1 Chip-specific MQS information....2035
39.2 Overview....2035
39.3 Functional Description....2036
39.4 External Signals....2037
39.5 Application Information....2038

Chapter 40
Sony/Philips Digital Interface (SPDIF)

40.1 Chip-specific SPDIF information....2039
40.2 Overview....2039
40.3 Functional Description....2041
40.4 External Signals....2051
40.5 Application Information....2052
40.6 SPDIF Memory Map/Register Definition....2053

Chapter 41
10/100-Mbps Ethernet MAC (ENET)

41.1 Chip-specific ENET information....2079
41.2 Overview....2079
41.3 Functional description....2083
41.4 External Signals....2144
41.5 Memory map/register definition....2146

Chapter 42
Universal Serial Bus Controller (USB)

42.1 Chip-specific USB information....2251
42.2 Overview....2251
42.3 External Signals....2254
42.4 Functional Description....2255
42.5 USB Operation Model....2259
42.6 USB Non-Core Memory Map/Register Definition....2424
42.7 USB Core Memory Map/Register Definition....2433

Chapter 43
Universal Serial Bus 2.0 Integrated PHY (USB-PHY)

43.1 Chip-specific USB-PHY information....2511
43.2 USB PHY Overview....2511
43.3 Operation....2511
43.4 USB PHY Memory Map/Register Definition....2522
43.5 USB Analog Memory Map/Register Definition....2539

Chapter 44
Flexible Controller Area Network (FLEXCAN)

44.1 Chip-specific FLEXCAN information....2561
44.2 Overview....2561
44.3 External Signals....2566
44.4 Clocks....2567
44.5 Message Buffer Structure....2568

44.6 Rx FIFO Structure....2572
44.7 Functional Description....2575
44.8 Initialization/Application Information....2602
44.9 FLEXCAN Memory Map/Register Definition....2603

Chapter 45
Flexible Data-rate Controller Area Network (CANFD/FlexCAN3)

45.1 Chip-specific FLEXCAN information....2647
45.2 Overview....2647
45.3 Functional description....2650
45.4 FlexCAN signal descriptions....2706
45.5 Initialization/application information....2707
45.6 Memory map/register definition....2708

Chapter 46
Keypad Port (KPP)

46.1 Chip-specific KPP information....2773
46.2 Overview....2773
46.3 Functional Description....2775
46.4 External Signals....2783
46.5 Initialization/Application Information....2785
46.6 Memory Map and Register Definition....2786

Chapter 47
Low Power Inter-Integrated Circuit (LPI2C)

47.1 Chip-specific LPI2C information....2793
47.2 Overview....2793
47.3 Functional description....2796
47.4 Signal descriptions....2808
47.5 Memory Map and Registers....2809

Chapter 48
Low Power Serial Peripheral Interface (LPSPI)

48.1 Chip-specific LPSPI information....2853

48.2 Overview....2853
48.3 Functional description....2855
48.4 Signal descriptions....2866
48.5 Memory map and registers....2867

Chapter 49
Low Power Universal Asynchronous Receiver/Transmitter (LPUART)

49.1 Chip-specific LPUART information....2893
49.2 Overview....2893
49.3 Functional description....2896
49.4 Clocking and Resets....2914
49.5 External signals....2914
49.6 Register definition....2915

Chapter 50
Flexible I/O (FlexIO)

50.1 Chip-specific FlexIO information....2943
50.2 Overview....2944
50.3 Functional description....2946
50.4 Application Information....2959
50.5 FlexIO Signal Descriptions....2978
50.6 Memory Map and Registers....2978

Chapter 51
Timers Overview

51.1 Overview....3009

Chapter 52
General Purpose Timer (GPT)

52.1 Chip-specific GPT information....3013
52.2 Overview....3013
52.3 External Signals....3015
52.4 Clocks....3016
52.5 Functional Description....3018

52.6 Initialization/ Application Information 3022
52.7 Memory map and register definitions..... 3023

Chapter 53
Periodic Interrupt Timer (PIT)

53.1 Chip-specific PIT information..... 3035
53.2 Introduction..... 3035
53.3 Modes of operation..... 3037
53.4 Functional description..... 3037
53.5 Initialization and application information..... 3041
53.6 PIT register descriptions..... 3042

Chapter 54
Quad Timer (TMR)

54.1 Chip-specific TMR information..... 3051
54.2 Overview..... 3051
54.3 Modes of Operation..... 3053
54.4 Functional Description..... 3053
54.5 Resets..... 3070
54.6 Clocks..... 3070
54.7 Interrupts..... 3070
54.8 DMA..... 3072
54.9 Memory map/register definition..... 3073

Chapter 55
Enhanced Flex Pulse Width Modulator (eFlexPWM)

55.1 Chip-specific FlexPWM information..... 3091
55.2 Overview..... 3091
55.3 Functional Description..... 3094
55.4 External Signals 3130
55.5 Resets..... 3132
55.6 Interrupts..... 3132
55.7 DMA..... 3134

55.8 PWM register descriptions....3135

Chapter 56
Quadrature Decoder (QDC)

56.1 Chip-specific QDC information....3209
56.2 Overview....3209
56.3 Functional Description....3213
56.4 Signal Descriptions....3225
56.5 Memory Map and Registers....3226

Chapter 57
Watchdog Timer (WDOG1-2)

57.1 Chip-specific WDOG information....3249
57.2 Overview....3249
57.3 External signals....3251
57.4 Clocks....3251
57.5 Watchdog mechanism and system integration....3252
57.6 Functional description....3252
57.7 Initialization....3260
57.8 WDOG Memory Map/Register Definition....3261

Chapter 58
RTWDOG (WDOG3)

58.1 Chip-specific RTWDOG information....3269
58.2 Overview....3270
58.3 Functional description....3271
58.4 Application Information....3278
58.5 Memory map and register definition....3279

Chapter 59
External Watchdog Monitor (EWM)

59.1 Chip-specific EWM information....3287
59.2 Overview....3287
59.3 Functional Description....3289

59.4 EWM Signal Descriptions.... 3294
59.5 Memory Map/Register Definition....3294

Chapter 60
On Chip Cross Triggers Overview

60.1 Overview....3303

Chapter 61
Inter-Peripheral Crossbar Switch A (XBARA)

61.1 Chip-specific XBAR information....3305
61.2 Overview....3305
61.3 Functional Description....3307
61.4 External Signals.... 3309
61.5 Memory Map and Register Descriptions.... 3310

Chapter 62
Inter-Peripheral Crossbar Switch B (XBARB)

62.1 Chip-specific XBAR information....3351
62.2 Overview....3351
62.3 Functional Description....3352
62.4 External Signals.... 3352
62.5 Memory Map and Register Descriptions.... 3352

Chapter 63
And-Or-Inverter (AOI)

63.1 Chip-specific AOI information....3359
63.2 Overview....3359
63.3 Functional Description....3362
63.4 External Signal Description.... 3365
63.5 Memory Map and Register Descriptions.... 3365

Chapter 64
Analog Overview

64.1 Overview....3371

Chapter 65

Analog Comparator (ACMP)

65.1 Chip-specific CMP information....3373
65.2 Introduction....3373
65.3 Memory map/register definitions....3378
65.4 Functional description....3384
65.5 CMP interrupts....3398
65.6 DMA support....3398
65.7 Digital-to-analog converter....3399
65.8 DAC functional description....3399
65.9 DAC resets....3400
65.10 DAC clocks....3400
65.11 DAC interrupts....3400

Chapter 66
Analog-to-Digital Converter (ADC)

66.1 Chip-specific ADC information....3401
66.2 Overview....3401
66.3 External Signals....3405
66.4 Clocks....3407
66.5 Functional Description....3407
66.6 Initialization Information....3422
66.7 Application Information....3424
66.8 Memory map and register definition....3428

Chapter 67
ADC External Trigger Control (ADC_ETC)

67.1 Chip-specific ADC_ETC information....3447
67.2 Overview....3447
67.3 Functional description....3450
67.4 Initialization....3453
67.5 Application information....3453
67.6 Memory Map and register definition....3456

Chapter 68
Touch Screen Controller (TSC)

68.1 Chip-specific TSC information....3487
68.2 Overview....3487
68.3 Functional Description....3489
68.4 TSC Memory Map/Register Definition....3494

Chapter 1
About this Document

1.1 Audience

The reference manual is intended for the board-level product designers and product software developers. This manual assumes that the reader has a background in computer engineering and/or software engineering and understands the concepts of the digital system design, microprocessor architecture, input/output (I/O) devices, industry standard communication, and device interface protocols.

1.2 Organization

The reference manual describes the chip at a system level and provides an architectural overview. It also describes the system memory map, system-level interrupt events, external pins and pin multiplexing, external memory, system debug, system boot, multimedia subsystem, power management, and system security.

1.3 Suggested Reading

This section lists additional resources that provide background for the information in the reference manual, as well as general information about the architecture.

Table 1-1. Suggested Reading

Type	Description	Resource
Fact Sheet	The Fact Sheet is an overview of the product key features and its uses.	i.MX RT Series Crossover Processor Fact Sheet
Reference Manual	The Reference Manual contains a comprehensive description of the structure and function (operation) of a device.	This document
Data Sheet	The Data Sheet includes electrical characteristics and signal connections.	i.MX RT1060 Data Sheet - Industrial Products

Table continues on the next page...

Table 1-1. Suggested Reading (continued)

Type	Description	Resource
		i.MX RT1060 Data Sheet - Consumer Productsi.MX RT1060 Data Sheet Supplement
Chip Errata	The Chip mask set Errata provides additional or corrective information for a particular device mask set.	i.MX RT1060 Chip Errata
Application Notes	Provides additional information about particular device feature or function.	
	AN12419: Secure JTAG for i.MXRT10xx	AN12419
	AN12085: How to use i.MX RT Low Power Feature	AN12085
	AN12077: Using the i.MX RT FlexRAM	AN12077
	Other Application Notes	Other i.MX RT1060 Application Notes
Web Sites	Product summary page on nxp.com with documentation, software, and resources for the device.	i.MX RT1060 Product Summary Page
	Product documentation page on nxp.com with a list of all documentation related to the device.	i.MX RT1060 Documentation
Community Forum	i.MX RT Community support forum for questions, support, and information about the device.	i.MX RT Community

1.4 Conventions

The reference manual uses the following notational conventions:

cleared / set

When a bit has a value of zero, it is said to be cleared; when it has a value of one, it is said to be set.

mnemonics

Instruction mnemonics are shown in lowercase bold.

italics

Italics indicate variable command parameters, for example, **bcctr***x*.

The book titles in the text are set in italics.

15

An integer in decimal.

0x

the prefix to denote a hexadecimal number.

0b

The prefix to denote a binary number. Binary values of 0 and 1 are written without a prefix.

n'H4000CA00

The n-bit hexadecimal number.

BLK_REG_NAME

The register names are all uppercase. The block mnemonic is prepended with an underscore delimiter (_).

BLK_REG[FIELD]

The fields within registers appear in brackets. For example, ESR[RLS] refers to the Receive Last Slot field of the ESAI Status Register.

BLK_REG[*n*]

The bit number *n* within the BLK.REG register.

BLK_REG[*l:r*]

The register bit ranges. The ranges are indicated by the left-most bit number *l* and the right-most bit number *r*, separated by a colon (:). For example, ESR[15:0] refers to the lower half word in the ESAI Status Register.

x, U

In some contexts, such as signal encodings, an unitalicized x indicates a "don't care" or "uninitialized". The binary value can be 1 or 0.

x

An italicized *x* indicates an alphanumeric variable.

n, m

Italicized *n or m* represent integer variables.

!

Binary logic operator NOT.

&&

Binary logic operator AND.

||

Binary logic operator OR.

^ or <O+>

Binary logic operator XOR. For example, A <O+> B.

|

Bit-wise OR. For example, 0b0001 | 0b1000 yields the value of 0b1001.

&

Bit-wise AND. For example, 0b0001 & 0b1000 yields the value of 0b0000.

{A,B}

Concatenation, where the *n*-bit value A is prepended to the *m*-bit value B to form an (*n* +*m*)-bit value. For example, {0, REG*m* [14:0]} yeilds a 16-bit value with 0 in the most significant bit.

- or grey fill

Indicates a reserved bit field in a register. Although these bits can be written to with ones or zeros, they always read zeros.

>>

Shift right logical one position.

<<
    Shift left logical one position.
<=
    Assignment.
==
    Compare equal.
!=
    Compare not equal.
>
    Greater than.
<
    Less than.

1.5 Register Access

1.5.1 Register Diagram Field Access Type Legend

This figure provides the interpretation of the notation used in the register diagrams for a number of common field access types:

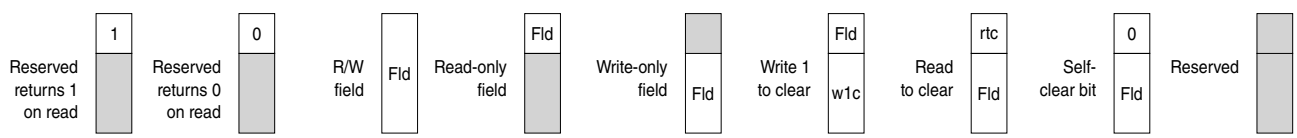


Figure 1-1. Register Field Conventions

NOTE

For reserved register fields, the software should mask off the data in the field after a read (the software can't rely on the contents of data read from a reserved field) and always write all zeros.

1.5.2 Register Macro Usage

A common operation is to update one field without disturbing the contents of the remaining fields in the register. Normally, this requires a read-modify-write (RMW) operation, where the CPU reads the register, modifies the target field, then writes the results back to the register. This is an expensive operation in terms of CPU cycles, because of the initial register read.

To address this issue, some hardware registers are implemented as a group, including registers that can be used to either set, clear, or toggle (SCT) individual bits of the primary register. When writing to an SCT register, all the bits set to 1 perform the associated operation on the primary register, while the bits set to 0 are not affected. The SCT registers always read back 0, and should be considered write-only. The SCT registers are not implemented if the primary register is read-only.

With this architecture, it is possible to update one or more fields using only register writes. First, all bits of the target fields are cleared by a write to the associated clear register, then the desired value of the target fields is written to the set register. This sequence of two writes is referred to as a clear-set (CS) operation.

A CS operation does have one potential drawback. Whenever a field is modified, the hardware sees a value of 0 before the final value is written. For most fields, passing through the 0 state is not a problem. Nonetheless, this behavior is something to consider when using a CS operation.

Also, a CS operation is not required for fields that are one-bit wide. While the CS operation works in this case, it is more efficient to simply set or clear the target bit (that is, one write instead of two). A simple set or clear operation is also atomic, while a CS operation is not.

Note that not all macros for set, clear, or toggle (SCT) are atomic. For registers that do not provide hardware support for this functionality, these macros are implemented as a sequence of read-modify-write operations. When an atomic operation is required, the developer should pay attention to this detail, because unexpected behavior might result if an interrupt occurs in the middle of the critical section comprising the update sequence.

A set of SCT registers is offered for registers in many modules on this device, as described in this manual. In a module memory map table, the suffix _SET, _CLR, or _TOG is added to the base name of the register. For example, the CCM_ANALOG_PLL_ARM register has three other registers called CCM_ANALOG_PLL_ARM_SET, CCM_ANALOG_PLL_ARM_CLR, and CCM_ANALOG_PLL_ARM_TOG.

In the sub-section that describes one of these sets of registers, a short-hand convention is used to denote that a register has the SCT register set. There is an italicized *n* appended to the end of the short register name. Using the above example, the name used for this register is CCM_ANALOG_PLL_ARM*n*. When you see this designation, there is a SCT register set associated with the register, and you can verify this by checking it in the memory map table. The address offset for each of these registers is given in the form of the following example:

Address: 20C_8000h base + 0h offset + (4d × i), where i=0d to 3d

In this example, the address for each of the base registers and their three SCT registers can be calculated as:

Register	Address
CCM_ANALOG_PLL_ARM	20C_8000h
CCM_ANALOG_PLL_ARM_SET	20C_8004h
CCM_ANALOG_PLL_ARM_CLR	20C_8008h
CCM_ANALOG_PLL_ARM_TOG	20C_800Ch

1.6 Acronyms and Abbreviations

The table below contains acronyms and abbreviations used in this document.

Acronyms and Abbreviated Terms

Term	Meaning
ACMP	Analog Comparator
ADC	Analog-to-Digital Converter
AHB	Advanced High-performance Bus
AIPS	Arm IP Bus
ALU	Arithmetic Logic Unit
AMBA	Advanced Microcontroller Bus Architecture
APB	Advanced Peripheral Bus
ASRC	Asynchronous Sample Rate Converter
AXI	Advanced eXtensible Interface
BEE	Bus Encryption Engine
CA/CM	Arm Cortex-A/Cortex-M
CAN	Controller Area Network
CCM	Clock Controller Module
CM7	Arm Cortex M7 Core
CPU	Central Processing Unit
CSI	CMOS Sensor Interface
CSU	Central Security Unit
CTI	Cross Trigger Interface
DAP	Debug Access Port
DCP	Data Co-Processor
DDR	Double data rate
DMA	Direct memory access
DPLL	Digital phase-locked loop
DRAM	Dynamic random access memory

Table continues on the next page...

Term	Meaning
ECC	Error correcting codes
LPSPI	Low-power SPI
EDMA	Enhanced Direct Memory Access
EIM	External Interface Module
ENET	Ethernet
EPIT	Enhanced Periodic Interrupt Timer
EPROM	Erasable Programmable Read-Only Memory
ETF	Embedded Trace FIFO
ETM	Embedded Trace Macrocell
FIFO	First-In-First-Out
GIC	General Interrupt Controller
GPC	General Power Controller
GPIO	General-Purpose I/O
GPR	General-Purpose Register
GPS	Global Positioning System
GPT	General-Purpose Timer
GPU	Graphics Processing Unit
GPV	Global Programmers View
HAB	High-Assurance Boot
I2C or I^2C	Inter-Integrated Circuit
IC	Integrated Circuit
IEEE	Institute of Electrical and Electronics Engineers
IOMUX	Input-Output Multiplexer
IP	Intellectual Property
IrDA	Infrared Data Association
JTAG	Joint Test Action Group (a serial bus protocol usually used for test purposes)
ELCDIF	Liquid Crystal Display Interface
LDO	Low-Dropout
LIFO	Last-In-First-Out
LRU	Least-Recently Used
LSB	Least-Significant Byte
LUT	Look-Up Table
LVDS	Low Voltage Differential Signaling
MAC	Medium Access Control
MCM	Miscellaneous Control Module
MMC	Multimedia Card
MSB	Most-Significant Byte
MT/s	Mega Transfers per second
OCRAM	On-Chip Random-Access Memory
OCOTP	On-Chip One-Time Programmable Controller
PCI	Peripheral Component Interconnect

Table continues on the next page...

Term	Meaning
PCIe	PCI express
PGC	Power Gating Controller
PIC	Programmable Interrupt Controller
PMU	Power Management Unit
POR	Power-On Reset
PSRAM	Pseudo-Static Random Access Memory
PWM	Pulse Width Modulation
PXP	Pixel Pipeline
QoS	Quality of Service
R2D	Radians to Degrees
RISC	Reduced Instruction Set Computing
ROM	Read-Only Memory
RTOS	Real-Time Operating System
Rx	Receive
SAI	Synchronous Audio Interface
SD	Secure Digital
SDIO	Secure Digital Input/Output
SDLC	Synchronous Data Link Control
SDMA	Smart DMA
SIM	Subscriber Identification Module
SNVS	Secure Non-Volatile Storage
SoC	System-on-Chip
SPBA	Shared Peripheral Bus Arbiter
SPDIF	Sony Philips Digital Interface
SPI	Serial Peripheral Interface
SRAM	Static Random-Access Memory
SRC	System Reset Controller
TFT	Thin-Film Transistor
TPIU	Trace Port Interface Unit
TSGEN	Time Stamp Generator
Tx	Transmit
TZASC	TrustZone Address Space Controller
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus
USDHC	Ultra Secured Digital Host Controller
WDOG	Watchdog
WLAN	Wireless Local Area Network
WXGA	Wide Extended Graphics Array

Chapter 2
Introduction

2.1 Introduction

The i.MX RT1060 is a new processor family featuring NXP's advanced implementation of the high performance Arm Cortex®-M7 Core. It offers high-performance processing optimized for lowest power consumption and best real-time response.

This device provides various memory interfaces, including SDRAM, Raw NAND FLASH, NOR FLASH, SD/eMMC, Quad SPI (FlexSPI), and a wide range of other interfaces for connecting peripherals, such as WLAN, Bluetooth™,displays, camera sensors, and GPS. Same as other i.MX processors, this i.MX RT series also has rich audio and video features, including LCD display, basic 2D graphics, camera interface, SPDIF and I2S audio interface.

2.1.1 Block Diagram

The functional block diagram is shown in the figure below. This diagram provides a view of the chip's major functional components and core complexes.

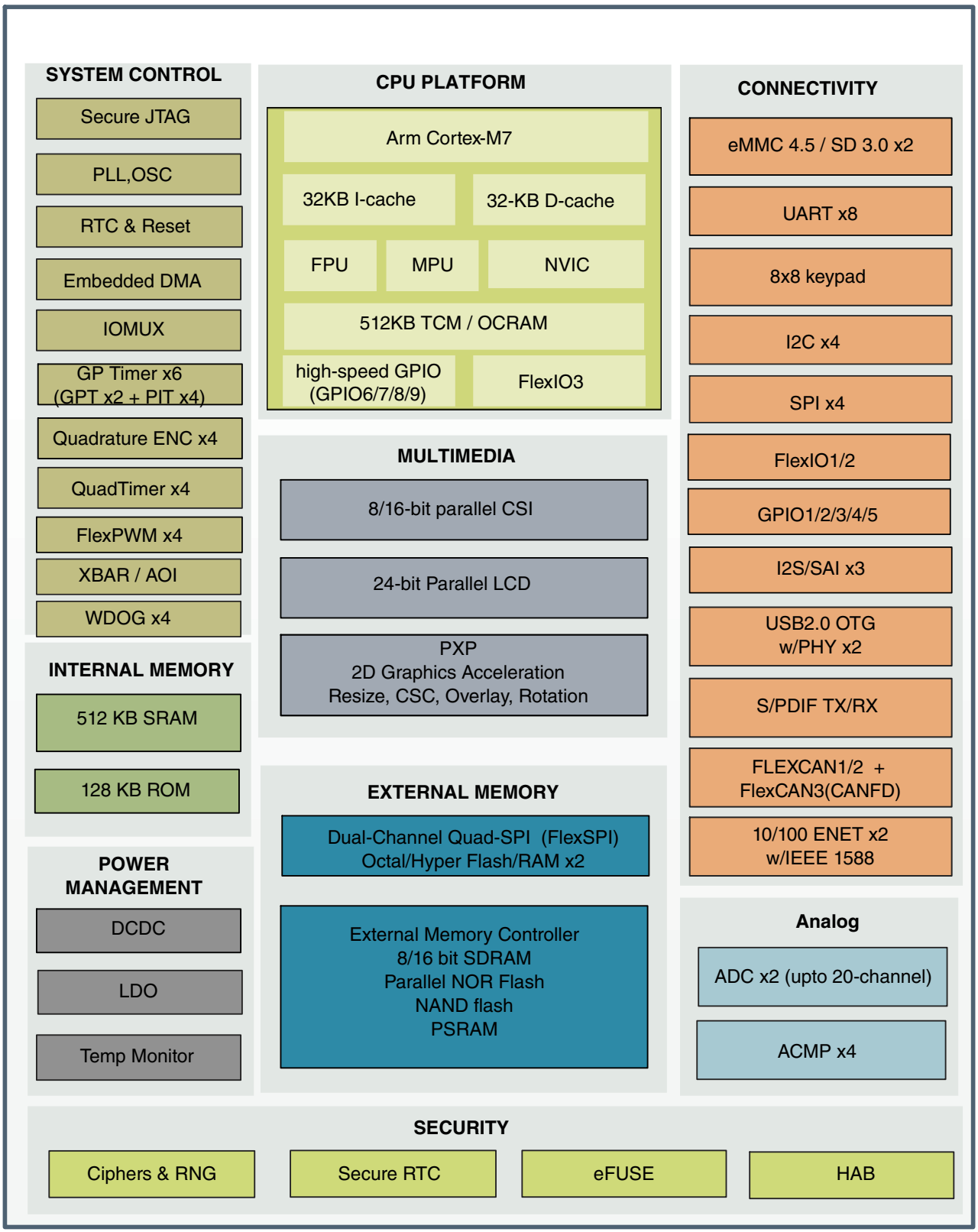


Figure 2-1. Simplified Block Diagram

2.2 Features

The i.MX RT1060 processors are based on Arm®Cortex®-M7 Platform, which have the following features:

Single Arm Cortex-M7 with:
- 32KB L1 Instruction Cache
- 32KB L1 Data Cache
- Single-precision and double-precision FPU (Floating Point Unit)
- Integrated Memory Protection Unit (MPU), up to 16 individual protection regions
- Tightly coupled GPIOs (GPIO6/7/8/9), operating at the same frequency as Arm
- Up to 512KB I-TCM and D-TCM in total

On Chip Memory:
- Boot ROM (128KB)
- On-chip RAM, configurable RAM up to 1MB (512KB OCRAM shared between ITCM/DTCM and OCRAM, as well as dedicated 512KB OCRAM)

External Memory Interfaces:
- 8/16-bit SDRAM, up to SDRAM-166
- 8-bit SLC NAND FLASH, with ECC handled in software
- SD/eMMC
- ×2 Single/Dual channel Quad SPI FLASH with XIP support
- Parallel NOR FLASH with XIP support
- SPI NOR/NAND FLASH

Graphics:
- Generic 2D Graphics engine (PXP)
  - BitBlit
  - Flexible image composition options – alpha, chroma key
  - Image rotation (90, 180, 270 degree rotation)
  - Porter-Duff operation
  - Image resize
  - Color space conversion
  - Multiple pixel format support (RGB, YUV444, YUV422, YUV420, YUV400)
  - Standard 2D-DMA operation

Display Interface:
- Parallel RGB LCD interface
  - Support 8-/16-/24-bit interface
  - Support up to WXGA resolution

- Smart LCD Display with 8/16-bit MPU/8080 interface
- Support Index color with 256 entry × 24 bit color LUT

Camera Sensor Interface:
- Support 8/16/24-bit CSI Input

Audio:
- SPDIF Input and Output
- 3x SAI (synchronous audio interface) modules supporting I2S, AC97, TDM, and Codec/DSP interfaces
- MQS interface for medium quality audio via GPIO pads

Connectivity:
- 2x USB 2.0 OTG controller with integrated PHY interfaces
- 2x Ultra Secure Digital Host Controller (uSDHC) interfaces
- 2× 10M/100M Ethernet controller with support for IEEE1588
- 8x Universal asynchronous receiver/transmitter (UARTs) modules
- 4x I2C modules
- 4x SPI modules
- 2x FlexCAN modules
- 1x FlexCAN module with Flexible Data-rate support (CANFD/FlexCAN3)
- 3x FlexIO modules

Timers:
- 2x General Programmable Timer (GPT)
- 4-channel Periodical Interrupt Timer (PIT)
- 4x Quad Timer (QTimer)
- 4x FlexPWM

Analog:
- 2x Analog-Digital-Converters (ADC) (up to 20 channels)
- 4x Analog Comparators (ACMP)

Security:
- High Assurance Boot (HAB)
- Data Co-Processor (DCP)
  - AES-128, ECB, and CBC mode
  - SHA-1 and SHA-256
  - CRC-32
- Bus Encryption Engine (BEE)
  - AES-128, ECB, and CTR mode
  - On-the-fly QSPI Flash decryption
- True Random Number Generator (TRNG)
- Secure Non-volatile Storage (SNVS)

    - Secure real-time clock (RTC)
- Secure JTAG Controller (SJC)

System Debug:
- Arm CoreSight debug and trace architecture
- Trace Port Interface Unit (TPIU) to support off-chip real-time trace
- Cross Triggering Interface (CTI)
- Support for 5-pin (JTAG) and SWD debug interfaces

Power Management:
- Full PMIC integration, including on-chip DCDC and LDO
- Temperature sensor with programmable trim points
- GPC hardware power management controller

2.3 Target Applications

This processor can be used in areas such as industrial HMI, IoT, motor control and home appliances, etc.

The architecture's flexibility enables it to be used in a wide variety of other general embedded applications too. This processor provides all interfaces necessary to connect peripherals such as WLAN, Bluetooth™, GPS, camera sensors, and multiple displays.

2.4 Endianness Support

The chip supports only the Little Endian mode.

Chapter 3
Memory Maps

3.1 Memory system overview

This section introduces the memory architecture of the chip.

NOTE

Accessing the reserved memory regions can result in unpredictable behavior.

3.2 Arm Platform Memory Map

The chip memory map has been provided in the following tables.

Table 3-1. System memory map (CM7)

Start Address	End Address	Size	Description
E010_0000	FFFF_FFFF	511MB	Reserved
E000_0000	E00F_FFFF	1MB	CM7 PPB ARM peripherals
8000_0000	DFFF_FFFF	1.5GB	SEMC external memories (SDRAM, NOR, PSRAM, NAND and 8080) shared memory space
7FC0_0000	7FFF_FFFF	4MB	FlexSPI RX FIFO
7F80_0000	7FBF_FFFF	4MB	FlexSPI TX FIFO
7F40_0000	7F7F_FFFF	4MB	FlexSPI2 RX FIFO
7F00_0000	7F3F_FFFF	4MB	FlexSPI2 TX FIFO
7000_0000	7EFF_FFFF	240MB	FlexSPI2/ FlexSPI2 ciphertext PSRAM (EXTMEM)
6000_0000	6FFF_FFFF	256MB	FlexSPI/ FlexSPI ciphertext Flash (PROGMEM, FLASHMEM)
4800_0000	5FFF_FFFF	384MB	Reserved
4400_0000	47FF_FFFF	64MB	Reserved
4200_00004210_0000	43FF_FFFF	32MB31MB	Reserved
4200_0000	420F_FFFF	1MB	AIPS-5 Low latency peripherals (sadly, no DMA access)

Table continues on the next page...

Table 3-1. System memory map (CM7) (continued)

Start Address	End Address	Size	Description
4180_0000	41FF_FFFF	8MB	Reserved
4170_0000	417F_FFFF	1MB	GPV Reserved
4160_0000	416F_FFFF	1MB	GPV Reserved
4150_0000	415F_FFFF	1MB	GPV Reserved
4140_0000	414F_FFFF	1MB	"cpu" configuration port
4130_0000	413F_FFFF	1MB	Reserved for "ems" GPV
4120_0000	412F_FFFF	1MB	Reserved for "per" GPV
4110_0000	411F_FFFF	1MB	"m" configuration port
4100_0000	410F_FFFF	1MB	"main" configuration port
4040_0000	40FF_FFFF	12MB	Reserved
4030_0000	403F_FFFF	1MB	AIPS-4 Normal peripherals on AIPS1-4
4020_0000	402F_FFFF	1MB	AIPS-3 (DMA can access anything here)
4010_0000	401F_FFFF	1MB	AIPS-2
4000_0000	400F_FFFF	1MB	AIPS-1
3000_0000	3FFF_FFFF	256MB	Reserved
2040_0000	2FFF_FFFF	252MB	Reserved
2030_0000	203F_FFFF	1MB	OCRAM Reserved
2028_0000	202F_FFFF	512KB	OCRAM - FlexRAM
2020_0000	2027_FFFF	512KB	OCRAM2 RAM2 memory (DMAMEM, malloc / new)
2010_0000	201F_FFFF	1MB	Reserved
2008_0000	200F_FFFF	512KB	DTCM Reserved
2000_0000	2007_FFFF	512KB	DTCM RAM1 - Variables default location
1000_0000	1FFF_FFFF	256MB	SEMC (Aliased)Reserved
0800_0000	0FFF_FFFF	128MB	FlexSPI (Aliased)Reserved
0040_0000	07FF_FFFF	124MB	Reserved
0028_0000	003F_FFFF	1536KB	Reserved
0021_80000022_0000	0027_FFFF	416KB384KB	Reserved
0020_0000	0021_7FFF0021_FFFF	96KB128KB	Reserved
0010_0000	001F_FFFF	1MB	ITCM Reserved
0008_0000	000F_FFFF	512KB	ITCM Reserved
0000_0000	0007_FFFF	512KB	ITCM RAM1 - Code default location

Teensy startup code automatically partitions RAM1 between ITCM and DTCM depending on compiled code size, by using IOMUXC_GPR_GPR17 (page 363).

The table below shows the Arm IP Bus (AIPS) detailed memory map.

Table 3-2. AIPS-1 memory map

Start Address	End Address	Region	Size	NIC Port
400F_C000	400F_FFFF	AIPS-1	16KB	CCM(CCM)

Table continues on the next page...

Table 3-2. AIPS-1 memory map (continued)

Start Address	End Address	Region	Size	NIC Port
400F_8000	400F_BFFF		16KB	SRC(SRC)
400F_4000	400F_7FFF		16KB	GPC
400F_0000	400F_3FFF		16KB	Reserved
400E_C000	400E_FFFF		16KB	DMA_CH_MUX
400E_8000	400E_BFFF		16KB	EDMA
400E_4000	400E_7FFF		16KB	SJC
400E_0000	400E_3FFF		16KB	TSC_DIG
400D_C000	400D_FFFF		16KB	CSU
400D_8000	400D_BFFF		16KB	ANALOG
400D_4000	400D_7FFF		16KB	SNVS_HP
400D_0000	400D_3FFF		16KB	WDOG2
400C_C000	400C_FFFF		16KB	TRNG
400C_8000	400C_BFFF		16KB	ADC2
400C_4000	400C_7FFF		16KB	ADC1
400C_0000	400C_3FFF		16KB	GPIO5
400B_C000	400B_FFFF		16KB	WDOG3
400B_8000	400B_BFFF		16KB	WDOG1
400B_4000	400B_7FFF		16KB	EWM
400B_0000	400B_3FFF		16KB	CM7_MXRT (FLEXRAM)
400A_C000	400A_FFFF		16KB	IOMUXC_GPR
400A_8000	400A_BFFF		16KB	IOMUXC_SNVS
400A_4000	400A_7FFF		16KB	IOMUXC_SNVS_GPR
400A_0000	400A_3FFF		16KB	Reserved
4009_C000	4009_FFFF		16KB	Reserved
4009_8000	4009_BFFF		16KB	Reserved
4009_4000	4009_7FFF		16KB	ACMP
4009_0000	4009_3FFF		16KB	Reserved
4008_C000	4008_FFFF		16KB	Reserved
4008_8000	4008_BFFF		16KB	Reserved
4008_4000	4008_7FFF		16KB	PIT
4008_0000	4008_3FFF		16KB	DCDC
4007_C000	4007_FFFF		16KB	AIPS-1 Configuration
4004_0000	4007_BFFF		240KB	Reserved
4000_0000	4003_FFFF		256KB	Reserved

The table below shows the AIPS-2 detailed memory map.

Table 3-3. AIPS-2 memory map

Start Address	End Address	Region	Size	NIC Port
401F_C000	401F_FFFF	AIPS-2	16KB	KPP
401F_8000	401F_BFFF		16KB	IOMUXC
401F_4000	401F_7FFF		16KB	OCOTP
401F_0000	401F_3FFF		16KB	GPT2
401E_C000	401E_FFFF		16KB	GPT1
401E_8000	401E_BFFF		16KB	QTimer4
401E_4000	401E_7FFF		16KB	QTimer3
401E_0000	401E_3FFF		16KB	QTimer2
401D_C000	401D_FFFF		16KB	QTimer1
401D_8000	401D_BFFF		16KB	CANFD (FlexCAN3)
401D_4000	401D_7FFF		16KB	FlexCAN2
401D_0000	401D_3FFF		16KB	FlexCAN1
401C_C000	401C_FFFF		16KB	Reserved
401C_8000	401C_BFFF		16KB	Reserved
401C_4000	401C_7FFF		16KB	GPIO4
401C_0000	401C_3FFF		16KB	GPIO3
401B_C000	401B_FFFF		16KB	GPIO2
401B_8000	401B_BFFF		16KB	GPIO1
401B_4000	401B_7FFF		16KB	Reserved
401B_0000	401B_3FFF		16KB	FlexIO2
401A_C000	401A_FFFF		16KB	FlexIO1
401A_8000	401A_BFFF		16KB	Reserved
401A_4000	401A_7FFF		16KB	Reserved
401A_0000	401A_3FFF		16KB	LPUART8
4019_C000	4019_FFFF		16KB	LPUART7
4019_8000	4019_BFFF		16KB	LPUART6
4019_4000	4019_7FFF		16KB	LPUART5
4019_0000	4019_3FFF		16KB	LPUART4
4018_C000	4018_FFFF		16KB	LPUART3
4018_8000	4018_BFFF		16KB	LPUART2
4018_4000	4018_7FFF		16KB	LPUART1
4018_0000	4018_3FFF		16KB	Reserved
4017_C000	4017_FFFF		16KB	AIPS-2 Configuration
4014_0000	4017_BFFF		240KB	Reserved
4010_0000	4013_FFFF		256KB	Reserved

The table below shows the AIPS-3 detailed memory map.

Table 3-4. AIPS-3 memory map

Start Address	End Address	Region	Size	NIC Port
402F_C000	402F_FFFF	AIPS-3	16KB	DCP
402F_8000	402F_BFFF		16KB	Reserved
402F_4000	402F_7FFF		16KB	Reserved
402F_0000	402F_3FFF		16KB	SEMC
402E_C000	402E_FFFF		16KB	Reserved
402E_8000	402E_BFFF		16KB	Reserved
402E_4000	402E_7FFF		16KB	Reserved
402E_0000	402E_3FFF		16KB	USB(USB)
402D_C000	402D_FFFF		16KB	Reserved
402D_8000	402D_BFFF		16KB	ENET
402D_4000	402D_7FFF		16KB	ENET2
402D_0000	402D_3FFF		16KB	Reserved
402C_C000	402C_FFFF		16KB	Reserved
402C_8000	402C_BFFF		16KB	Reserved
402C_4000	402C_7FFF		16KB	USDHC2
402C_0000	402C_3FFF		16KB	USDHC1
402B_C000	402B_FFFF		16KB	CSI
402B_8000	402B_BFFF		16KB	LCDIF
402B_4000	402B_7FFF		16KB	PXP
402B_0000	402B_3FFF		16KB	Reserved
402A_C000	402A_FFFF		16KB	Reserved
402A_8000	402A_BFFF		16KB	FlexSPI
402A_4000	402A_7FFF		16KB	ReservedFlexSPI2
402A_0000	402A_3FFF		16KB	Reserved
4029_C000	4029_FFFF		16KB	Reserved
4029_8000	4029_BFFF		16KB	Reserved
4029_4000	4029_7FFF		16KB	Reserved
4029_0000	4029_3FFF		16KB	Reserved
4028_C000	4028_FFFF		16KB	Reserved
4028_8000	4028_BFFF		16KB	Reserved
4028_4000	4028_7FFF		16KB	Reserved
4028_0000	4028_3FFF		16KB	Reserved
4027_C000	4027_FFFF		16KB	AIPS-3 Configuration
4024_0000	4027_BFFF		240KB	Reserved
4020_0000	4023_FFFF		256KB	Reserved

The table below shows the AIPS-4 detailed memory map.

Table 3-5. AIPS-4 memory map

Start Address	End Address	Region	Size	NIC Port
403F_C000	403F_FFFF	AIPS-4	16KB	LPI2C4
403F_8000	403F_BFFF		16KB	LPI2C3
403F_4000	403F_7FFF		16KB	LPI2C2
403F_0000	403F_3FFF		16KB	LPI2C1
403E_C000	403E_FFFF		16KB	BEE
403E_8000	403E_BFFF		16KB	FLEXPWM4
403E_4000	403E_7FFF		16KB	FLEXPWM3
403E_0000	403E_3FFF		16KB	FLEXPWM2
403D_C000	403D_FFFF		16KB	FLEXPWM1
403D_8000	403D_BFFF		16KB	Reserved
403D_4000	403D_7FFF		16KB	QDC4
403D_0000	403D_3FFF		16KB	QDC3
403C_C000	403C_FFFF		16KB	QDC2
403C_8000	403C_BFFF		16KB	QDC1
403C_4000	403C_7FFF		16KB	XBAR3
403C_0000	403C_3FFF		16KB	XBAR2
403B_C000	403B_FFFF		16KB	XBAR1
403B_8000	403B_BFFF		16KB	AOI2
403B_4000	403B_7FFF		16KB	AOI1
403B_0000	403B_3FFF		16KB	ADC_ETC
403A_C000	403A_FFFF		16KB	Reserved
403A_8000	403A_BFFF		16KB	Reserved
403A_4000	403A_7FFF		16KB	Reserved
403A_0000	403A_3FFF		16KB	LPSPI4
4039_C000	4039_FFFF		16KB	LPSPI3
4039_8000	4039_BFFF		16KB	LPSPI2
4039_4000	4039_7FFF		16KB	LPSPI1
4039_0000	4039_3FFF		16KB	Reserved
4038_C000	4038_FFFF		16KB	SAI3
4038_8000	4038_BFFF		16KB	SAI2
4038_4000	4038_7FFF		16KB	SAI1
4038_0000	4038_3FFF		16KB	SPDIF
4037_C000	4037_FFFF		16KB	AIPS-4 Configuration
4034_0000	4037_BFFF		240KB	Reserved
4030_0000	4033_FFFF		256KB	Reserved

The table below shows the AIPS-5 detailed memory map.

Table 3-6. AIPS-5 memory map No DMA access

Start Address	End Address	Region	Size	NIC Port
4208_0000	420F_FFFF	AIPS-5	512KB	Reserved off platform slots
4207_C000	4207_FFFF		16KB	Reserved
4207_8000	4207_BFFF		16KB	Reserved
4207_4000	4207_7FFF		16KB	Reserved
4207_0000	4207_3FFF		16KB	Reserved
4206_C000	4206_FFFF		16KB	Reserved
4206_8000	4206_BFFF		16KB	Reserved
4206_4000	4206_7FFF		16KB	Reserved
4206_0000	4206_3FFF		16KB	Reserved
4205_C000	4205_FFFF		16KB	Reserved
4205_8000	4205_BFFF		16KB	Reserved
4205_4000	4205_7FFF		16KB	Reserved
4205_0000	4205_3FFF		16KB	Reserved
4204_C000	4204_FFFF		16KB	Reserved
4204_8000	4204_BFFF		16KB	Reserved
4204_4000	4204_7FFF		16KB	Reserved
4204_0000	4204_3FFF		16KB	Reserved
4203_C000	4203_FFFF		16KB	Reserved
4203_8000	4203_BFFF		16KB	Reserved
4203_4000	4203_7FFF		16KB	Reserved
4203_0000	4203_3FFF		16KB	Reserved
4202_C000	4202_FFFF		16KB	Reserved
4202_8000	4202_BFFF		16KB	Reserved
4202_4000	4202_7FFF		16KB	Reserved
4202_0000	4202_3FFF		16KB	FlexIO3
4201_C000	4201_FFFF		16KB	Reserved
4201_8000	4201_BFFF		16KB	Reserved
4201_4000	4201_7FFF		16KB	Reserved
4201_0000	4201_3FFF		16KB	Reserved
4200_C000	4200_FFFF		16KB	GPIO9
4200_8000	4200_BFFF		16KB	GPIO8
4200_4000	4200_7FFF		16KB	GPIO7
4200_0000	4200_3FFF		16KB	GPIO6

The table below shows the PPB detailed memory map.

Table 3-7. PPB memory map

Start Address	End Address	Size	Allocation
E00F_F000	E00F_FFFF	4KB	PPB ROM
E00F_E000	E00F_EFFF	4KB	Processor ROM
E00F_D000	E00F_DFFF	4KB	SYS ROM
E00F_0000	E00F_CFFF	52KB	PPB Reserved
E008_1000	E00E_FFFF	444KB	PPB Reserved
E008_0000	E008_0FFF	4KB	MCM
E004_5000	E007_FFFF	236KB	PPB Reserved
E004_4000	E004_4FFF	4KB	PPB RES
E004_3000	E004_3FFF	4KB	TSGEN
E004_2000	E004_2FFF	4KB	CTI
E004_1000	E004_1FFF	4KB	ETM
E004_0000	E004_0FFF	4KB	TPIU

NOTE

Accessing the reserved memory regions can result in unpredictable behavior.

Chapter 4
Interrupts, DMA Events, and XBAR Assignments

4.1 Chip-specific Interrupt information

Table 4-1. Reference links to related information

Topic	Related module or subsystem	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Audio Subsystem	Audio Subsystem	Audio Subsystem Overview
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
Interrupts, DMA Events and XBAR Assignments	-	Interrupts, DMA Events and XBAR Assignments

4.2 Overview

This section describes the Interrupt assignments, DMA events, and XBAR resource assignments.

4.3 CM7 interrupts

The Nested Vectored Interrupt Controller (NVIC) collects interrupt request sources and provides an interface to the Cortex-M7 core.

The table below describes the Cortex-M7 interrupt sources:

NOTE

Due to the clock frequency difference between CPU and peripheral, in some corner case, peripheral interrupt flag may not be really cleared before CPU exit ISR. In such case, user should add DSB instruction right after instruction to clear interrupt flag.

Table 4-2. CM7 domain interrupt summary

IRQ	Module	Logic	Description
0	eDMA	OR	eDMA Channel 0 Transfer Complete
			eDMA Channel 16 Transfer Complete
1		OR	eDMA Channel 1 Transfer Complete
			eDMA Channel 17 Transfer Complete
2		OR	eDMA Channel 2 Transfer Complete
			eDMA Channel 18 Transfer Complete
3		OR	eDMA Channel 3 Transfer Complete
			eDMA Channel 19 Transfer Complete
4		OR	eDMA Channel 4 Transfer Complete
			eDMA Channel 20 Transfer Complete
5		OR	eDMA Channel 5 Transfer Complete
			eDMA Channel 21 Transfer Complete
6		OR	eDMA Channel 6 Transfer Complete
			eDMA Channel 22 Transfer Complete
7		OR	eDMA Channel 7 Transfer Complete
			eDMA Channel 23 Transfer Complete
8		OR	eDMA Channel 8 Transfer Complete
			eDMA Channel 24 Transfer Complete
9		OR	eDMA Channel 9 Transfer Complete
			eDMA Channel 25 Transfer Complete
10		OR	eDMA Channel 10 Transfer Complete
			eDMA Channel 26 Transfer Complete
11		OR	eDMA Channel 11 Transfer Complete
			eDMA Channel 27 Transfer Complete
12		OR	eDMA Channel 12 Transfer Complete
			eDMA Channel 28 Transfer Complete
13		OR	eDMA Channel 13 Transfer Complete
			eDMA Channel 29 Transfer Complete
14		OR	eDMA Channel 14 Transfer Complete
			eDMA Channel 30 Transfer Complete
15		OR	eDMA Channel 15 Transfer Complete
			eDMA Channel 31 Transfer Complete
16		-	Error Interrupt, Channels 0-15 / 16-31

Table continues on the next page...

Table 4-2. CM7 domain interrupt summary (continued)

IRQ	Module	Logic	Description
17	CM7	-	CTI trigger outputs
18		-	CTI trigger outputs
19		-	Core Platform exception IRQ
20	LPUART1 Serial6	OR	UART1 TX interrupt
			UART1 RX interrupt
21	LPUART2 Serial3	OR	UART2 TX interrupt
			UART2 RX interrupt
22	LPUART3 Serial2 (Serial4 on MM)	OR	UART3 TX interrupt
			UART3 RX interrupt
23	LPUART4 Serial4 (Serial2 on MM)	OR	UART4 TX interrupt
			UART4 RX interrupt
24	LPUART5 Serial8	OR	UART5 TX interrupt
			UART5 RX interrupt
25	LPUART6 Serial1	OR	UART6 TX interrupt
			UART6 RX interrupt
26	LPUART7 Serial7	OR	UART7 TX interrupt
			UART7 RX interrupt
27	LPUART8 Serial5	OR	UART8 TX interrupt
		OR	UART8 RX interrupt
28	LPI2C1 Wire	OR	LPI2C1 Interrupt master async
			LPI2C1 Interrupt slave async
			LPI2C1 Interrupt master
			LPI2C1 Interrupt slave
29	LPI2C2 Wire3 on MM	OR	LPI2C2 Interrupt master async
			LPI2C2 Interrupt slave async
			LPI2C2 Interrupt master
			LPI2C2 Interrupt slave
30	LPI2C3 Wire1 (Wire2 on MM)	OR	LPI2C3 Interrupt master async
			LPI2C3 Interrupt slave async
			LPI2C3 Interrupt master
			LPI2C3 Interrupt slave
31	LPI2C4 Wire2 (Wire1 on MM)	OR	LPI2C4 Interrupt master async
			LPI2C4 Interrupt slave async
			LPI2C4 Interrupt master
			LPI2C4 Interrupt slave
32	LPSPI1 SPI2	-	LPSPI interrupt request line to the core
33	LPSPI2	-	LPSPI interrupt request line to the core
34	LPSPI3 SPI1	-	LPSPI interrupt request line to the core
35	LPSPI4 SPI	-	LPSPI interrupt request line to the core

Table continues on the next page...

Table 4-2. CM7 domain interrupt summary (continued)

IRQ	Module	Logic	Description
36	FLEXCAN1	-	Combined interrupt of ini_int_busoff, ini_int_error, ipi_int_mbor, ipi_int_rxwarning, ipi_int_txwarning and ipi_int_wakein, etc.
37	FLEXCAN2	-	Combined interrupt of ini_int_busoff, ini_int_error, ipi_int_mbor, ipi_int_rxwarning, ipi_int_txwarning and ipi_int_wakein, etc.
38	CM7	-	FlexRAM address out of range Or access hit IRQ
39	KPP	-	Keypad Interrupt
40	TSC_DIG	-	TSC interrupt
41	GPR_IRQ	-	Used to notify cores on exception condition while boot
42	LCDIF	-	LCDIF Sync Interrupt
43	CSI	-	CSI interrupt
44	PXP	-	PXP interrupt
45	WDOG2	-	Watchdog Timer reset
46	SNVS_HP_WRAPPER	-	SNVS Functional Interrupt
47		-	SNVS Security Interrupt
48	SNVS_LP_WRAPPER	OR	ON-OFF button press shorter than 5 secs (pulse event)
	SNVS_HP_WRAPPER		ON-OFF button press shorter than 5 secs (pulse event)
49	CSU	-	CSU Interrupt Request 1. Indicates to the processor that one or more alarm inputs were asserted
50	DCP	-	Combined DCP channel interrupts (except channel 0) and CRC interrupt
51		-	IRQ of DCP channel 0
52		-	Reserved
53	TRNG	-	TRNG interrupt
54	Reserved	-	Reserved
55	BEE	-	BEE IRQ
56	SAI1	OR	SAI RX interrupt
			SAI RX async interrupt
			SAI TX interrupt
			SAI TX async interrupt
57	SAI2	OR	SAI RX interrupt
			SAI RX async interrupt
			SAI TX interrupt
			SAI TX async interrupt
58	SAI3	-	SAI RX interrupt
59		-	SAI TX interrupt
60	SPDIF	OR	SPDIF Rx interrupt
			SPDIF Tx interrupt
61	PMU	-	Brown-out event on either the 1.1, 2.5 or 3.0 regulators.
62	Reserved	-	Reserved

Table continues on the next page...

Table 4-2. CM7 domain interrupt summary (continued)

IRQ	Module	Logic	Description
63	Temperature Monitor	OR	TempSensor low
			TempSensor high
64		-	TempSensor panic
65	USB PHY	-	USBPHY (UTMI0), Interrupt
66		-	USBPHY (UTMI1), Interrupt
67	ADC1	OR	ADC1 interrupt
			ADC1 async interrupt
68	ADC2	OR	ADC2 interrupt
			ADC2 async interrupt
69	DCDC	-	DCDC IRQ
70	Reserved	-	Reserved Used by Audio library
71	Reserved	-	Reserved
72	GPIO1	-	Interrupt from GPIO pin0
73		-	Interrupt from GPIO pin1
74		-	Interrupt from GPIO pin2
75		-	Interrupt from GPIO pin3
76		-	Interrupt from GPIO pin4
77		-	Interrupt from GPIO pin5
78		-	Interrupt from GPIO pin6
79		-	Interrupt from GPIO pin7
80		-	Combined interrupt indication for GPIO1 signal 0 through 15
81		-	Combined interrupt indication for GPIO1 signal 16 through 31
82	GPIO2	-	Combined interrupt indication for GPIO2 signal 0 through 15
83		-	Combined interrupt indication for GPIO2 signal 16 through 31
84	GPIO3	-	Combined interrupt indication for GPIO3 signal 0 through 15
85		-	Combined interrupt indication for GPIO3 signal 16 through 31
86	GPIO4	-	Combined interrupt indication for GPIO4 signal 0 through 15
87		-	Combined interrupt indication for GPIO4 signal 16 through 31
88	GPIO5	-	Combined interrupt indication for GPIO5 signal 0 through 15
89		-	Combined interrupt indication for GPIO5 signal 16 through 31
90	FLEXIO1	OR	FlexIO interrupt
			FlexIO asynchronous interrupt

Table continues on the next page...

Table 4-2. CM7 domain interrupt summary (continued)

IRQ	Module	Logic	Description
91	FLEXIO2	OR	FlexIO interrupt
			FlexIO asynchronous interrupt
92	WDOG1	-	Watchdog Timer reset
93	RTWDOG (WDOG3)	OR	Watchdog Timer reset
			Watchdog Timer Async reset
94	EWM	-	EWM IRQ
95	CCM	-	CCM, Interrupt Request 1
96		-	CCM, Interrupt Request 2
97	GPC	-	GPC, Interrupt Request
98	SRC	-	SRC interrupt request
99	Reserved	-	Reserved
100	GPT1	-	All interrupts combined. OR of GPT1 Rollover interrupt line, Input Capture 1 & 2 lines, Output Compare 1,2 &3 Interrupt lines
101	GPT2	-	All interrupts combined. OR of GPT2 Rollover interrupt line, Input Capture 1 & 2 lines, Output Compare 1,2 &3 Interrupt lines
102	FLEXPWM1	OR	capture PWM0 interrupt
			compare PWM0 interrupt
			reload PWM0 interrupt
103		OR	capture PWM1 interrupt
			compare PWM1 interrupt
			reload PWM1 interrupt
104		OR	capture PWM2 interrupt
			compare PWM2 interrupt
			reload PWM2 interrupt
105		OR	capture PWM3 interrupt
			compare PWM3 interrupt
			reload PWM3 interrupt
106		OR	fault interrupt
			reload error interrupt
107	FLEXSPI2	-	FlexSPI2 IRQ
108	FLEXSPI	-	FlexSPI IRQ
109	SEMC	-	SEMC
110	USDHC1	-	uSDHC1 Interrupt Request
111	USDHC2	-	uSDHC2 Interrupt Request
112	USB	-	USBO2 USB OTG2
113		-	USBO2 USB OTG1
114	ENET	OR	MAC 0 Periodic Timer Overflow
			MAC 0 Time Stamp Available
			MAC 0 Payload Receive Error

Table continues on the next page...

Table 4-2. CM7 domain interrupt summary (continued)

IRQ	Module	Logic	Description
			MAC 0 Transmit FIFO Underrun
			MAC 0 Collision Retry Limit
			MAC 0 Late Collision
			MAC 0 Ethernet Bus Error
			MAC 0 MII Data Transfer Done
			MAC 0 Receive Buffer Done
			MAC 0 Receive Frame Done
			MAC 0 Transmit Buffer Done
			MAC 0 Transmit Frame Done
			MAC 0 Graceful Stop
			MAC 0 Babbling Transmit Error
			MAC 0 Babbling Receive Error
			MAC 0 Wakeup Request (sync)
115		-	MAC 0 1588 Timer Interrupt – synchronous
116	XBAR1	OR	XBAR IRQ0
			XBAR IRQ1
117		OR	XBAR IRQ2
			XBAR IRQ3
118	ADC_ETC	-	adc_etc IRQ0
119		-	adc_etc IRQ1
120		-	adc_etc IRQ2
121		-	adc_etc Error IRQ
122	PIT	OR	PIT timer 0 interrupt
			PIT timer 1 interrupt
			PIT timer 2 interrupt
			PIT timer 3 interrupt
123	ACMP	-	ACMP 1 IRQ
124		-	ACMP 2 IRQ
125		-	ACMP 3 IRQ
126		-	ACMP 4 IRQ
127	Reserved	-	Reserved
128	Reserved	-	Reserved
129	QDC1	OR	Index marker interrupt
			Home marker interrupt
			watchdog timeout interrupt
			compare interrupt
			simultaneous input switching interrupt
130	QDC2	OR	Index marker interrupt
			Home marker interrupt

Table continues on the next page...

Table 4-2. CM7 domain interrupt summary (continued)

IRQ	Module	Logic	Description
			watchdog timeout interrupt
			compare interrupt
			simultaneous input switching interrupt
131	QDC3	OR	Index marker interrupt
			Home marker interrupt
			watchdog timeout interrupt
			compare interrupt
			simultaneous input switching interrupt
132	QDC4	OR	Index marker interrupt
			Home marker interrupt
			watchdog timeout interrupt
			compare interrupt
			simultaneous input switching interrupt
133	QTIMER1	OR	Interrupt request for timer #0
			Interrupt request for timer #1
			Interrupt request for timer #2
			Interrupt request for timer #3
134	QTIMER2	OR	Interrupt request for timer #0
			Interrupt request for timer #1
			Interrupt request for timer #2
			Interrupt request for timer #3
135	QTIMER3	OR	Interrupt request for timer #0
			Interrupt request for timer #1
			Interrupt request for timer #2
			Interrupt request for timer #3
136	QTIMER4	OR	Interrupt request for timer #0
			Interrupt request for timer #1
			Interrupt request for timer #2
			Interrupt request for timer #3
137	FLEXPWM2	OR	capture PWM0 interrupt
			compare PWM0 interrupt
			reload PWM0 interrupt
138		OR	capture PWM1 interrupt
			compare PWM1 interrupt
			reload PWM1 interrupt
139		OR	capture PWM2 interrupt
			compare PWM2 interrupt
			reload PWM2 interrupt
140		OR	capture PWM3 interrupt

Table continues on the next page...

Table 4-2. CM7 domain interrupt summary (continued)

IRQ	Module	Logic	Description
			compare PWM3 interrupt
			reload PWM3 interrupt
141		OR	fault interrupt
			reload error interrupt
142	FLEXPWM3	OR	capture PWM0 interrupt
			compare PWM0 interrupt
			reload PWM0 interrupt
143		OR	capture PWM1 interrupt
			compare PWM1 interrupt
			reload PWM1 interrupt
144		OR	capture PWM2 interrupt
			compare PWM2 interrupt
			reload PWM2 interrupt
145		OR	capture PWM3 interrupt
			compare PWM3 interrupt
			reload PWM3 interrupt
146		OR	fault interrupt
			reload error interrupt
147	FLEXPWM4	OR	capture PWM0 interrupt
			compare PWM0 interrupt
			reload PWM0 interrupt
148		OR	capture PWM1 interrupt
			compare PWM1 interrupt
			reload PWM1 interrupt
149		OR	capture PWM2 interrupt
			compare PWM2 interrupt
			reload PWM2 interrupt
150		OR	capture PWM3 interrupt
			compare PWM3 interrupt
			reload PWM3 interrupt
151		OR	fault interrupt
			reload error interrupt
152	ENET2	OR	MAC 0 Periodic Timer Overflow
			MAC 0 Time Stamp Available
			MAC 0 Payload Receive Error
			MAC 0 Transmit FIFO Underrun
			MAC 0 Collision Retry Limit
			MAC 0 Late Collision
			MAC 0 Ethernet Bus Error

Table continues on the next page...

Table 4-2. CM7 domain interrupt summary (continued)

IRQ	Module	Logic	Description
			MAC 0 MII Data Transfer Done
			MAC 0 Receive Buffer Done
			MAC 0 Receive Frame Done
			MAC 0 Transmit Buffer Done
			MAC 0 Transmit Frame Done
			MAC 0 Graceful Stop
			MAC 0 Babbling Transmit Error
			MAC 0 Babbling Receive Error
			MAC 0 Wakeup Request (sync)
153	ENET2	-	MAC 0 1588 Timer Interrupt – synchronous
154	FLEXCAN3 (CANFD)	OR	Interrupt from busoff
			Interrupt from CAN line error
			OR'ed interrupts from ipi_int_MB
			RX warning Interrupt
			TX warning Interrupt
			Interrupt from wake up
			Interrupt from match in PN
			Interrupt from timeout in PN
			Correctable error interrupt
			Non correctable error int host
			Non correctable error int internal
			Busoff done interrupt
			FD error interrupt
155	Reserved	-	Reserved
156	FLEXIO3	OR	FlexIO3 interrupt
			FlexIO3 asynchronous interrupt
157	GPIO6/7/8/9	-	OR'ed interrupt of gpio6, gpio7, gpio8 and gpio9
158-159	Reserved	-	Reserved

4.4 DMA Mux

The table below shows the DMA request signals for the peripherals in the chip:

Table 4-3. DMA MUX Mapping

Channel	Module	Logic	Description
0	FLEXIO1	OR	FlexIO DMA Request 0
		OR	FlexIO Async DMA Request 0

Table continues on the next page...

Table 4-3. DMA MUX Mapping (continued)

Channel	Module	Logic	Description
		OR	FlexIO DMA Request 1
		OR	FlexIO Async DMA Request 1
1	FLEXIO2	OR	FlexIO DMA Request 0
		OR	FlexIO Async DMA Request 0
		OR	FlexIO DMA Request 1
		OR	FlexIO Async DMA Request 1
2	LPUART1	OR	UART TX FIFO DMA Request
		OR	UART TX FIFO Async DMA Request
3	LPUART1	OR	UART RX FIFO DMA Request
		OR	UART RX FIFO Async DMA Request
4	LPUART3	OR	UART TX FIFO DMA Request
		OR	UART TX FIFO Async DMA Request
5	LPUART3	OR	UART RX FIFO DMA Request
		OR	UART RX FIFO Async DMA Request
6	LPUART5	OR	UART TX FIFO DMA Request
		OR	UART TX FIFO Async DMA Request
7	LPUART5	OR	UART RX FIFO DMA Request
		OR	UART RX FIFO Async DMA Request
8	LPUART7	OR	UART TX FIFO DMA Request
		OR	UART TX FIFO Async DMA Request
9	LPUART7	OR	UART RX FIFO DMA Request
		OR	UART RX FIFO Async DMA Request
10	Reserved	-	Reserved
11	FLEXCAN3 (CANFD)	-	FLEXCAN3 (CANFD) DMA Request
12	CSI	-	CSI Write DMA Request
13	LPSPI1 SPI2	OR	LPSPI RX FIFO DMA Request
		OR	LPSPI RX FIFO Async DMA Request
14	LPSPI1	OR	LPSPI TX FIFO DMA Request
		OR	LPSPI TX FIFO Async DMA Request
15	LPSPI3 SPI1	OR	LPSPI RX FIFO DMA Request
		OR	LPSPI RX FIFO Async DMA Request
16	LPSPI3	OR	LPSPI TX FIFO DMA Request

Table continues on the next page...

Table 4-3. DMA MUX Mapping (continued)

Channel	Module	Logic	Description
		OR	LPSPI TX FIFO Async DMA Request
17	LPI2C1 Wire	OR	I2C Master RX FIFO DMA Request
		OR	I2C Master RX FIFO Async DMA Request
		OR	I2C Slave RX FIFO DMA Request
		OR	I2C Slave RX FIFO Async DMA Request
		OR	I2C Master TX FIFO DMA Request
		OR	I2C Master TX FIFO Async DMA Request
		OR	I2C Slave TX FIFO DMA Request
		OR	I2C Slave TX FIFO Async DMA Request
18	LPI2C3 Wire1 (Wire2 on MM)	OR	I2C Master RX FIFO DMA Request
		OR	I2C Master RX FIFO Async DMA Request
		OR	I2C Slave RX FIFO DMA Request
		OR	I2C Slave RX FIFO Async DMA Request
		OR	I2C Master TX FIFO DMA Request
		OR	I2C Master TX FIFO Async DMA Request
		OR	I2C Slave TX FIFO DMA Request
		OR	I2C Slave TX FIFO Async DMA Request
19	SAI1	-	SAI RX FIFO DMA Request
20	SAI1	-	SAI TX FIFO DMA Request
21	SAI2	-	SAI RX FIFO DMA Request
22	SAI2	-	SAI TX FIFO DMA Request
23	ADC_ETC	-	ADC_ETC DMA Request
24	ADC1	-	ADC DMA Request
25	ACMP	-	ACMP1 DMA Request
26			ACMP3 DMA Request
27	Reserved	-	Reserved
28	FLEXSPI	-	FlexSPI RX FIFO DMA Request
29	FLEXSPI	-	FlexSPI TX FIFO DMA Request
30	XBAR1	-	XBAR DMA Request 0
31	XBAR1	-	XBAR DMA Request 1
32	FLEXPWM1	-	Read DMA request for capture regs of sub-module PWM0
33	FLEXPWM1	-	Read DMA request for capture regs of sub-module PWM1

Table continues on the next page...

Table 4-3. DMA MUX Mapping (continued)

Channel	Module	Logic	Description
34	FLEXPWM1	-	Read DMA request for capture regs of sub-module PWM2
35	FLEXPWM1	-	Read DMA request for capture regs of sub-module PWM3
36	FLEXPWM1	-	Write DMA request for value regs of sub-module PWM0
37	FLEXPWM1	-	Write DMA request for value regs of sub-module PWM1
38	FLEXPWM1	-	Write DMA request for value regs of sub-module PWM2
39	FLEXPWM1	-	Write DMA request for value regs of sub-module PWM3
40	FLEXPWM3	-	Read DMA request for capture regs of sub-module PWM0
41	FLEXPWM3	-	Read DMA request for capture regs of sub-module PWM1
42	FLEXPWM3	-	Read DMA request for capture regs of sub-module PWM2
43	FLEXPWM3	-	Read DMA request for capture regs of sub-module PWM3
44	FLEXPWM3	-	Write DMA request for value regs of sub-module PWM0
45	FLEXPWM3	-	Write DMA request for value regs of sub-module PWM1
46	FLEXPWM3	-	Write DMA request for value regs of sub-module PWM2
47	FLEXPWM3	-	Write DMA request for value regs of sub-module PWM3
48	QTIMER1	-	DMA read request for capt in timer #0
49	QTIMER1	-	DMA read request for capt in timer #1
50	QTIMER1	-	DMA read request for capt in timer #2
51	QTIMER1	-	DMA read request for capt in timer #3
52	QTIMER1	OR	DMA write request for cmpld1 in timer #0
		OR	DMA write request for cmpld2 in timer #1
53	QTIMER1	OR	DMA write request for cmpld1 in timer #1
		OR	DMA write request for cmpld2 in timer #0
54	QTIMER1	OR	DMA write request for cmpld1 in timer #2

Table continues on the next page...

Table 4-3. DMA MUX Mapping (continued)

Channel	Module	Logic	Description
		OR	DMA write request for cmpld2 in timer #3
55	QTIMER1	OR	DMA write request for cmpld1 in timer #3
		OR	DMA write request for cmpld2 in timer #2
56	QTIMER3	OR	DMA read request for capt in timer #0
		OR	DMA write request for cmpld1 in timer #0
		OR	DMA write request for cmpld2 in timer #1
57	QTIMER3	OR	DMA read request for capt in timer #1
		OR	DMA write request for cmpld1 in timer #1
		OR	DMA write request for cmpld2 in timer #0
58	QTIMER3	OR	DMA read request for capt in timer #2
		OR	DMA write request for cmpld1 in timer #2
		OR	DMA write request for cmpld2 in timer #3
59	QTIMER3	OR	DMA read request for capt in timer #3
		OR	DMA write request for cmpld1 in timer #3
		OR	DMA write request for cmpld2 in timer #2
60	FlexSPI2	-	FlexSPI2 RX FIFO DMA Request
61	FlexSPI2	-	FlexSPI2 TX FIFO DMA Request
62	Reserved	-	Reserved
63	Reserved	-	Reserved
64	FLEXIO1	OR	FlexIO DMA Request 2
		OR	FlexIO Async DMA Request 2
		OR	FlexIO DMA Request 3
		OR	FlexIO Async DMA Request 3
65	FLEXIO2	OR	FlexIO DMA Request 2
		OR	FlexIO Async DMA Request 2
		OR	FlexIO DMA Request 3
		OR	FlexIO Async DMA Request 3
66	LPUART2	OR	UART TX FIFO DMA Request

Table continues on the next page...

Table 4-3. DMA MUX Mapping (continued)

Channel	Module	Logic	Description
		OR	UART TX FIFO Async DMA Request
67	LPUART2	OR	UART RX FIFO DMA Request
		OR	UART RX FIFO Async DMA Request
68	LPUART4	OR	UART TX FIFO DMA Request
		OR	UART TX FIFO Async DMA Request
69	LPUART4	OR	UART RX FIFO DMA Request
		OR	UART RX FIFO Async DMA Request
70	LPUART6	OR	UART TX FIFO DMA Request
		OR	UART TX FIFO Async DMA Request
71	LPUART6	OR	UART RX FIFO DMA Request
		OR	UART RX FIFO Async DMA Request
72	LPUART8	OR	UART TX FIFO DMA Request
		OR	UART TX FIFO Async DMA Request
73	LPUART8	OR	UART RX FIFO DMA Request
		OR	UART RX FIFO Async DMA Request
74	Reserved	-	Reserved
75	PXP	-	PXP DMA Event
76	LCDIF	-	LCDIF DMA Event
77	LPSPI2	OR	LPSPI RX FIFO DMA Request
		OR	LPSPI RX FIFO Async DMA Request
78	LPSPI2	OR	LPSPI TX FIFO DMA Request
		OR	LPSPI TX FIFO Async DMA Request
79	LPSPI4	OR	LPSPI RX FIFO DMA Request
		OR	LPSPI RX FIFO Async DMA Request
80	LPSPI4	OR	LPSPI TX FIFO DMA Request
		OR	LPSPI TX FIFO Async DMA Request
81	LPI2C2	OR	I2C Master RX FIFO DMA Request
	Wire3 on MM	OR	I2C Master RX FIFO Async DMA Request
		OR	I2C Slave RX FIFO DMA Request

Table continues on the next page...

Table 4-3. DMA MUX Mapping (continued)

Channel	Module	Logic	Description
		OR	I2C Slave RX FIFO Async DMA Request
		OR	I2C Master TX FIFO DMA Request
		OR	I2C Master TX FIFO Async DMA Request
		OR	I2C Master TX FIFO DMA Request
		OR	I2C Master TX FIFO Async DMA Request
82	LPI2C4 Wire2 (Wire1 on MM)	OR	I2C Master RX FIFO DMA Request
		OR	I2C Master RX FIFO Async DMA Request
		OR	I2C Slave RX FIFO DMA Request
		OR	I2C Slave RX FIFO Async DMA Request
		OR	I2C Master TX FIFO DMA Request
		OR	I2C Master TX FIFO Async DMA Request
		OR	I2C Master TX FIFO DMA Request
		OR	I2C Master TX FIFO Async DMA Request
83	SAI3	-	SAI RX FIFO DMA Request
84	SAI3	-	SAI TX FIFO DMA Request
85	SPDIF	-	SPDIF RX DMA Request
86	SPDIF	-	SPDIF TX DMA Request
87	Reserved	-	Reserved
88	ADC2	-	ADC DMA Request
89	ACMP	-	ACMP2 DMA Request
90			ACMP4 DMA Request
91	Reserved	-	Reserved
92	ENET	-	ENET Timer DMA Request 0
93	ENET	-	ENET Timer DMA Request 1
94	XBAR1	-	XBAR DMA Request 2
95	XBAR1	-	XBAR DMA Request 3
96	FLEXPWM2	-	Read DMA request for capture regs of sub-module PWM0
97	FLEXPWM2	-	Read DMA request for capture regs of sub-module PWM1
98	FLEXPWM2	-	Read DMA request for capture regs of sub-module PWM2
99	FLEXPWM2	-	Read DMA request for capture regs of sub-module PWM3
100	FLEXPWM2	-	Write DMA request for value regs of sub-module PWM0

Table continues on the next page...

Table 4-3. DMA MUX Mapping (continued)

Channel	Module	Logic	Description
101	FLEXPWM2	-	Write DMA request for value regs of sub-module PWM1
102	FLEXPWM2	-	Write DMA request for value regs of sub-module PWM2
103	FLEXPWM2	-	Write DMA request for value regs of sub-module PWM3
104	FLEXPWM4	-	Read DMA request for capture regs of sub-module PWM0
105	FLEXPWM4	-	Read DMA request for capture regs of sub-module PWM1
106	FLEXPWM4	-	Read DMA request for capture regs of sub-module PWM2
107	FLEXPWM4	-	Read DMA request for capture regs of sub-module PWM3
108	FLEXPWM4	-	Write DMA request for value regs of sub-module PWM0
109	FLEXPWM4	-	Write DMA request for value regs of sub-module PWM1
110	FLEXPWM4	-	Write DMA request for value regs of sub-module PWM2
111	FLEXPWM4	-	Write DMA request for value regs of sub-module PWM3
112	QTIMER2	-	DMA read request for capt in timer #0
113	QTIMER2	-	DMA read request for capt in timer #1
114	QTIMER2	-	DMA read request for capt in timer #2
115	QTIMER2	-	DMA read request for capt in timer #3
116	QTIMER2	OR	DMA write request for cmpld1 in timer #0
		OR	DMA write request for cmpld2 in timer #1
117	QTIMER2	OR	DMA write request for cmpld1 in timer #1
		OR	DMA write request for cmpld2 in timer #0
118	QTIMER2	OR	DMA write request for cmpld1 in timer #2
		OR	DMA write request for cmpld2 in timer #3
119	QTIMER2	OR	DMA write request for cmpld1 in timer #3
		OR	DMA write request for cmpld2 in timer #2

Table continues on the next page...

Table 4-3. DMA MUX Mapping (continued)

Channel	Module	Logic	Description
120	QTIMER4	OR	DMA read request for capt in timer #0
		OR	DMA write request for cmpld1 in timer #0
		OR	DMA write request for cmpld2 in timer #1
121	QTIMER4	OR	DMA read request for capt in timer #1
		OR	DMA write request for cmpld1 in timer #1
		OR	DMA write request for cmpld2 in timer #0
122	QTIMER4	OR	DMA read request for capt in timer #2
		OR	DMA write request for cmpld1 in timer #2
		OR	DMA write request for cmpld2 in timer #3
123	QTIMER4	OR	DMA read request for capt in timer #3
		OR	DMA write request for cmpld1 in timer #3
		OR	DMA write request for cmpld2 in timer #2
124	ENET2	-	ENET2 Timer DMA Request 0
125	ENET2	-	ENET2 Timer DMA Request 1
126-127	Reserved	-	Reserved

4.5 PIT Channel Assignments For Periodic DMA Triggering

This table shows the PIT channel to DMA channel correspondence.

Table 4-4. PIT Channel Assignments For Periodic DMA Triggering

DMA Channel Number	PIT Channel
eDMA Channel 0	PIT Channel 0
eDMA Channel 1	PIT Channel 1
eDMA Channel 2	PIT Channel 2
eDMA Channel 3	PIT Channel 3

4.6 XBAR Resource Assignments

This table shows the XBAR resource assignments in the chip.

NOTE

ADC_ETC0_COCO0 ... ADC_ETC1_COCO3 in the table are corresponding to coco0 ... coco7, also ADC_ETC_TRIG00 ... ADC_ETC_TRIG13 corresponding to trg0 ... trg7, see the figure "ADC_ETC block diagram" in the ADC_ETC chapter.

Table 4-5. XBAR1 Input Assignments

Assigned Input	XBAR1 Input	Gate
LOGIC LOW	XBAR1_IN00	-
LOGIC HIGH	XBAR1_IN01	-
IOMUX_XBAR_IN02	XBAR1_IN02	-
IOMUX_XBAR_IN03	XBAR1_IN03	-
IOMUX_XBAR_INOUT04	XBAR1_IN04	-
IOMUX_XBAR_INOUT05	XBAR1_IN05	-
IOMUX_XBAR_INOUT06	XBAR1_IN06	-
IOMUX_XBAR_INOUT07	XBAR1_IN07	-
IOMUX_XBAR_INOUT08	XBAR1_IN08	-
IOMUX_XBAR_INOUT09	XBAR1_IN09	-
IOMUX_XBAR_INOUT10	XBAR1_IN10	-
IOMUX_XBAR_INOUT11	XBAR1_IN11	-
IOMUX_XBAR_INOUT12	XBAR1_IN12	-
IOMUX_XBAR_INOUT13	XBAR1_IN13	-
IOMUX_XBAR_INOUT14	XBAR1_IN14	-
IOMUX_XBAR_INOUT15	XBAR1_IN15	-
IOMUX_XBAR_INOUT16	XBAR1_IN16	-
IOMUX_XBAR_INOUT17	XBAR1_IN17	-
IOMUX_XBAR_INOUT18	XBAR1_IN18	-
IOMUX_XBAR_INOUT19	XBAR1_IN19	-
IOMUX_XBAR_IN20	XBAR1_IN20	-
IOMUX_XBAR_IN21	XBAR1_IN21	-
IOMUX_XBAR_IN22	XBAR1_IN22	-
IOMUX_XBAR_IN23	XBAR1_IN23	-
IOMUX_XBAR_IN24	XBAR1_IN24	-
IOMUX_XBAR_IN25	XBAR1_IN25	-
ACMP1_OUT	XBAR1_IN26	-
ACMP2_OUT	XBAR1_IN27	-
ACMP3_OUT	XBAR1_IN28	-

Table continues on the next page...

Table 4-5. XBAR1 Input Assignments (continued)

Assigned Input	XBAR1 Input	Gate
ACMP4_OUT	XBAR1_IN29	-
Reserved	XBAR1_IN30	-
Reserved	XBAR1_IN31	-
QTIMER3_TIMER0	XBAR1_IN32	-
QTIMER3_TIMER1	XBAR1_IN33	-
QTIMER3_TIMER2	XBAR1_IN34	-
QTIMER3_TIMER3	XBAR1_IN35	-
QTIMER4_TIMER0	XBAR1_IN36	-
QTIMER4_TIMER1	XBAR1_IN37	-
QTIMER4_TIMER2	XBAR1_IN38	-
QTIMER4_TIMER3	XBAR1_IN39	-
FLEXPWM1_PWM1_OUT_TRIG0	XBAR1_IN40	OR
FLEXPWM1_PWM1_OUT_TRIG1		OR
FLEXPWM1_PWM2_OUT_TRIG0	XBAR1_IN41	OR
FLEXPWM1_PWM2_OUT_TRIG1		OR
FLEXPWM1_PWM3_OUT_TRIG0	XBAR1_IN42	OR
FLEXPWM1_PWM3_OUT_TRIG1		OR
FLEXPWM1_PWM4_OUT_TRIG0	XBAR1_IN43	OR
FLEXPWM1_PWM4_OUT_TRIG1		OR
FLEXPWM2_PWM1_OUT_TRIG0	XBAR1_IN44	OR
FLEXPWM2_PWM1_OUT_TRIG1		OR
FLEXPWM2_PWM2_OUT_TRIG0	XBAR1_IN45	OR
FLEXPWM2_PWM2_OUT_TRIG1		OR
FLEXPWM2_PWM3_OUT_TRIG0	XBAR1_IN46	OR
FLEXPWM2_PWM3_OUT_TRIG1		OR
FLEXPWM2_PWM4_OUT_TRIG0	XBAR1_IN47	OR
FLEXPWM2_PWM4_OUT_TRIG1		OR
FLEXPWM3_PWM1_OUT_TRIG0	XBAR1_IN48	OR
FLEXPWM3_PWM1_OUT_TRIG1		OR
FLEXPWM3_PWM2_OUT_TRIG0	XBAR1_IN49	OR
FLEXPWM3_PWM2_OUT_TRIG1		OR
FLEXPWM3_PWM3_OUT_TRIG0	XBAR1_IN50	OR
FLEXPWM3_PWM3_OUT_TRIG1		OR
FLEXPWM3_PWM4_OUT_TRIG0	XBAR1_IN51	OR
FLEXPWM3_PWM4_OUT_TRIG1		OR
FLEXPWM4_PWM1_OUT_TRIG0	XBAR1_IN52	OR
FLEXPWM4_PWM1_OUT_TRIG1		OR
FLEXPWM4_PWM2_OUT_TRIG0	XBAR1_IN53	OR
FLEXPWM4_PWM2_OUT_TRIG1		OR

Table continues on the next page...

Table 4-5. XBAR1 Input Assignments (continued)

Assigned Input	XBAR1 Input	Gate
FLEXPWM4_PWM3_OUT_TRIG0	XBAR1_IN54	OR
FLEXPWM4_PWM3_OUT_TRIG1		OR
FLEXPWM4_PWM4_OUT_TRIG0	XBAR1_IN55	OR
FLEXPWM4_PWM4_OUT_TRIG1		OR
PIT_TRIGGER0	XBAR1_IN56	-
PIT_TRIGGER1	XBAR1_IN57	-
PIT_TRIGGER2	XBAR1_IN58	-
PIT_TRIGGER3	XBAR1_IN59	-
QDC1_POS_MATCH	XBAR1_IN60	-
QDC2_POS_MATCH	XBAR1_IN61	-
QDC3_POS_MATCH	XBAR1_IN62	-
QDC4_POS_MATCH	XBAR1_IN63	-
DMA_DONE0	XBAR1_IN64	-
DMA_DONE1	XBAR1_IN65	-
DMA_DONE2	XBAR1_IN66	-
DMA_DONE3	XBAR1_IN67	-
DMA_DONE4	XBAR1_IN68	-
DMA_DONE5	XBAR1_IN69	-
DMA_DONE6	XBAR1_IN70	-
DMA_DONE7	XBAR1_IN71	-
AOI1_OUT0	XBAR1_IN72	-
AOI1_OUT1	XBAR1_IN73	-
AOI1_OUT2	XBAR1_IN74	-
AOI1_OUT3	XBAR1_IN75	-
AOI2_OUT0	XBAR1_IN76	-
AOI2_OUT1	XBAR1_IN77	-
AOI2_OUT2	XBAR1_IN78	-
AOI2_OUT3	XBAR1_IN79	-
ADC_ETC0_COCO0	XBAR1_IN80	-
ADC_ETC0_COCO1	XBAR1_IN81	-
ADC_ETC0_COCO2	XBAR1_IN82	-
ADC_ETC0_COCO3	XBAR1_IN83	-
ADC_ETC1_COCO0	XBAR1_IN84	-
ADC_ETC1_COCO1	XBAR1_IN85	-
ADC_ETC1_COCO2	XBAR1_IN86	-
ADC_ETC1_COCO3	XBAR1_IN87	-

Table 4-6. XBAR2 Input Assignments

Assigned Input	XBAR2 Input	Gate
LOGIC LOW	XBAR2_IN00	-
LOGIC HIGH	XBAR2_IN01	-
Reserved	XBAR2_IN02	-
Reserved	XBAR2_IN03	-
Reserved	XBAR2_IN04	-
Reserved	XBAR2_IN05	-
ACMP1_OUT	XBAR2_IN06	-
ACMP2_OUT	XBAR2_IN07	-
ACMP3_OUT	XBAR2_IN08	-
ACMP4_OUT	XBAR2_IN09	-
Reserved	XBAR2_IN10	-
Reserved	XBAR2_IN11	-
QTIMER3_TIMER0	XBAR2_IN12	-
QTIMER3_TIMER1	XBAR2_IN13	-
QTIMER3_TIMER2	XBAR2_IN14	-
QTIMER3_TIMER3	XBAR2_IN15	-
QTIMER4_TIMER0	XBAR2_IN16	-
QTIMER4_TIMER1	XBAR2_IN17	-
QTIMER4_TIMER2	XBAR2_IN18	-
QTIMER4_TIMER3	XBAR2_IN19	-
FLEXPWM1_PWM1_OUT_TRIG0	XBAR2_IN20	OR
FLEXPWM1_PWM1_OUT_TRIG1	XBAR2_IN20	OR
FLEXPWM1_PWM2_OUT_TRIG0	XBAR2_IN21	OR
FLEXPWM1_PWM2_OUT_TRIG1	XBAR2_IN21	OR
FLEXPWM1_PWM3_OUT_TRIG0	XBAR2_IN22	OR
FLEXPWM1_PWM3_OUT_TRIG1	XBAR2_IN22	OR
FLEXPWM1_PWM4_OUT_TRIG0	XBAR2_IN23	OR
FLEXPWM1_PWM4_OUT_TRIG1	XBAR2_IN23	OR
FLEXPWM2_PWM1_OUT_TRIG0	XBAR2_IN24	OR
FLEXPWM2_PWM1_OUT_TRIG1	XBAR2_IN24	OR
FLEXPWM2_PWM2_OUT_TRIG0	XBAR2_IN25	OR
FLEXPWM2_PWM2_OUT_TRIG1	XBAR2_IN25	OR
FLEXPWM2_PWM3_OUT_TRIG0	XBAR2_IN26	OR
FLEXPWM2_PWM3_OUT_TRIG1	XBAR2_IN26	OR
FLEXPWM2_PWM4_OUT_TRIG0	XBAR2_IN27	OR
FLEXPWM2_PWM4_OUT_TRIG1	XBAR2_IN27	OR
FLEXPWM3_PWM1_OUT_TRIG0	XBAR2_IN28	OR
FLEXPWM3_PWM1_OUT_TRIG1	XBAR2_IN28	OR
FLEXPWM3_PWM2_OUT_TRIG0	XBAR2_IN29	OR

Table continues on the next page...

Table 4-6. XBAR2 Input Assignments (continued)

Assigned Input	XBAR2 Input	Gate
FLEXPWM3_PWM2_OUT_TRIG1	XBAR2_IN29	OR
FLEXPWM3_PWM3_OUT_TRIG0	XBAR2_IN30	OR
FLEXPWM3_PWM3_OUT_TRIG1	XBAR2_IN30	OR
FLEXPWM3_PWM4_OUT_TRIG0	XBAR2_IN31	OR
FLEXPWM3_PWM4_OUT_TRIG1	XBAR2_IN31	OR
FLEXPWM4_PWM1_OUT_TRIG0	XBAR2_IN32	OR
FLEXPWM4_PWM1_OUT_TRIG1	XBAR2_IN32	OR
FLEXPWM4_PWM2_OUT_TRIG0	XBAR2_IN33	OR
FLEXPWM4_PWM2_OUT_TRIG1	XBAR2_IN33	OR
FLEXPWM4_PWM3_OUT_TRIG0	XBAR2_IN34	OR
FLEXPWM4_PWM3_OUT_TRIG1	XBAR2_IN34	OR
FLEXPWM4_PWM4_OUT_TRIG0	XBAR2_IN35	OR
FLEXPWM4_PWM4_OUT_TRIG1	XBAR2_IN35	OR
PIT_TRIGGER0	XBAR2_IN36	-
PIT_TRIGGER1	XBAR2_IN37	-
ADC_ETC0_COCO0	XBAR2_IN38	-
ADC_ETC0_COCO1	XBAR2_IN39	-
ADC_ETC0_COCO2	XBAR2_IN40	-
ADC_ETC0_COCO3	XBAR2_IN41	-
ADC_ETC1_COCO0	XBAR2_IN42	-
ADC_ETC1_COCO1	XBAR2_IN43	-
ADC_ETC1_COCO2	XBAR2_IN44	-
ADC_ETC1_COCO3	XBAR2_IN45	-
QDC1_POS_MATCH	XBAR2_IN46	-
QDC2_POS_MATCH	XBAR2_IN47	-
QDC3_POS_MATCH	XBAR2_IN48	-
QDC4_POS_MATCH	XBAR2_IN49	-
DMA_DONE0	XBAR2_IN50	-
DMA_DONE1	XBAR2_IN51	-
DMA_DONE2	XBAR2_IN52	-
DMA_DONE3	XBAR2_IN53	-
DMA_DONE4	XBAR2_IN54	-
DMA_DONE5	XBAR2_IN55	-
DMA_DONE6	XBAR2_IN56	-
DMA_DONE7	XBAR2_IN57	-

Table 4-7. XBAR3 Input Assignments

Assigned Input	XBAR3 Input	Gate
LOGIC LOW	XBAR3_IN00	-
LOGIC HIGH	XBAR3_IN01	-
Reserved	XBAR3_IN02	-
Reserved	XBAR3_IN03	-
Reserved	XBAR3_IN04	-
Reserved	XBAR3_IN05	-
ACMP1_OUT	XBAR3_IN06	-
ACMP2_OUT	XBAR3_IN07	-
ACMP3_OUT	XBAR3_IN08	-
ACMP4_OUT	XBAR3_IN09	-
Reserved	XBAR3_IN10	-
Reserved	XBAR3_IN11	-
QTIMER3_TIMER0	XBAR3_IN12	-
QTIMER3_TIMER1	XBAR3_IN13	-
QTIMER3_TIMER2	XBAR3_IN14	-
QTIMER3_TIMER3	XBAR3_IN15	-
QTIMER4_TIMER0	XBAR3_IN16	-
QTIMER4_TIMER1	XBAR3_IN17	-
QTIMER4_TIMER2	XBAR3_IN18	-
QTIMER4_TIMER3	XBAR3_IN19	-
FLEXPWM1_PWM1_OUT_TRIG0	XBAR3_IN20	OR
FLEXPWM1_PWM1_OUT_TRIG1	XBAR3_IN20	OR
FLEXPWM1_PWM2_OUT_TRIG0	XBAR3_IN21	OR
FLEXPWM1_PWM2_OUT_TRIG1	XBAR3_IN21	OR
FLEXPWM1_PWM3_OUT_TRIG0	XBAR3_IN22	OR
FLEXPWM1_PWM3_OUT_TRIG1	XBAR3_IN22	OR
FLEXPWM1_PWM4_OUT_TRIG0	XBAR3_IN23	OR
FLEXPWM1_PWM4_OUT_TRIG1	XBAR3_IN23	OR
FLEXPWM2_PWM1_OUT_TRIG0	XBAR3_IN24	OR
FLEXPWM2_PWM1_OUT_TRIG1	XBAR3_IN24	OR
FLEXPWM2_PWM2_OUT_TRIG0	XBAR3_IN25	OR
FLEXPWM2_PWM2_OUT_TRIG1	XBAR3_IN25	OR
FLEXPWM2_PWM3_OUT_TRIG0	XBAR3_IN26	OR
FLEXPWM2_PWM3_OUT_TRIG1	XBAR3_IN26	OR
FLEXPWM2_PWM4_OUT_TRIG0	XBAR3_IN27	OR
FLEXPWM2_PWM4_OUT_TRIG1	XBAR3_IN27	OR
FLEXPWM3_PWM1_OUT_TRIG0	XBAR3_IN28	OR
FLEXPWM3_PWM1_OUT_TRIG1	XBAR3_IN28	OR
FLEXPWM3_PWM2_OUT_TRIG0	XBAR3_IN29	OR

Table continues on the next page...

Table 4-7. XBAR3 Input Assignments (continued)

Assigned Input	XBAR3 Input	Gate
FLEXPWM3_PWM2_OUT_TRIG1	XBAR3_IN29	OR
FLEXPWM3_PWM3_OUT_TRIG0	XBAR3_IN30	OR
FLEXPWM3_PWM3_OUT_TRIG1	XBAR3_IN30	OR
FLEXPWM3_PWM4_OUT_TRIG0	XBAR3_IN31	OR
FLEXPWM3_PWM4_OUT_TRIG1	XBAR3_IN31	OR
FLEXPWM4_PWM1_OUT_TRIG0	XBAR3_IN32	OR
FLEXPWM4_PWM1_OUT_TRIG1	XBAR3_IN32	OR
FLEXPWM4_PWM2_OUT_TRIG0	XBAR3_IN33	OR
FLEXPWM4_PWM2_OUT_TRIG1	XBAR3_IN33	OR
FLEXPWM4_PWM3_OUT_TRIG0	XBAR3_IN34	OR
FLEXPWM4_PWM3_OUT_TRIG1	XBAR3_IN34	OR
FLEXPWM4_PWM4_OUT_TRIG0	XBAR3_IN35	OR
FLEXPWM4_PWM4_OUT_TRIG1	XBAR3_IN35	OR
PIT_TRIGGER0	XBAR3_IN36	-
PIT_TRIGGER1	XBAR3_IN37	-
ADC_ETC0_COCO0	XBAR3_IN38	-
ADC_ETC0_COCO1	XBAR3_IN39	-
ADC_ETC0_COCO2	XBAR3_IN40	-
ADC_ETC0_COCO3	XBAR3_IN41	-
ADC_ETC1_COCO0	XBAR3_IN42	-
ADC_ETC1_COCO1	XBAR3_IN43	-
ADC_ETC1_COCO2	XBAR3_IN44	-
ADC_ETC1_COCO3	XBAR3_IN45	-
QDC1_POS_MATCH	XBAR3_IN46	-
QDC2_POS_MATCH	XBAR3_IN47	-
QDC3_POS_MATCH	XBAR3_IN48	-
QDC4_POS_MATCH	XBAR3_IN49	-
DMA_DONE0	XBAR3_IN50	-
DMA_DONE1	XBAR3_IN51	-
DMA_DONE2	XBAR3_IN52	-
DMA_DONE3	XBAR3_IN53	-
DMA_DONE4	XBAR3_IN54	-
DMA_DONE5	XBAR3_IN55	-
DMA_DONE6	XBAR3_IN56	-
DMA_DONE7	XBAR3_IN57	-

Table 4-8. XBAR1 Output Assignments

XBAR1 Output	Assigned Output	Gate
XBAR1_OUT00	DMA_CH_MUX_REQ30	-
XBAR1_OUT01	DMA_CH_MUX_REQ31	-
XBAR1_OUT02	DMA_CH_MUX_REQ94	-
XBAR1_OUT03	DMA_CH_MUX_REQ95	-
XBAR1_OUT04	IOMUX_XBAR_INOUT04	-
XBAR1_OUT05	IOMUX_XBAR_INOUT05	-
XBAR1_OUT06	IOMUX_XBAR_INOUT06	-
XBAR1_OUT07	IOMUX_XBAR_INOUT07	-
XBAR1_OUT08	IOMUX_XBAR_INOUT08	-
XBAR1_OUT09	IOMUX_XBAR_INOUT09	-
XBAR1_OUT10	IOMUX_XBAR_INOUT10	-
XBAR1_OUT11	IOMUX_XBAR_INOUT11	-
XBAR1_OUT12	IOMUX_XBAR_INOUT12	-
XBAR1_OUT13	IOMUX_XBAR_INOUT13	-
XBAR1_OUT14	IOMUX_XBAR_INOUT14	-
XBAR1_OUT15	IOMUX_XBAR_INOUT15	-
XBAR1_OUT16	IOMUX_XBAR_INOUT16	-
XBAR1_OUT17	IOMUX_XBAR_INOUT17	-
XBAR1_OUT18	IOMUX_XBAR_INOUT18	-
XBAR1_OUT19	IOMUX_XBAR_INOUT19	-
XBAR1_OUT20	ACMP1_SAMPLE	-
XBAR1_OUT21	ACMP2_SAMPLE	-
XBAR1_OUT22	ACMP3_SAMPLE	-
XBAR1_OUT23	ACMP4_SAMPLE	-
XBAR1_OUT24	Reserved	-
XBAR1_OUT25	Reserved	-
XBAR1_OUT26	FLEXPWM1_PWM0_EXTA	-
XBAR1_OUT27	FLEXPWM1_PWM1_EXTA	-
XBAR1_OUT28	FLEXPWM1_PWM2_EXTA	-
XBAR1_OUT29	FLEXPWM1_PWM3_EXTA	-
XBAR1_OUT30	FLEXPWM1_PWM0_EXT_SYNC	-
XBAR1_OUT31	FLEXPWM1_PWM1_EXT_SYNC	-
XBAR1_OUT32	FLEXPWM1_PWM2_EXT_SYNC	-
XBAR1_OUT33	FLEXPWM1_PWM3_EXT_SYNC	-
XBAR1_OUT34	FLEXPWM1_EXT_CLK	-
XBAR1_OUT35	FLEXPWM1_FAULT0	-
XBAR1_OUT36	FLEXPWM1_FAULT1	-
XBAR1_OUT37	FLEXPWM1_FAULT2	-
	FLEXPWM2_FAULT2	-

Table continues on the next page...

Table 4-8. XBAR1 Output Assignments (continued)

XBAR1 Output	Assigned Output	Gate
	FLEXPWM3_FAULT2	-
	FLEXPWM4_FAULT2	-
XBAR1_OUT38	FLEXPWM1_FAULT3	-
	FLEXPWM2_FAULT3	-
	FLEXPWM3_FAULT3	-
	FLEXPWM4_FAULT3	-
XBAR1_OUT39	FLEXPWM1_EXT_FORCE	-
XBAR1_OUT40	FLEXPWM2_PWM0_EXTA	-
	FLEXPWM3_PWM0_EXTA	-
	FLEXPWM4_PWM0_EXTA	-
XBAR1_OUT41	FLEXPWM2_PWM1_EXTA	-
	FLEXPWM3_PWM1_EXTA	-
	FLEXPWM4_PWM1_EXTA	-
XBAR1_OUT42	FLEXPWM2_PWM2_EXTA	-
	FLEXPWM3_PWM2_EXTA	-
	FLEXPWM4_PWM2_EXTA	-
XBAR1_OUT43	FLEXPWM2_PWM3_EXTA	-
	FLEXPWM3_PWM3_EXTA	-
	FLEXPWM4_PWM3_EXTA	-
XBAR1_OUT44	FLEXPWM2_PWM0_EXT_SYNC	-
XBAR1_OUT45	FLEXPWM2_PWM1_EXT_SYNC	-
XBAR1_OUT46	FLEXPWM2_PWM2_EXT_SYNC	-
XBAR1_OUT47	FLEXPWM2_PWM3_EXT_SYNC	-
XBAR1_OUT48	FLEXPWM2_EXT_CLK	-
	FLEXPWM3_EXT_CLK	-
	FLEXPWM4_EXT_CLK	-
XBAR1_OUT49	FLEXPWM2_FAULT0	-
XBAR1_OUT50	FLEXPWM2_FAULT1	-
XBAR1_OUT51	FLEXPWM2_EXT_FORCE	-
XBAR1_OUT52	FLEXPWM3_EXT_SYNC0	-
XBAR1_OUT53	FLEXPWM3_EXT_SYNC1	-
XBAR1_OUT54	FLEXPWM3_EXT_SYNC2	-
XBAR1_OUT55	FLEXPWM3_EXT_SYNC3	-
XBAR1_OUT56	FLEXPWM3_FAULT0	-
XBAR1_OUT57	FLEXPWM3_FAULT1	-
XBAR1_OUT58	FLEXPWM3_EXT_FORCE	-
XBAR1_OUT59	FLEXPWM4_EXT_SYNC0	-
XBAR1_OUT60	FLEXPWM4_EXT_SYNC1	-
XBAR1_OUT61	FLEXPWM4_EXT_SYNC2	-

Table continues on the next page...

Table 4-8. XBAR1 Output Assignments (continued)

XBAR1 Output	Assigned Output	Gate
XBAR1_OUT62	FLEXPWM4_EXT_SYNC3	-
XBAR1_OUT63	FLEXPWM4_FAULT0	-
XBAR1_OUT64	FLEXPWM4_FAULT1	-
XBAR1_OUT65	FLEXPWM4_EXT_FORCE	-
XBAR1_OUT66	QDC1_PHASEA_INPUT	-
XBAR1_OUT67	QDC1_PHASEB_INPUT	-
XBAR1_OUT68	QDC1_INDEX	-
XBAR1_OUT69	QDC1_HOME	-
XBAR1_OUT70	QDC1_TRIGGER	-
XBAR1_OUT71	QDC2_PHASEA_INPUT	-
XBAR1_OUT72	QDC2_PHASEB_INPUT	-
XBAR1_OUT73	QDC2_INDEX	-
XBAR1_OUT74	QDC2_HOME	-
XBAR1_OUT75	QDC2_TRIGGER	-
XBAR1_OUT76	QDC3_PHASEA_INPUT	-
XBAR1_OUT77	QDC3_PHASEB_INPUT	-
XBAR1_OUT78	QDC3_INDEX	-
XBAR1_OUT79	QDC3_HOME	-
XBAR1_OUT80	QDC3_TRIGGER	-
XBAR1_OUT81	QDC4_PHASEA_INPUT	-
XBAR1_OUT82	QDC4_PHASEB_INPUT	-
XBAR1_OUT83	QDC4_INDEX	-
XBAR1_OUT84	QDC4_HOME	-
XBAR1_OUT85	QDC4_TRIGGER	-
XBAR1_OUT86	QTIMER1_TIMER0	Selectable by setting IOMUXC_GPR_GPR6[QTIMER1_TRM0_INPUT_SEL]
XBAR1_OUT87	QTIMER1_TIMER1	Selectable by setting IOMUXC_GPR_GPR6[QTIMER1_TRM1_INPUT_SEL]
XBAR1_OUT88	QTIMER1_TIMER2	Selectable by setting IOMUXC_GPR_GPR6[QTIMER1_TRM2_INPUT_SEL]
XBAR1_OUT89	QTIMER1_TIMER3	Selectable by setting IOMUXC_GPR_GPR6[QTIMER1_TRM3_INPUT_SEL]
XBAR1_OUT90	QTIMER2_TIMER0	Selectable by setting IOMUXC_GPR_GPR6[QTIMER2_TRM0_INPUT_SEL]
XBAR1_OUT91	QTIMER2_TIMER1	Selectable by setting

Table continues on the next page...

Table 4-8. XBAR1 Output Assignments (continued)

XBAR1 Output	Assigned Output	Gate
		IOMUXC_GPR_GPR6[QTIMER2_TRM1_INPUT_SEL]
XBAR1_OUT92	QTIMER2_TIMER2	Selectable by setting IOMUXC_GPR_GPR6[QTIMER2_TRM2_INPUT_SEL]
XBAR1_OUT93	QTIMER2_TIMER3	Selectable by setting IOMUXC_GPR_GPR6[QTIMER2_TRM3_INPUT_SEL]
XBAR1_OUT94	QTIMER3_TIMER0	Selectable by setting IOMUXC_GPR_GPR6[QTIMER3_TRM0_INPUT_SEL]
XBAR1_OUT95	QTIMER3_TIMER1	Selectable by setting IOMUXC_GPR_GPR6[QTIMER3_TRM1_INPUT_SEL]
XBAR1_OUT96	QTIMER3_TIMER2	Selectable by setting IOMUXC_GPR_GPR6[QTIMER3_TRM2_INPUT_SEL]
XBAR1_OUT97	QTIMER3_TIMER3	Selectable by setting IOMUXC_GPR_GPR6[QTIMER3_TRM3_INPUT_SEL]
XBAR1_OUT98	QTIMER4_TIMER0	Selectable by setting IOMUXC_GPR_GPR6[QTIMER4_TRM0_INPUT_SEL]
XBAR1_OUT99	QTIMER4_TIMER1	Selectable by setting IOMUXC_GPR_GPR6[QTIMER4_TRM1_INPUT_SEL]
XBAR1_OUT100	QTIMER4_TIMER2	Selectable by setting IOMUXC_GPR_GPR6[QTIMER4_TRM2_INPUT_SEL]
XBAR1_OUT101	QTIMER4_TIMER3	Selectable by setting IOMUXC_GPR_GPR6[QTIMER4_TRM3_INPUT_SEL]
XBAR1_OUT102	EWM_EWM_IN	-
XBAR1_OUT103	ADC_ETC_TRIG00	-
XBAR1_OUT104	ADC_ETC_TRIG01	-
XBAR1_OUT105	ADC_ETC_TRIG02	-
XBAR1_OUT106	ADC_ETC_TRIG03	-
XBAR1_OUT107	ADC_ETC_TRIG10	-
XBAR1_OUT108	ADC_ETC_TRIG11	-
XBAR1_OUT109	ADC_ETC_TRIG12	-
XBAR1_OUT110	ADC_ETC_TRIG13	-
XBAR1_OUT111 Wire	LPI2C1_TRG_INPUT	-

Table continues on the next page...

Table 4-8. XBAR1 Output Assignments (continued)

XBAR1 Output	Assigned Output	Gate
XBAR1_OUT112 Wire3 on MM	LPI2C2_TRG_INPUT	-
XBAR1_OUT113 Wire1 (Wire2 MM)	LPI2C3_TRG_INPUT	-
XBAR1_OUT114 Wire2 (Wire1 MM)	LPI2C4_TRG_INPUT	-
XBAR1_OUT115 SPI2	LPSPI1_TRG_INPUT	-
XBAR1_OUT116	LPSPI2_TRG_INPUT	-
XBAR1_OUT117 SPI1	LPSPI3_TRG_INPUTReserved	-
XBAR1_OUT118 SPI	LPSPI4_TRG_INPUT	-
XBAR1_OUT119	LPUART1_TRG_INPUT	-
XBAR1_OUT120	LPUART2_TRG_INPUT	-
XBAR1_OUT121	LPUART3_TRG_INPUT	-
XBAR1_OUT122	LPUART4_TRG_INPUT	-
XBAR1_OUT123	LPUART5_TRG_INPUT	-
XBAR1_OUT124	LPUART6_TRG_INPUT	-
XBAR1_OUT125	LPUART7_TRG_INPUT	-
XBAR1_OUT126	LPUART8_TRG_INPUT	-
XBAR1_OUT127	FLEXIO1_TRIGGER_IN0	-
XBAR1_OUT128	FLEXIO1_TRIGGER_IN1	-
XBAR1_OUT129	FLEXIO2_TRIGGER_IN0	-
XBAR1_OUT130	FLEXIO2_TRIGGER_IN1	-
XBAR1_OUT131	Reserved	-

Table 4-9. XBAR2 Output Assignments

XBAR2 Output	Assigned Output	Gate
XBAR2_OUT00	AOI1_IN00	-
XBAR2_OUT01	AOI1_IN01	-
XBAR2_OUT02	AOI1_IN02	-
XBAR2_OUT03	AOI1_IN03	-
XBAR2_OUT04	AOI1_IN04	-
XBAR2_OUT05	AOI1_IN05	-
XBAR2_OUT06	AOI1_IN06	-
XBAR2_OUT07	AOI1_IN07	-
XBAR2_OUT08	AOI1_IN08	-
XBAR2_OUT09	AOI1_IN09	-
XBAR2_OUT10	AOI1_IN10	-
XBAR2_OUT11	AOI1_IN11	-
XBAR2_OUT12	AOI1_IN12	-
XBAR2_OUT13	AOI1_IN13	-
XBAR2_OUT14	AOI1_IN14	-
XBAR2_OUT15	AOI1_IN15	-

Table 4-10. XBAR3 Output Assignments

XBAR3 Output	Assigned Output	Gate
XBAR3_OUT00	AOI2_IN00	-
XBAR3_OUT01	AOI2_IN01	-
XBAR3_OUT02	AOI2_IN02	-
XBAR3_OUT03	AOI2_IN03	-
XBAR3_OUT04	AOI2_IN04	-
XBAR3_OUT05	AOI2_IN05	-
XBAR3_OUT06	AOI2_IN06	-
XBAR3_OUT07	AOI2_IN07	-
XBAR3_OUT08	AOI2_IN08	-
XBAR3_OUT09	AOI2_IN09	-
XBAR3_OUT10	AOI2_IN10	-
XBAR3_OUT11	AOI2_IN11	-
XBAR3_OUT12	AOI2_IN12	-
XBAR3_OUT13	AOI2_IN13	-
XBAR3_OUT14	AOI2_IN14	-
XBAR3_OUT15	AOI2_IN15	-

Chapter 5
Direct Memory Access Multiplexer (DMAMUX)

5.1 Chip-specific DMAMUX information

Table 5-1. Reference links to related information

Topic	Related module(s)	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
DMA Mux Mapping	DMAMUX	DMA Mux

See the "DMA MUX Mapping" table in the "Interrupts, DMA Events, and XBAR Assignments" chapter for details about DMA source and channel information.

5.2 Overview

The Direct Memory Access Multiplexer (DMAMUX) routes DMA sources, called slots, to any of the 32 DMA channels.

5.2.1 Block diagram

The following figure illustrates the block diagram of DMAMUX.

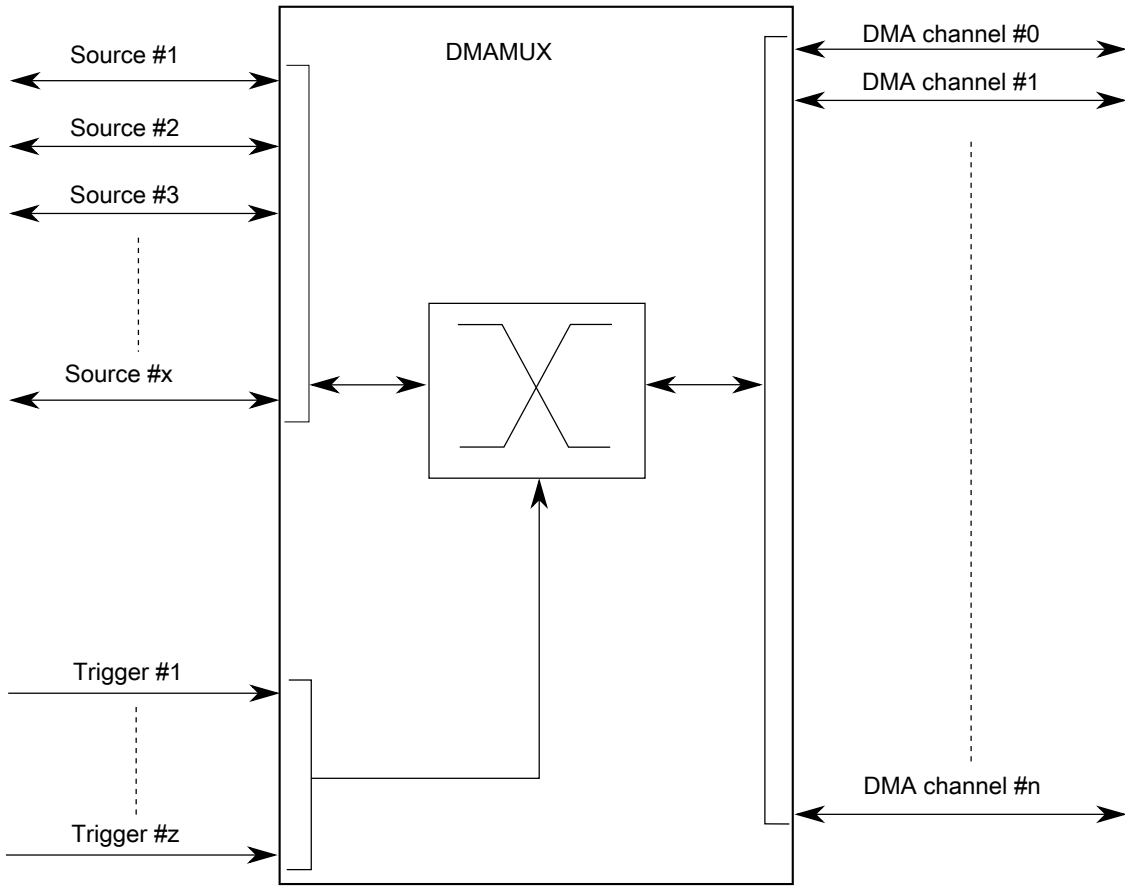


Figure 5-1. DMAMUX block diagram

5.2.2 Features

The DMAMUX module provides these features:

- Up to 128 peripheral slots can be routed to 32 channels.
- 32 independently selectable DMA channel routers.
  - Each channel output can be individually configured to be Always On and not depend on any of the peripheral slots.
  - The first 4 channels additionally provide a trigger functionality.
- Each channel router can be assigned to one of the possible peripheral DMA slots.
- On every memory map configuration change for a any channel, this module signals to the DMA Controller to reset the internal state machine for that channel and it can accept a new request based on the new configuration.

5.3 Functional description

The primary purpose of the DMAMUX is to provide flexibility in the system's use of the available DMA channels.

As such, configuration of the DMAMUX is intended to be a static procedure done during execution of the system boot code. However, if the procedure outlined in Enabling and configuring sources is followed, the configuration of the DMAMUX may be changed during the normal operation of the system.

Functionally, the DMAMUX channels may be divided into two classes:

- Channels that implement the normal routing functionality plus periodic triggering capability
- Channels that implement only the normal routing functionality

5.3.1 Modes of operation

The following operating modes are available:

- Disabled mode

  In this mode, the DMA channel is disabled. Because disabling and enabling of DMA channels is done primarily via the DMA configuration registers, this mode is used mainly as the reset state for a DMA channel in the DMA channel MUX. It may also be used to temporarily suspend a DMA channel while reconfiguration of the system takes place, for example, changing the period of a DMA trigger.

- Normal mode

  In this mode, a DMA source is routed directly to the specified DMA channel. The operation of the DMAMUX in this mode is completely transparent to the system.

- Periodic Trigger mode

  In this mode, a DMA source may only request a DMA transfer, such as when a transmit buffer becomes empty or a receive buffer becomes full, periodically.

  Configuration of the period is done in the registers of the periodic interrupt timer (PIT). This mode is available only for channels 0 to 3.

5.3.2 DMA channels with periodic triggering capability

Besides the normal routing functionality, the first 4 channels of the DMAMUX provide a special periodic triggering capability that can be used to provide an automatic mechanism to transmit bytes, frames, or packets at fixed intervals without the need for processor intervention.

The trigger is generated by the periodic interrupt timer (PIT); as such, the configuration of the periodic triggering interval is done via configuration registers in the PIT. See the section on periodic interrupt timer for more information on this topic.

Note

Because of the dynamic nature of the system (due to DMA channel priorities, bus arbitration, interrupt service routine lengths, etc.), the number of clock cycles between a trigger and the actual DMA transfer cannot be guaranteed.

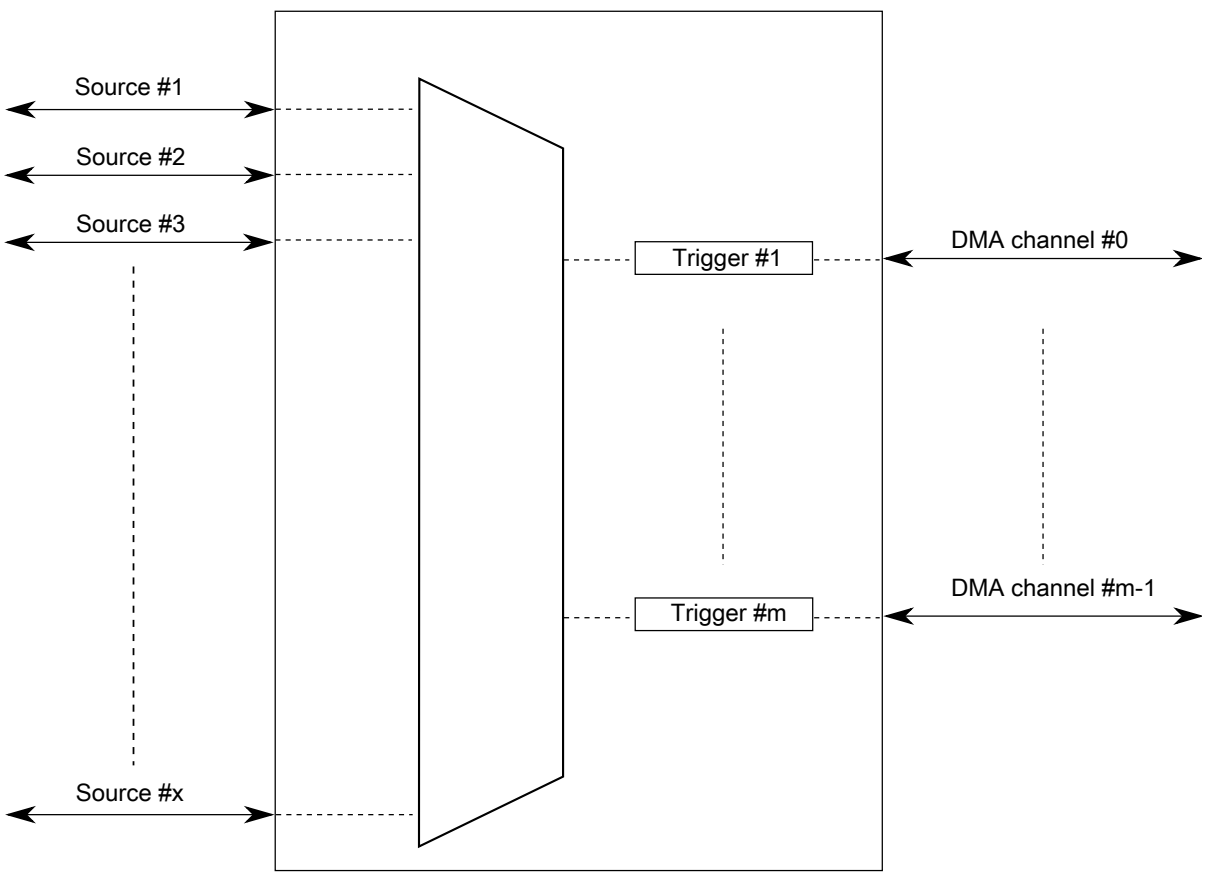


Figure 5-2. DMAMUX triggered channels

The DMA channel triggering capability allows the system to schedule regular DMA transfers, usually on the transmit side of certain peripherals, without the intervention of the processor. This trigger works by gating the request from the peripheral to the DMA until a trigger event has been seen. This is illustrated in the following figure.

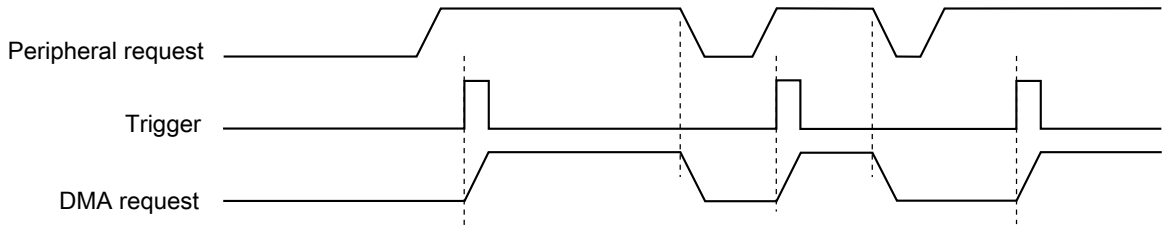


Figure 5-3. DMAMUX channel triggering: normal operation

After the DMA request has been serviced, the peripheral will negate its request, effectively resetting the gating mechanism until the peripheral reasserts its request and the next trigger event is seen. This means that if a trigger is seen, but the peripheral is not requesting a transfer, then that trigger will be ignored. This situation is illustrated in the following figure.

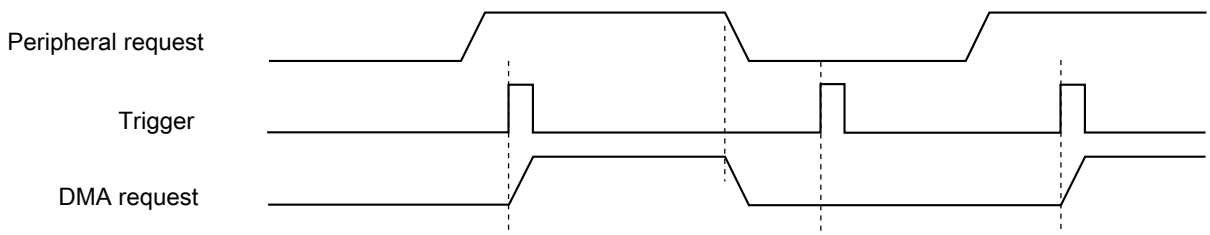


Figure 5-4. DMAMUX channel triggering: ignored trigger

This triggering capability may be used with any peripheral that supports DMA transfers, and is most useful for two types of situations:

- Periodically polling external devices on a particular bus

  As an example, the transmit side of an SPI is assigned to a DMA channel with a trigger, as described above. After it has been set up, the SPI will request DMA transfers, presumably from memory, as long as its transmit buffer is empty. By using a trigger on this channel, the SPI transfers can be automatically performed every 5 µs (as an example). On the receive side of the SPI, the SPI and DMA can be configured to transfer receive data into memory, effectively implementing a method to periodically read data from external devices and transfer the results into memory without processor intervention.

- Using the GPIO ports to drive or sample waveforms

By configuring the DMA to transfer data to one or more GPIO ports, it is possible to create complex waveforms using tabular data stored in on-chip memory. Conversely, using the DMA to periodically transfer data from one or more GPIO ports, it is possible to sample complex waveforms and store the results in tabular form in on-chip memory.

A more detailed description of the capability of each trigger, including resolution, range of values, and so on, may be found in the periodic interrupt timer section.

5.3.3 Always-enabled DMA sources

In addition to the peripherals that can be used as DMA sources, each DMA Channel can be individually configured to function as an Always Enabled source. Unlike the peripheral DMA sources, where the peripheral controls the flow of data during DMA transfers, the always enabled channel provide no such "throttling" of the data transfers. These sources are most useful in the following cases:

- Performing DMA transfers to/from GPIO—Moving data from/to one or more GPIO pins, either unthrottled (that is, as fast as possible), or periodically (using the DMA triggering capability).
- Performing DMA transfers from memory to memory—Moving data from memory to memory, typically as fast as possible, sometimes with software activation.
- Performing DMA transfers from memory to the external bus, or vice-versa—Similar to memory to memory transfers, this is typically done as quickly as possible.
- Any DMA transfer that requires software activation—Any DMA transfer that should be explicitly started by software.

In cases where software should initiate the start of a DMA transfer, an always-enabled DMA channel can be used to provide maximum flexibility. When activating a DMA channel via software, subsequent executions of the minor loop require that a new start event be sent. This can either be a new software activation, or a transfer request from the DMA channel MUX. The options for doing this are:

- Transfer all data in a single minor loop.

By configuring the DMA to transfer all of the data in a single minor loop (that is, major loop counter = 1), no reactivation of the channel is necessary. The disadvantage to this option is the reduced granularity in determining the load that the DMA transfer will impose on the system. For this option, the DMA channel must be disabled in the DMA channel MUX.

- Use explicit software reactivation.

  In this option, the DMA is configured to transfer the data using both minor and major loops, but the processor is required to reactivate the channel by writing to the DMA registers *after every minor loop*. For this option, the DMA channel must be disabled in the DMA channel MUX.

- Use an always-enabled DMA source.

  In this option, the DMA is configured to transfer the data using both minor and major loops, and the DMA channel MUX does the channel reactivation. For this option, the DMA channel should be enabled and configured as "always enabled" channel. Note that the reactivation of the channel can be continuous (DMA triggering is disabled) or can use the DMA triggering capability. In this manner, it is possible to execute periodic transfers of packets of data from one source to another, without processor intervention.

NOTE

When a channel is configured as "Always Enabled", then the peripheral DMA sources for that channel are ignored; i.e. SOURCE field has no effect.

5.3.4 Clocks

The following table describes the clock sources for DMAMUX. See the clocking chapter of your device to know the clock setting, configuration and gating information.

Table 5-2. DMAMUX clocks

Clock name	Description
ipg_clk	Peripheral clock
ipg_clk_s	Peripheral access clock

5.3.5 Reset

The reset state of each individual bit is shown in Memory map/register definition. In summary, after reset, all channels are disabled and must be explicitly enabled before use.

5.4 External signals

The DMAMUX has no external pins.

5.5 Initialization/application information

This section provides instructions for initializing the DMA channel MUX.

5.5.1 Enabling and configuring sources

To enable a source with periodic triggering:

1. Determine with which DMA channel the source will be associated. Note that only the first 4 DMA channels have periodic triggering capability.
2. Clear the CHCFG[ENBL] and CHCFG[TRIG] fields of the DMA channel.
3. Ensure that the DMA channel is properly configured in the DMA. The DMA channel may be enabled at this point.
4. Configure the corresponding timer.
5. Select the source to be routed to the DMA channel. Write to the corresponding CHCFG register, ensuring that the CHCFG[ENBL] and CHCFG[TRIG] fields are set.

NOTE

The following is an example. See the chip configuration details for the number of this device's DMA channels that have triggering capability.

To configure source #5 transmit for use with DMA channel 1, with periodic triggering capability:

1. Write 0x00000000 to CHCFG1.
2. Configure channel 1 in the DMA, including enabling the channel.
3. Configure a timer for the desired trigger interval.
4. Write 0xC0000005 to CHCFG1.

The following code example illustrates steps 1 and 4 above:

```
void DMAMUX_Init(uint8_t DMA_CH, uint8_t DMAMUX_SOURCE)
{
    DMAMUX_0.CHCFG[DMA_CH].B.SOURCE = DMAMUX_SOURCE;
    DMAMUX_0.CHCFG[DMA_CH].B.ENBL   = 1;
    DMAMUX_0.CHCFG[DMA_CH].B.TRIG   = 1;
}
```

To enable a source without periodic triggering:

1. Determine with which DMA channel the source will be associated. Note that only the first 4 DMA channels have periodic triggering capability.
2. Clear the CHCFG[ENBL] and CHCFG[TRIG] fields of the DMA channel.
3. Ensure that the DMA channel is properly configured in the DMA. The DMA channel may be enabled at this point.
4. Select the source to be routed to the DMA channel. Write to the corresponding CHCFG register, ensuring that CHCFG[ENBL] is setwhile CHCFG[TRIG] is cleared.

NOTE

The following is an example. See the chip configuration details for the number of this device's DMA channels that have triggering capability.

To configure source #5 transmit for use with DMA channel 1 with no periodic triggering capability :

1. Write 0x00000000 to CHCFG1.
2. Configure channel 1 in the DMA, including enabling the channel.
3. Write 0x80000005 to CHCFG1.

The following code example illustrates steps 1 and 3 above:

```
In File registers.h:
#define DMAMUX_BASE_ADDR     0x40021000/* Example only ! */
/* Following example assumes long is 32-bits */
volatile unsigned long *CHCFG0 = (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x0000);
volatile unsigned long *CHCFG1 = (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x0004);
volatile unsigned long *CHCFG2 = (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x0008);
volatile unsigned long *CHCFG3 = (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x000C);
volatile unsigned long *CHCFG4 = (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x0010);
volatile unsigned long *CHCFG5 = (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x0014);
volatile unsigned long *CHCFG6 = (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x0018);
volatile unsigned long *CHCFG7 = (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x001C);
volatile unsigned long *CHCFG8 = (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x0020);
volatile unsigned long *CHCFG9 = (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x0024);
volatile unsigned long *CHCFG10= (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x0028);
volatile unsigned long *CHCFG11= (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x002C);
volatile unsigned long *CHCFG12= (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x0030);
volatile unsigned long *CHCFG13= (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x0034);
volatile unsigned long *CHCFG14= (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x0038);
volatile unsigned long *CHCFG15= (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x003C);

In File main.c:
#include "registers.h"
:
:
*CHCFG1 = 0x00000000;
*CHCFG1 = 0x80000005;
```

To disable a source:

A particular DMA source may be disabled by not writing the corresponding source value into any of the CHCFG registers. Additionally, some module-specific configuration may be necessary. See the appropriate section for more details.

To switch the source of a DMA channel:

1. Disable the DMA channel in the DMA and reconfigure the channel for the new source.
2. Clear the CHCFG[ENBL] and CHCFG[TRIG] bits of the DMA channel.
3. Select the source to be routed to the DMA channel. Write to the corresponding CHCFG register, ensuring that the CHCFG[ENBL] and CHCFG[TRIG] fields are set.

To switch DMA channel 8 from source #5 transmit to source #7 transmit:

1. In the DMA configuration registers, disable DMA channel 8 and reconfigure it to handle the transfers to peripheral slot 7. This example assumes channel 8 doesn't have triggering capability.
2. Write 0x00000000 to CHCFG8.
3. Write 0x80000007 to CHCFG8. (In this example, setting CHCFG[TRIG] would have no effect due to the assumption that channel 8 does not support the periodic triggering functionality.)

The following code example illustrates steps 2 and 3 above:

```
In File registers.h:
#define DMAMUX_BASE_ADDR     0x40021000/* Example only ! */
/* Following example assumes long is 32-bits */
volatile unsigned long *CHCFG0 = (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x0000);
volatile unsigned long *CHCFG1 = (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x0004);
volatile unsigned long *CHCFG2 = (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x0008);
volatile unsigned long *CHCFG3 = (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x000C);
volatile unsigned long *CHCFG4 = (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x0010);
volatile unsigned long *CHCFG5 = (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x0014);
volatile unsigned long *CHCFG6 = (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x0018);
volatile unsigned long *CHCFG7 = (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x001C);
volatile unsigned long *CHCFG8 = (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x0020);
volatile unsigned long *CHCFG9 = (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x0024);
volatile unsigned long *CHCFG10= (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x0028);
volatile unsigned long *CHCFG11= (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x002C);
volatile unsigned long *CHCFG12= (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x0030);
volatile unsigned long *CHCFG13= (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x0034);
volatile unsigned long *CHCFG14= (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x0038);
volatile unsigned long *CHCFG15= (volatile unsigned long *) (DMAMUX_BASE_ADDR+0x003C);

In File main.c:
#include "registers.h"
:
:
*CHCFG8 = 0x00000000;
*CHCFG8 = 0x80000007;
```

5.5.1.1 Configuration options

Table 5-3. Channel Configuration Options

ENBL	TRIG	A_ON	Function	Mode
0	X	X	DMA channel is disabled	Disabled Mode
1	0	0	DMA channel is enabled with no triggering (transparent)	Normal Mode
1	1	0	DMA channel is enabled with triggering	Periodic Trigger Mode
1	0	1	DMA channel is always enabled	Always On Mode
1	1	1	DMA channel is always enabled with triggering	Always On Trigger Mode

5.6 Memory map/register definition

This section provides a detailed description of all memory-mapped registers in the DMAMUX.

5.6.1 DMAMUX register descriptions

5.6.1.1 DMAMUX memory map

DMAMUX base address: 400E_C000h

Offset	Register	Width (In bits)	Access	Reset value
0h - 7Ch	Channel a Configuration Register (CHCFG0 - CHCFG31)	32	RW	0000_0000h

5.6.1.2 Channel a Configuration Register (CHCFG0 - CHCFG31)

5.6.1.2.1 Offset

For a = 0 to 31:

Register	Offset
CHCFGa	0h + (a × 4h)

5.6.1.2.2 Function

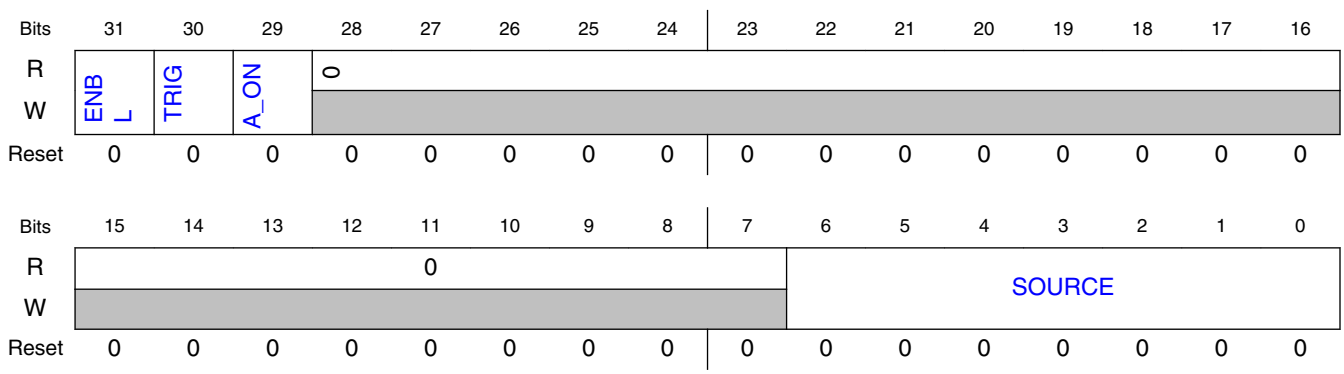
Each of the DMA channels can be independently enabled/disabled and associated with one of the DMA slots (peripheral slots or always-on slots) in the system.

NOTE

Setting multiple CHCFG registers with the same source value will result in unpredictable behavior. This is true, even if a channel is disabled (ENBL==0).

Before changing the trigger or source settings, a DMA channel must be disabled via CHCFGn[ENBL].

5.6.1.2.3 Diagram



5.6.1.2.4 Fields

Field	Function
31 ENBL	DMA Mux Channel Enable Enables the channel for DMA Mux. The DMA has separate channel enables/disables, which should be used to disable or reconfigure a DMA channel. 0b - DMA Mux channel is disabled 1b - DMA Mux channel is enabled
30 TRIG	DMA Channel Trigger Enable Enables the periodic trigger capability for the triggered DMA channel. 0b - Triggering is disabled. If triggering is disabled and ENBL is set, the DMA Channel will simply route the specified source to the DMA channel. (Normal mode) 1b - Triggering is enabled. If triggering is enabled and ENBL is set, the DMA_CH_MUX is in Periodic Trigger mode.
29 A_ON	DMA Channel Always Enable

Table continues on the next page...

Field	Function
	Enables the DMA Channel to be always ON. If TRIG bit is set, the module will assert request on every trigger. 0b - DMA Channel Always ON function is disabled 1b - DMA Channel Always ON function is enabled
28-7 —	Reserved field
6-0 SOURCE	DMA Channel Source (Slot Number) Specifies which DMA source, if any, is routed to a particular DMA channel. See the chip-specific DMA_CH_MUX information for details about the peripherals and their slot numbers.

Chapter 6
Enhanced Direct Memory Access (eDMA)

6.1 Chip-specific eDMA information

Table 6-1. Reference links to related information

Topic	Related module(s)	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
Interrupts, DMA Events and XBAR Assignments	-	Interrupts, DMA Events and XBAR Assignments

6.2 Overview

The enhanced direct memory access (eDMA) controller is a second-generation module capable of performing complex data transfers with minimal intervention from a host processor. The hardware microarchitecture includes:

- A DMA engine that performs:
  - Source address and destination address calculations
  - Data-movement operations
- Local memory containing transfer control descriptors for each of the 32 channels

6.2.1 Block diagram

The following figure illustrates the components of the eDMA system, including the eDMA module ("engine").

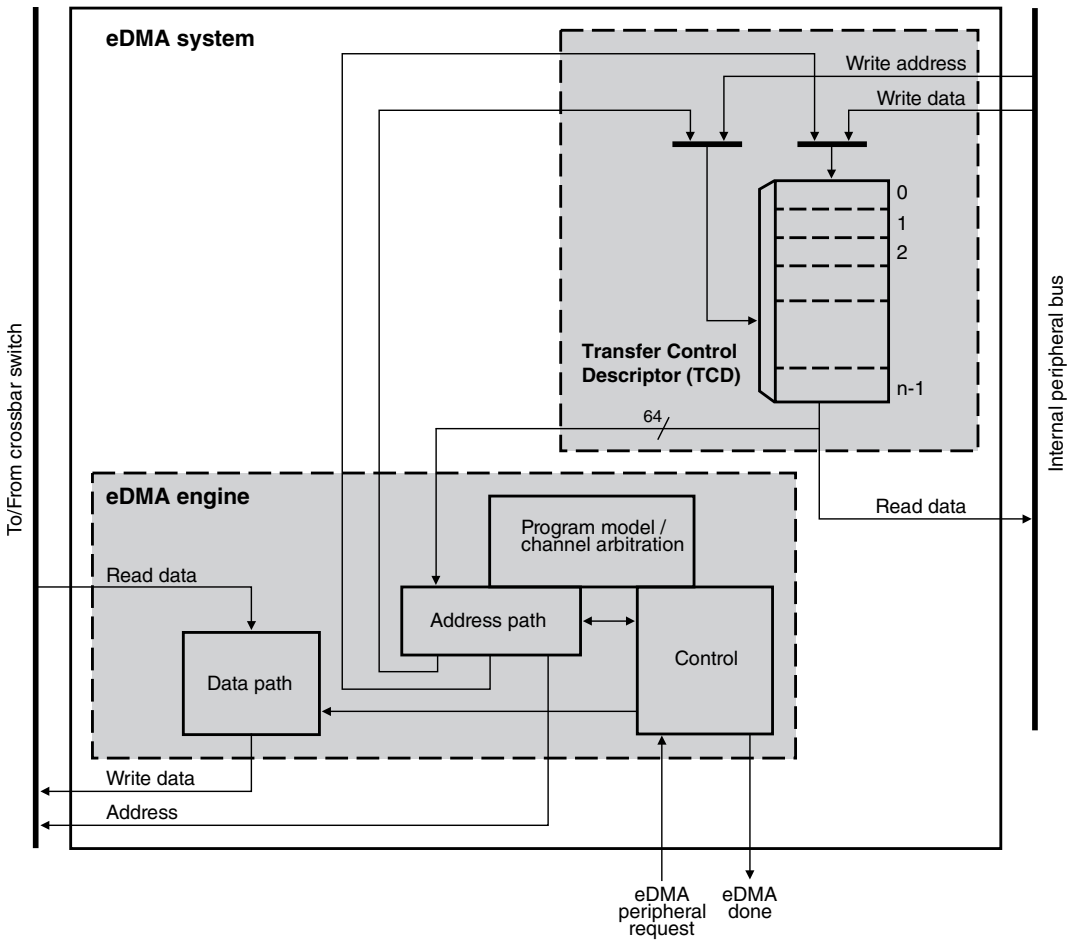


Figure 6-1. Block diagram

6.2.2 Block parts

The eDMA module comprises two major modules: the eDMA engine and the transfer-control descriptor local memory.

Table 6-2 describes the eDMA engine submodules.

Table 6-2. eDMA engine submodules

Submodule	Function
Address path	The address path block: • Provides registered versions of two channel Transfer Control Descriptors (TCDs)—channel x (normal start) and channel y (preemption start) • Manages all master bus-address calculations

Table continues on the next page...

Table 6-2. eDMA engine submodules (continued)

Submodule	Function
	All channels provide the same functionality. This structure enables preemption of data transfers associated with an active channel (after completion of a read/write sequence) if the eDMA engine asserts a higher priority channel activation. After eDMA activates a channel, it runs until the minor loop completes, unless preempted by a higher priority channel. This provides a mechanism (enabled by DCHPRI*n*[ECP]) in which the eDMA engine can preempt a large data move operation to minimize the time another channel stalls. When the eDMA engine selects a channel to execute, it reads the contents of the channel TCD from local memory and loads it into one of the following: • The address path channel x registers (normal start) • The address path channel y registers (preemption start) After the minor loop execution completes, the address path hardware writes the new values for the TCD*n*_{SADDR, DADDR, CITER} back to local memory. If the major iteration count completes, the eDMA engine performs additional processing, including: • Final address pointer updates • Reloading the TCD*n*_CITER field • A possible fetch of the next TCD*n* from memory as part of a scatter/gather operation.
Data path	The data path block implements the bus master read/write data path. It includes a data buffer and the necessary multiplex logic to support any required data alignment. The internal read data bus is the primary input, and the internal write data bus is the primary output. The address and data path modules directly support the two-stage pipelined internal bus. The address path module represents the first stage of the bus pipeline (address phase). The data path module implements the second stage of the pipeline (data phase).
Programming model/ channel arbitration	This block implements: • The first section of the eDMA programming model • Channel arbitration logic The programming model registers connect to the chip's internal peripheral bus. The eDMA peripheral request inputs and interrupt request outputs also connect to this block (via control logic).
Control	The control block provides all control functions for the eDMA engine. For data transfers in which the source size (SSIZE) and destination size (DSIZE) are equal, the eDMA engine performs a series of source read/destination write operations until it has transferred the number of bytes specified in the minor loop byte count (NBYTES). For TCDs in which the source and destination sizes are not equal, the eDMA engine executes multiple accesses of the smaller size data for each reference of the larger size. For example, if the source size (SSIZE) references 16-bit data and the destination size (DSIZE) is 32-bit data, eDMA performs two reads, then one 32-bit write.

Table 6-3 explains the partitioning of the TCD local memory.

Table 6-3. Transfer control descriptor memory

Submodule	Description
Memory controller	The Memory controller logic implements the required dual-ported controller, managing accesses from the eDMA engine as well as references from the internal peripheral bus. If simultaneous accesses occur, the eDMA engine receives priority and the peripheral transaction stalls.
Memory array	The Memory array provides TCD storage for the transfer profile for each channel.

6.2.3 Features

The eDMA module is a highly programmable data-transfer engine optimized to minimize any required intervention from the host processor. Use it for applications where you statically know the size of the data to be transferred and do not define the size within the transferred data itself. The eDMA module features:

- All data movement via dual-address transfers: read from source, write to destination
  - Programmable source and destination addresses and transfer size
  - Support for enhanced addressing modes
- 32-channel implementation performs complex data transfers with minimal intervention from a host processor
  - Internal data buffer, used as temporary storage to support 16- and 32-byte transfers
  - Connections to the crossbar switch (AXBS) for bus mastering the data movement
- TCD supports two-deep, nested transfer operations
  - 32-byte TCD stored in local memory for each channel
  - An inner data transfer loop defined by a minor byte transfer count
  - An outer data transfer loop defined by a major iteration count
- Channel activation via one of three methods:
  - Explicit software initiation
  - Initiation via a channel-to-channel linking mechanism for continuous transfers
  - Peripheral-paced hardware requests, one per channel
- Fixed-priority and round-robin channel arbitration
- Channel completion notification via programmable interrupt requests
  - One interrupt per channel. eDMA engine can generate an interrupt when major iteration count completes
  - Programmable error terminations per channel and logically summed together to form one error interrupt to the interrupt controller

- Programmable support for scatter/gather DMA processing
- Support for complex data structures

NOTE

In the discussion of this module, *n* is the channel number.

6.3 Functional description

The operation of the eDMA is described in the following subsections.

6.3.1 Modes of operation

eDMA operates in the following modes:

Table 6-4. Modes of operation

Mode	Description
Normal	In Normal mode, eDMA transfers data from a source to a destination. The source and destination can be a memory block or an I/O block capable of operation with eDMA. A *service request* initiates a transfer of a specific number of bytes (NBYTES) as specified in the TCD. • The *minor loop* is the sequence of read and write operations that transfers the NBYTES of data for a service request. • Each service request executes one iteration of the major loop, transferring NBYTES of data.
Debug	DMA operation is configurable in Debug mode via Control (CR) • If CR[EDBG] = 0, eDMA continues to operate normally when the chip is in debug mode. • If CR[EDBG] = 1, eDMA stops transferring data when the chip enters debug mode. If a channel is active when eDMA enters Debug mode, eDMA continues operation until the channel retires.
Wait	Before entering Wait mode, eDMA attempts to complete any transfer that is in progress. After the transfer completes, the chip enters Wait mode.

6.3.2 eDMA basic data flow

The basic flow of a data transfer can be partitioned into three segments.

As shown in the following diagram, the first segment involves the channel activation:

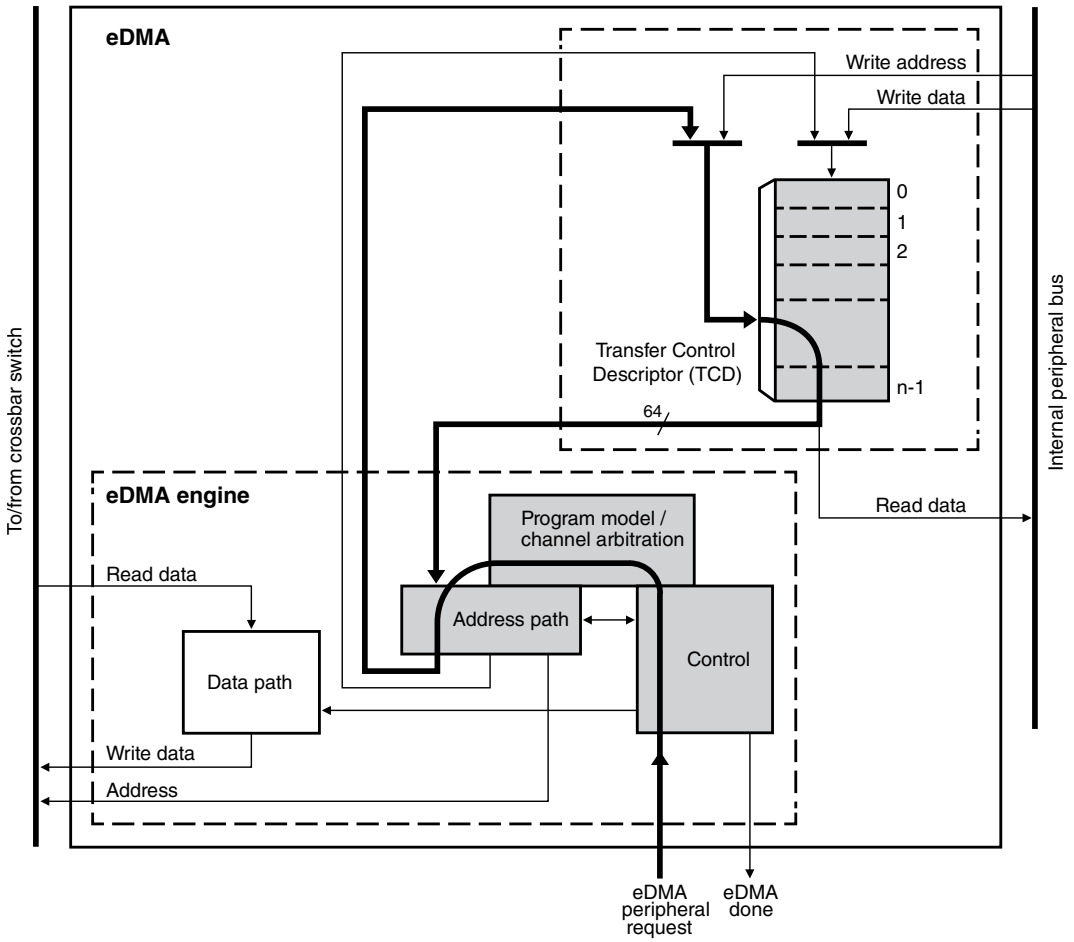


Figure 6-2. eDMA operation, part 1

This example uses the assertion of the eDMA peripheral request signal to request service for channel *n*. Channel activation via software and the TCD*n*_CSR[START] bit follows the same basic flow as peripheral requests. The eDMA request input signal is registered internally and then routed through the eDMA engine: first through the control module, then into the program model and channel arbitration. In the next cycle, the channel arbitration executes, using either the fixed-priority or round-robin algorithm. After arbitration is complete, the activated channel number is sent through the address path and converted into the required address to access the local memory for TCD*n*. Next, the TCD memory is accessed and the required descriptor read from the local memory and loaded into the eDMA engine's internal register file. The TCD memory is 64 bits wide to minimize the time needed to fetch the activated channel descriptor and load it into the internal register file.

The following diagram illustrates the second part of the basic data flow:

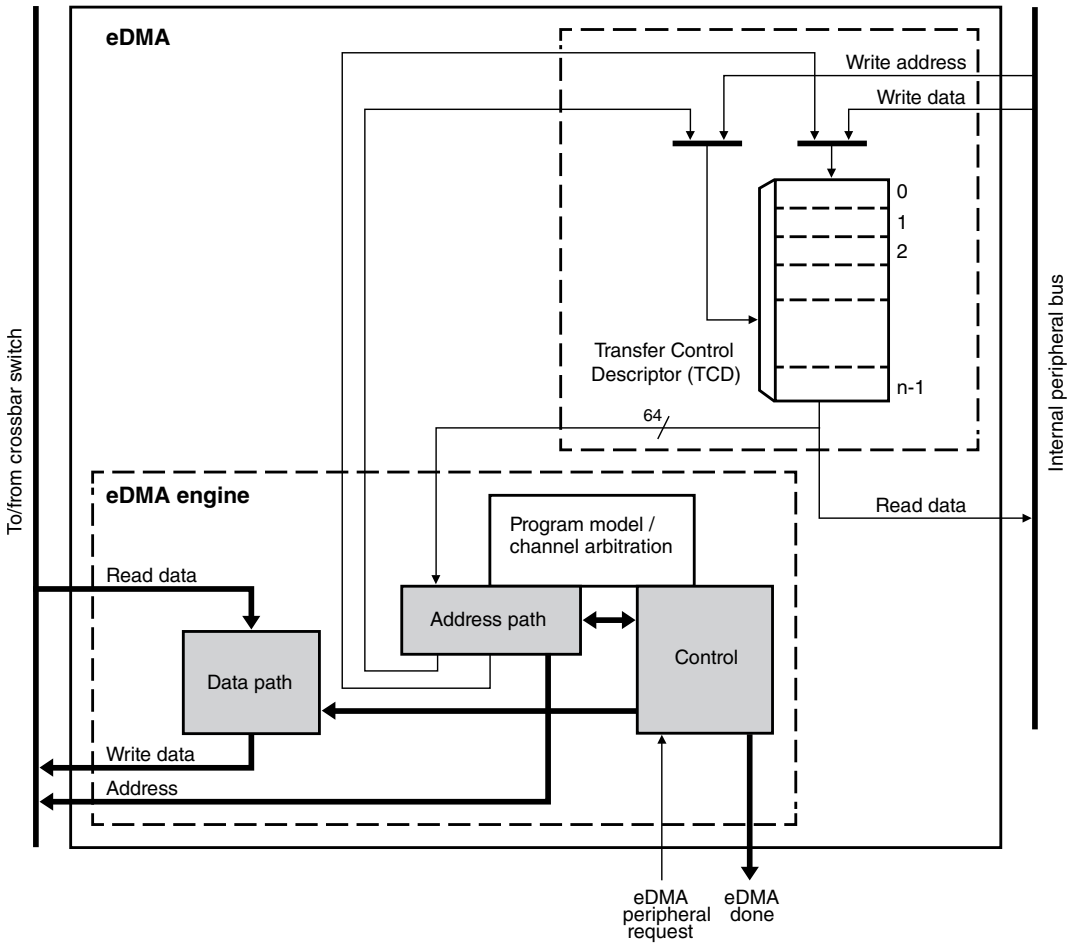


Figure 6-3. eDMA operation, part 2

The modules associated with the data transfer (address path, data path, and control) execute sequentially through the required source reads and destination writes to perform the data movement. The source reads are initiated and the fetched data is temporarily stored in the data path block until it is gated onto the internal bus during the destination write. This source read/destination write processing continues until the minor byte count has transferred.

After the minor byte count has moved, the final phase of the basic data flow is performed. In this segment, the address path logic performs the required updates to certain fields in the appropriate TCD, for example, SADDR, DADDR, CITER. If the major iteration count is exhausted, additional operations are performed. These include the final address adjustments and reloading of the BITER field into the CITER. Assertion of an optional interrupt request also occurs at this time, as does a possible fetch of a new TCD from memory using the scatter/gather address pointer included in the descriptor (if scatter/gather is enabled). The updates to the TCD memory and the assertion of an interrupt request are shown in the following diagram.

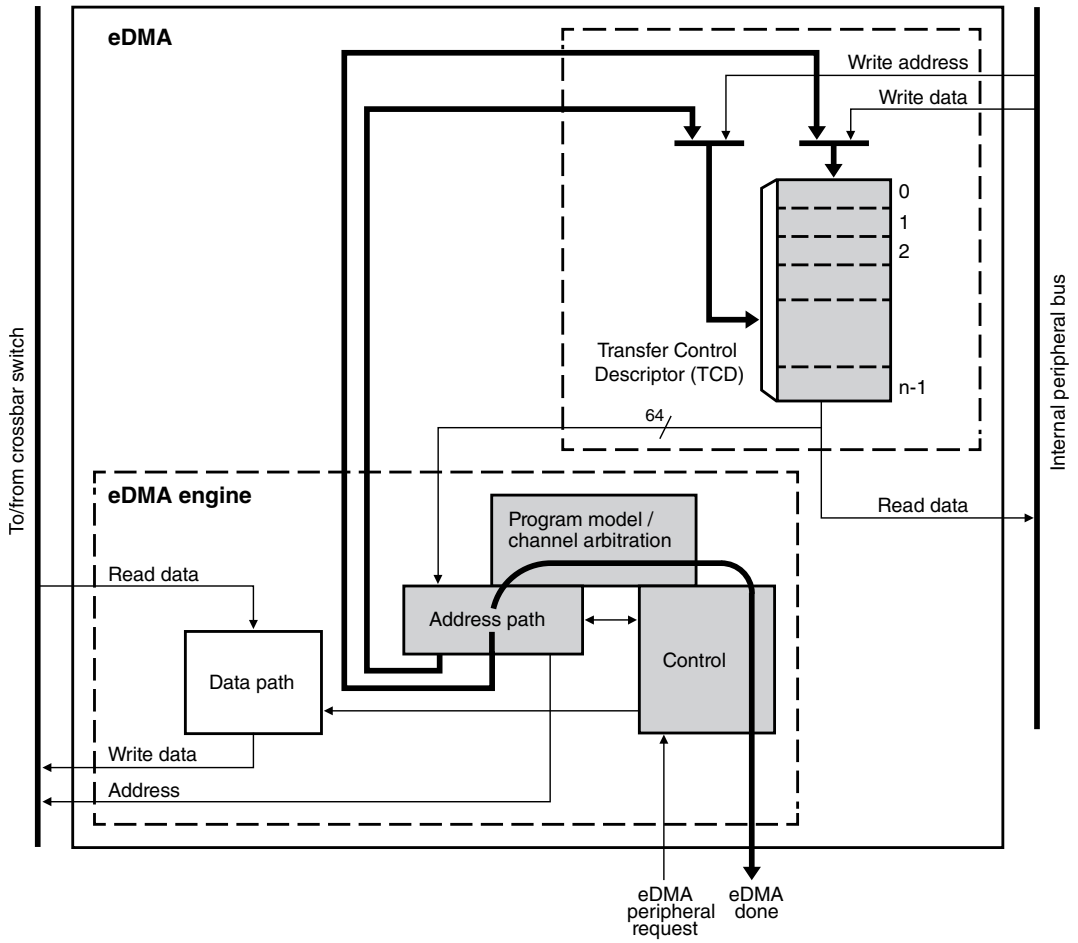


Figure 6-4. eDMA operation, part 3

6.3.3 Fault reporting and handling

Channel errors are reported in the Error Status register (DMAx_ES) and can be caused by:

- A configuration error, which is an illegal setting in the transfer-control descriptor or an illegal priority register setting in Fixed-Arbitration mode, or
- An error termination to a bus master read or write cycle

A configuration error is reported when the starting source or destination address, source or destination offsets, minor loop byte count, or the transfer size represent an inconsistent state. Each of these possible causes is detailed below:

- The addresses and offsets must be aligned on 0-modulo transfer size boundaries.
- The minor loop byte count must be a multiple of the source and destination transfer sizes.

- All source reads and destination writes must be configured to the natural boundary of the programmed transfer size respectively.
- In fixed arbitration mode, a configuration error is caused by any two channel priorities being equal. All channel priority levels must be unique when fixed arbitration mode is enabled.

NOTE

When two channels have the same priority, a channel priority error exists and is reported in the Error Status register. However, the channel number is not reported in the Error Status register. When all of the channel priorities within a group are not unique, the channel number selected by arbitration is undetermined.

To aid in Channel Priority Error (CPE) debug, set the Halt On Error bit in the DMA's Control register. If all channel priorities within a group are not unique, the DMA is halted after the CPE error is recorded. The DMA remains halted and does not process any channel service requests. After all of the channel priorities are set to unique numbers, the DMA may be enabled again by clearing the HALT bit.

- If a scatter/gather operation is enabled upon channel completion, a configuration error is reported if the scatter/gather address (DLAST_SGA) is not aligned on a 32-byte boundary.
- If minor loop channel linking is enabled upon channel completion, a configuration error is reported when the link is attempted if the TCDn_CITER[ELINK] bit does not equal the TCDn_BITER[ELINK] bit.

If enabled, all configuration error conditions, except the scatter/gather and minor-loop link errors, report as the channel activates and asserts an error interrupt request. A scatter/gather configuration error is reported when the scatter/gather operation begins at major loop completion, when properly enabled. A minor loop channel link configuration error is reported when the link operation is serviced at minor loop completion.

If a system bus read or write is terminated with an error, the data transfer is stopped and the appropriate bus error flag set. In this case, the state of the channel's transfer control descriptor is updated by the eDMA engine with the current source address, destination address, and current iteration count at the point of the fault. When a system bus error occurs, the channel terminates after the next transfer. Due to pipeline effect, the next transfer is already in progress when the bus error is received by the eDMA. If a bus error occurs on the last read prior to beginning the write sequence, the write executes using the

data captured during the bus error. If a bus error occurs on the last write prior to switching to the next read sequence, the read sequence executes before the channel terminates due to the destination bus error.

A transfer may be canceled by software with the CR[CX] bit. When a cancel transfer request is recognized, the DMA engine stops processing the channel. The current read-write sequence is allowed to finish. If the cancel occurs on the last read-write sequence of a major or minor loop, the cancel request is discarded and the channel retires normally.

The error cancel transfer is the same as a cancel transfer except the Error Status register (DMAx_ES) is updated with the canceled channel number and ECX is set. The TCD of a canceled channel contains the source and destination addresses of the last transfer saved in the TCD. If the channel needs to be restarted, you must re-initialize the TCD because the aforementioned fields no longer represent the original parameters. When a transfer is canceled by the error cancel transfer mechanism, the channel number is loaded into DMA_ES[ERRCHN] and ECX and VLD are set. In addition, an error interrupt may be generated if enabled.

NOTE

The cancel transfer request enables you to stop a large data transfer when the full data transfer is no longer needed. The cancel transfer bit does not abort the channel. It simply stops the transferring of data and then retires the channel through its normal shutdown sequence. The application software must manage the context of the cancel. If an interrupt is desired (or not), then the interrupt should be enabled (or disabled) before the cancel request. The application software must clean up the transfer control descriptor because the full transfer did not occur.

The occurrence of any error causes the eDMA engine to stop normal processing of the active channel immediately (it goes to its error processing states and the transaction to the system bus still has pipeline effect), and the appropriate channel bit in the eDMA error register is asserted. At the same time, the details of the error condition are loaded into the Error Status register (DMAx_ES). The major loop complete indicators, setting the transfer control descriptor DONE flag and the possible assertion of an interrupt request, are not affected when an error is detected. After the error status has been updated, the eDMA engine continues operating by servicing the next appropriate channel. A channel that experiences an error condition is not automatically disabled. If a channel is terminated by an error and then issues another service request before the error is fixed, that channel executes and terminates with the same error condition.

6.3.4 Channel preemption

Channel preemption is enabled on a per-channel basis by setting DCHPRIn[ECP]. Channel preemption enables the executing channel's data transfers to temporarily be suspended in favor of starting a higher priority channel. After the preempting channel has completed its minor loop data transfers, the preempted channel is restored and resumes execution. After the restored channel completes one read/write sequence, it is again eligible for preemption. If any higher priority channel is requesting service, the restored channel is suspended and the higher priority channel is serviced. Nested preemption, that is, attempting to preempt a preempting channel, is not supported. After a preempting channel begins execution, it cannot be preempted. Preemption is available only when fixed arbitration is selected.

A channel's ability to preempt another channel can be disabled by setting DCHPRIn[DPA]. When a channel's preemption ability is disabled, that channel cannot suspend a lower priority channel's data transfer, regardless of the lower priority channel's ECP setting. This enables a pool of low priority, large data-moving channels to be defined. These low priority channels can be configured to not preempt each other, thus preventing a low priority channel from consuming the preempt slot normally available to a true, high priority channel.

6.3.5 Clocks

The following table describes the clock sources for eDMA. Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 6-5. eDMA clocks

Clock name	Description
edma_hclk	Module clock
ipg_clk	Peripheral clock

6.4 Initialization/application information

The following sections discuss initialization of the eDMA and programming considerations.

6.4.1 eDMA initialization

To initialize the eDMA:

1. Write to the CR if a configuration other than the default is desired.
2. Write the channel priority levels to the DCHPRI*n* registers if a configuration other than the default is desired.
3. Enable error interrupts in the EEI register if desired.
4. Write the 32-byte TCD for each channel that may request service.
5. Enable any hardware service requests via the ERQ register.
6. Request channel service via either:
   - Software: setting TCD*n*_CSR[START]
   - Hardware: slave device asserting its eDMA peripheral request signal

After any channel requests service, a channel is selected for execution based on the arbitration and priority levels in the programming model. The eDMA engine reads the entire TCD, including the TCD control and status fields, as shown in Table 6-6, for the selected channel into its internal address path module.

As the TCD is read, the first transfer is initiated on the system bus, unless a configuration error is detected. Transfers from the source, as defined by TCD*n*_SADDR, to the destination, as defined by TCD*n*_DADDR, continue until the number of bytes specified by TCD*n*_NBYTES have been transferred.

When the transfer is complete, the eDMA engine's local TCD*n*_SADDR, TCD*n*_DADDR, and TCD*n*_CITER are written back to the main TCD memory and any minor loop channel linking is performed, if enabled. If the major loop is exhausted, further post-processing executes, such as interrupts, major loop channel linking, and scatter/gather operations, if enabled.

Table 6-6. TCD control and status fields

TCD*n*_CSR field name	Description
START	Control bit to start channel explicitly when using a software-initiated DMA service (automatically cleared by hardware)
ACTIVE	Status bit indicating the channel is currently in execution
DONE	Status bit indicating major loop completion (cleared by software when using a software-initiated DMA service)
DREQ	Control bit to disable DMA request at end of major loop completion when using a hardware-initiated DMA service
BWC	Control bits for throttling bandwidth control of a channel
ESG	Control bit to enable scatter/gather feature
INTHALF	Control bit to enable interrupt when major loop is half complete
INTMAJOR	Control bit to enable interrupt when major loop completes

The following figure shows how each DMA request initiates one minor-loop transfer, or iteration, without CPU intervention. DMA arbitration can occur after each minor loop, and one level of minor loop DMA preemption is allowed. The number of minor loops in a major loop is specified by the beginning iteration count (BITER).

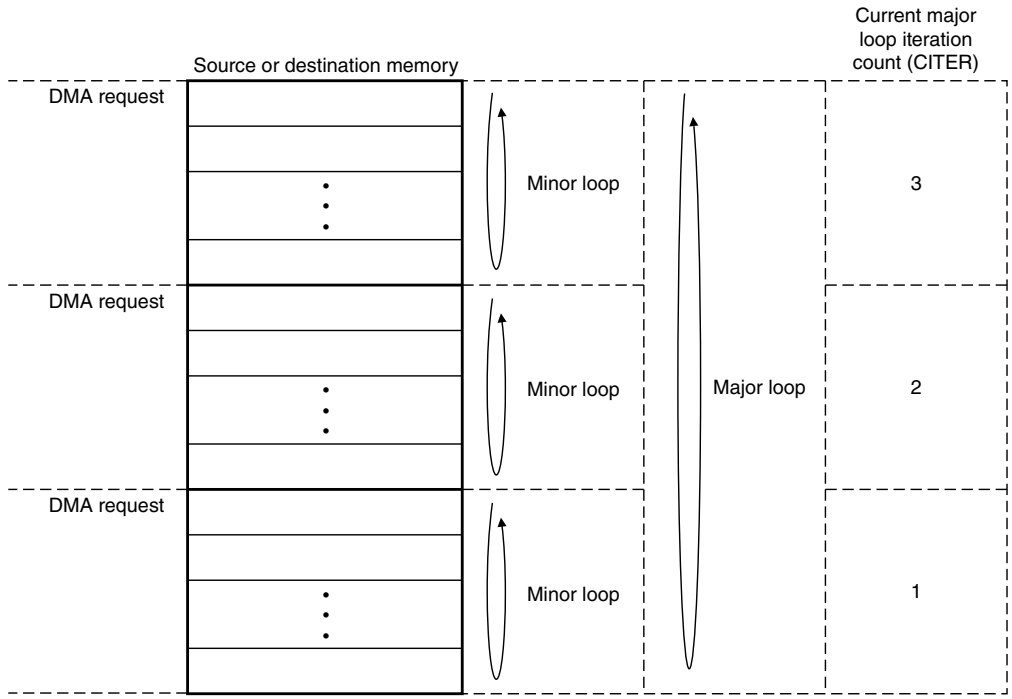


Figure 6-5. Example of multiple loop iterations

The following figure lists the memory array terms and how the TCD settings interrelate.

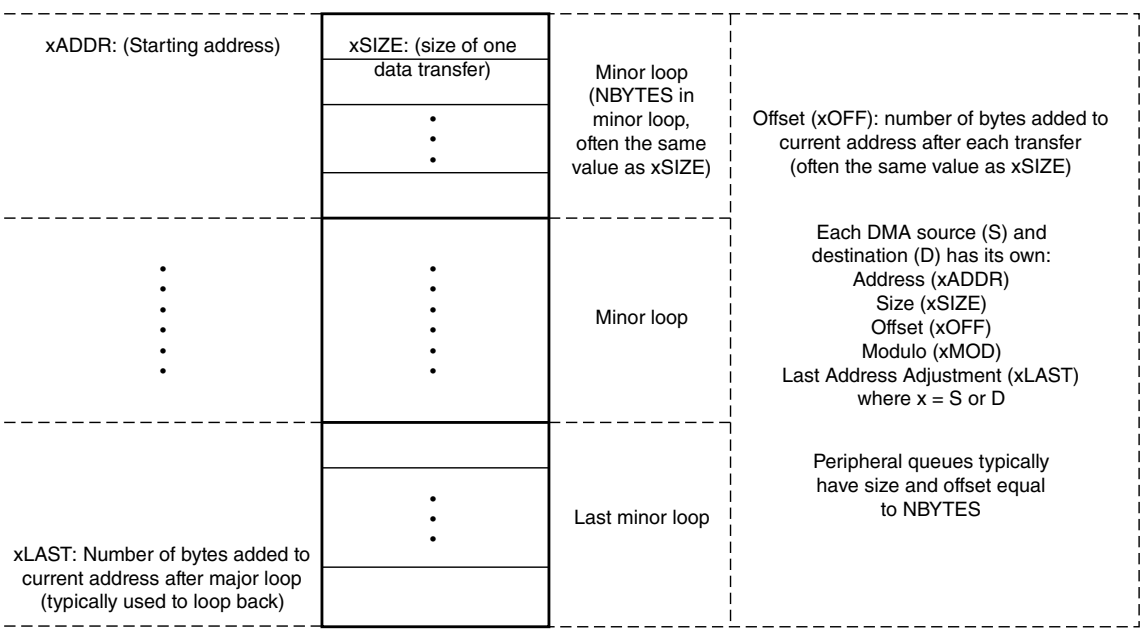


Figure 6-6. Memory array terms

6.4.2 Programming errors

The eDMA performs various tests on the transfer control descriptor to verify consistency in the descriptor data. Most programming errors are reported on a per-channel basis with the exception of channel priority error (ES[CPE]).

For all error types other than group or channel priority errors, the channel number causing the error is recorded in the Error Status register (DMAx_ES). If the error source is not removed before the next activation of the problem channel, the error is detected and recorded again.

Channel priority errors are identified within a group after that group has been selected as the active group. For example:

1. The eDMA is configured for fixed group and fixed channel arbitration modes.
2. Group 1 is the highest priority and all channels are unique in that group.
3. Group 0 is the next highest priority and has two channels with the same priority level.
4. If Group 1 has any service requests, those requests will be executed.
5. After all of Group 1 requests have completed, Group 0 will be the next active group.
6. If Group 0 has a service request, then an undefined channel in Group 0 will be selected and a channel priority error will occur.
7. This repeats until the all of Group 0 requests have been removed or a higher priority Group 1 request comes in.

In this sequence, for item 2, the eDMA acknowledge lines will assert only if the selected channel is requesting service via the eDMA peripheral request signal. If interrupts are enabled for all channels, the user will get an error interrupt, but the channel number for the ERR register and the error interrupt request line may be wrong because they reflect the selected channel. A group priority error is global and any request in any group will cause a group priority error.

If priority levels are not unique, when any channel requests service, a channel priority error is reported. The highest channel/group priority with an active request is selected, but the lowest numbered channel with that priority is selected by arbitration and executed by the eDMA engine. The hardware service request handshake signals, error interrupts, and error reporting are associated with the selected channel.

6.4.3 Arbitration mode considerations

This section discusses arbitration considerations for the eDMA.

6.4.3.1 Fixed group arbitration, Fixed channel arbitration

In this mode, the channel service request from the highest priority channel in the highest priority group is selected to execute. If the eDMA is programmed so that the channels within one group use "fixed" priorities, and that group is assigned the highest "fixed" priority of all groups, that group can take all the bandwidth of the eDMA controller. That is, no other groups will be serviced if there is always at least one DMA request pending on a channel in the highest priority group when the controller arbitrates the next DMA request. The advantage of this scenario is that latency can be small for channels that need to be serviced quickly. Preemption is available in this scenario only.

6.4.3.2 Fixed group arbitration, Round-robin channel arbitration

The highest priority group with a request will be serviced. Lower priority groups will be serviced if no pending requests exist in the higher priority groups.

Within each group, c hannels are serviced starting with the highest channel number and rotating through to the lowest channel number without regard to the channel priority levels assigned within the group.

This scenario could cause the same bandwidth consumption problem as indicated in Fixed group arbitration, Fixed channel arbitration, but all the channels in the highest priority group will be serviced. Service latency will be short on the highest priority group, but could potentially be very much longer as the group priority decreases.

6.4.4 DMA transfer examples

This section presents examples of how to perform DMA transfers with the eDMA.

6.4.4.1 Single request

To perform a simple transfer of n bytes of data with one activation, set the major loop to one (TCD*n*_CITER = TCD*n*_BITER = 1). The data transfer begins after the channel service request is acknowledged and the channel is selected to execute. After the transfer is complete, TCD*n*_CSR[DONE] is set and an interrupt generates if properly enabled.

For example, the following TCD entry is configured to transfer 16 bytes of data. The eDMA is programmed for one iteration of the major loop transferring 16 bytes per iteration. The source memory has an 8-bit memory port located at 0x1000. The

destination memory has a 32-bit port located at 0x2000. The address offsets are programmed in increments to match the transfer size: one byte for the source and four bytes for the destination. The final source and destination addresses are adjusted to return to their beginning values.

```
TCDn_CITER = TCDn_BITER = 1
TCDn_NBYTES = 16
TCDn_SADDR = 0x1000
TCDn_SOFF = 1
TCDn_ATTR[SSIZE] = 0
TCDn_SLAST = -16
TCDn_DADDR = 0x2000
TCDn_DOFF = 4
TCDn_ATTR[DSIZE] = 2
TCDn_DLAST_SGA= -16
TCDn_CSR[INTMAJOR] = 1
TCDn_CSR[START] = 1 (Should be written last after all other fields have been initialized)
All other TCDn fields = 0
```

This generates the following event sequence:

1. User write to the TCD*n*_CSR[START] bit requests channel service.
2. The channel is selected by arbitration for servicing.
3. eDMA engine writes: TCD*n*_CSR[DONE] = 0, TCD*n*_CSR[START] = 0, TCD*n*_CSR[ACTIVE] = 1.
4. eDMA engine reads: channel TCD data from local memory to internal register file.
5. The source-to-destination transfers are executed as follows:
   a. Read byte from location 0x1000, read byte from location 0x1001, read byte from 0x1002, read byte from 0x1003.
   b. Write 32 bits to location 0x2000 → first iteration of the minor loop.
   c. Read byte from location 0x1004, read byte from location 0x1005, read byte from 0x1006, read byte from 0x1007.
   d. Write 32 bits to location 0x2004 → second iteration of the minor loop.
   e. Read byte from location 0x1008, read byte from location 0x1009, read byte from 0x100A, read byte from 0x100B.
   f. Write 32 bits to location 0x2008 → third iteration of the minor loop.
   g. Read byte from location 0x100C, read byte from location 0x100D, read byte from 0x100E, read byte from 0x100F.
   h. Write 32 bits to location 0x200C → last iteration of the minor loop → major loop complete.
6. The eDMA engine writes: TCD*n*_SADDR = 0x1000, TCD*n*_DADDR = 0x2000, TCD*n*_CITER = 1 (TCD*n*_BITER).
7. The eDMA engine writes: TCD*n*_CSR[ACTIVE] = 0, TCD*n*_CSR[DONE] = 1, INT[*n*] = 1.
8. The channel retires and the eDMA goes idle or services the next channel.

6.4.4.2 Multiple requests

The following example transfers 32 bytes via two hardware requests, but is otherwise the same as the previous example. The only fields that change are the major loop iteration count and the final address offsets. The eDMA is programmed for two iterations of the major loop, transferring 16 bytes per iteration. After the channel's hardware requests are enabled in the ERQ register, the slave device initiates channel service requests.

```
TCDn_CITER = TCDn_BITER = 2
TCDn_SLAST = -32
TCDn_DLAST_SGA = -32
```

This would generate the following sequence of events:

1. First hardware, that is, the eDMA peripheral, requests channel service.
2. The channel is selected by arbitration for servicing.
3. eDMA engine writes: TCD*n*_CSR[DONE] = 0, TCD*n*_CSR[START] = 0, TCD*n*_CSR[ACTIVE] = 1.
4. eDMA engine reads: channel TCD*n* data from local memory to internal register file.
5. The source to destination transfers are executed as follows:
   a. Read byte from location 0x1000, read byte from location 0x1001, read byte from 0x1002, read byte from 0x1003.
   b. Write 32 bits to location 0x2000 → first iteration of the minor loop.
   c. Read byte from location 0x1004, read byte from location 0x1005, read byte from 0x1006, read byte from 0x1007.
   d. Write 32 bits to location 0x2004 → second iteration of the minor loop.
   e. Read byte from location 0x1008, read byte from location 0x1009, read byte from 0x100A, read byte from 0x100B.
   f. Write 32 bits to location 0x2008 → third iteration of the minor loop.
   g. Read byte from location 0x100C, read byte from location 0x100D, read byte from 0x100E, read byte from 0x100F.
   h. Write 32 bits to location 0x200C → last iteration of the minor loop.
6. eDMA engine writes: TCD*n*_SADDR = 0x1010, TCD*n*_DADDR = 0x2010, TCD*n*_CITER = 1.
7. eDMA engine writes: TCD*n*_CSR[ACTIVE] = 0.
8. The channel retires → one iteration of the major loop. The eDMA goes idle or services the next channel.
9. Second hardware, that is, eDMA peripheral, requests channel service.
10. The channel is selected by arbitration for servicing.
11. eDMA engine writes: TCD*n*_CSR[DONE] = 0, TCD*n*_CSR[START] = 0, TCD*n*_CSR[ACTIVE] = 1.
12. eDMA engine reads channel TCD data from local memory to internal register file.
13. The source to destination transfers are executed as follows:

a. Read byte from location 0x1010, read byte from location 0x1011, read byte from 0x1012, read byte from 0x1013.
b. Write 32 bits to location 0x2010 → first iteration of the minor loop.
c. Read byte from location 0x1014, read byte from location 0x1015, read byte from 0x1016, read byte from 0x1017.
d. Write 32 bits to location 0x2014 → second iteration of the minor loop.
e. Read byte from location 0x1018, read byte from location 0x1019, read byte from 0x101A, read byte from 0x101B.
f. Write 32 bits to location 0x2018 → third iteration of the minor loop.
g. Read byte from location 0x101C, read byte from location 0x101D, read byte from 0x101E, read byte from 0x101F.
h. Write 32 bits to location 0x201C → last iteration of the minor loop → major loop complete.

14. eDMA engine writes: TCD*n*_SADDR = 0x1000, TCD*n*_DADDR = 0x2000, TCD*n*_CITER = 2 (TCD*n*_BITER).
15. eDMA engine writes: TCD*n*_CSR[ACTIVE] = 0, TCD*n*_CSR[DONE] = 1, INT[n] = 1.
16. The channel retires → major loop complete. The eDMA goes idle or services the next channel.

6.4.4.3 Using the modulo feature

The modulo feature of the eDMA provides the ability to implement a circular data queue in which the size of the queue is a power of 2. MOD is a 5-bit field for the source and destination in the TCD, and it specifies which lower address bits increment from their original value after the address+offset calculation. All upper address bits remain the same as in the original value. A setting of zero for this field disables the modulo feature.

The following table shows how the transfer addresses are specified based on the setting of the MOD field. Here a circular buffer is created where the address wraps to the original value while the 28 upper address bits (0x1234567*x*) retain their original value. In this example the source address is set to 0x12345670, the offset is set to 4 bytes, and the MOD field is set to 4, allowing for a 2^4 byte (16-byte) size queue.

Table 6-7. Modulo example

Transfer number	Address
1	0x12345670
2	0x12345674
3	0x12345678
4	0x1234567C

Table continues on the next page...

Table 6-7. Modulo example (continued)

Transfer number	Address
5	0x12345670
6	0x12345674

6.4.5 Monitoring transfer descriptor status

This section discusses how to monitor eDMA status.

6.4.5.1 Testing for minor loop completion

There are two methods to test for minor loop completion when using software-initiated service requests. The first is to read the TCD*n*_CITER field and test for a change. Another method may be extracted from the sequence shown below. The second method is to test TCD*n*_CSR[START] and TCD*n*_CSR[ACTIVE]. The minor-loop-complete condition is indicated by both bits reading zero after TCD*n*_CSR[START] was set. Polling the TCD*n*_CSR[ACTIVE] bit may be inconclusive, because the active status may be missed if the channel execution is short in duration.

The TCD status bits execute the following sequence for a software activated channel:

Stage	TCD*n*_CSR bits			State
	START	ACTIVE	DONE	
1	1	0	0	Channel service request via software
2	0	1	0	Channel is executing
3a	0	0	0	Channel has completed the minor loop and is idle
3b	0	0	1	Channel has completed the major loop and is idle

The best method to test for minor-loop completion when using service requests initiated by hardware, that is, peripherals, is to read the TCD*n*_CITER field and test for a change. The hardware request and acknowledge handshake signals are not visible in the programmer's model.

The TCD status bits execute the following sequence for a hardware-activated channel:

Stage	TCD*n*_CSR bits			State
	START	ACTIVE	DONE	
1	0	0	0	Channel service request via hardware (peripheral request asserted)
2	0	1	0	Channel is executing
3a	0	0	0	Channel has completed the minor loop and is idle
3b	0	0	1	Channel has completed the major loop and is idle

For both activation types, the major-loop-complete status is explicitly indicated via the TCD*n*_CSR[DONE] bit.

The TCD*n*_CSR[START] bit is cleared automatically when the channel begins execution regardless of how the channel activates.

6.4.5.2 Reading the transfer descriptors of active channels

The eDMA reads back the true TCD*n*_SADDR, TCD*n*_DADDR, and TCD*n*_NBYTES values if read when a channel executes. The true values of SADDR, DADDR, and NBYTES are the values the eDMA engine currently uses in its internal register file and not the values in the TCD local memory for that channel. The addresses, SADDR and DADDR, and NBYTES, which decrement to zero as the transfer progresses, can give an indication of the progress of the transfer. All other values are read back from the TCD local memory.

6.4.5.3 Checking channel preemption status

Preemption is available only when fixed arbitration is selected for both group and channel arbitration modes. A preemptive situation is one in which a preempt-enabled channel runs and a higher priority request becomes active. When the eDMA engine is not operating in fixed group, fixed channel arbitration mode, determination of the actively running relative priority outstanding requests become undefined. Channel and/or group priorities are treated as equal, that is, constantly rotating, when Round-Robin Arbitration mode is selected.

The TCD*n*_CSR[ACTIVE] bit for the preempted channel remains asserted throughout the preemption. The preempted channel is temporarily suspended while the preempting channel executes one major loop iteration. If two TCD*n*_CSR[ACTIVE] bits are set simultaneously in the global TCD map, a higher priority channel is actively preempting a lower priority channel.

6.4.6 Channel linking

Channel linking (or chaining) is a mechanism where one channel sets the TCD*n*_CSR[START] bit of another channel (or itself), thus initiating a service request for that channel. When properly enabled, the EDMA engine automatically performs this operation at the major or minor loop completion.

The minor loop channel linking occurs at the completion of the minor loop (or one iteration of the major loop). The TCD*n*_CITER[ELINK] field determines whether a minor loop link is requested. When enabled, the channel link is made after each iteration of the major loop except for the last. When the major loop is exhausted, only the major loop channel link fields are used to determine if a channel link should be made. For example, the initial fields of:

```
TCDn_CITER[ELINK] = 1
TCDn_CITER[LINKCH] = 0xC
TCDn_CITER[CITER] value = 0x4
TCDn_CSR[MAJOR_ELINK] = 1
TCDn_CSR[MAJOR_LINKCH] = 0x3
```

executes as:

1. Minor loop done → set TCD2_CSR[START] bit.
2. Minor loop done → set TCD2_CSR[START] bit.
3. Minor loop done → set TCD2_CSR[START] bit.
4. Minor loop done, major loop done → set TCD3_CSR[START] bit.

When minor loop linking is enabled (TCD*n*_CITER[ELINK] = 1), the TCD*n*_CITER[CITER] field uses a nine-bit vector to form the current iteration count. When minor loop linking is disabled (TCD*n*_CITER[ELINK] = 0), the TCD*n*_CITER[CITER] field uses a 15-bit vector to form the current iteration count. The bits associated with the TCD*n*_CITER[LINKCH] field are concatenated onto the CITER value to increase the range of the CITER.

Note

The TCD*n*_CITER[ELINK] bit and the TCD*n*_BITER[ELINK] bit must be equal or a configuration error is reported. The CITER and BITER vector widths must be equal to calculate the major loop, half-way done interrupt point.

The following table summarizes how a DMA channel can link to another DMA channel, that is, use another channel's TCD, at the end of a loop.

Table 6-8. Channel linking parameters

Desired link behavior	TCD control field name	Description
Link at end of minor loop	CITER[ELINK]	Enable channel-to-channel linking on minor loop completion (current iteration)
	CITER[LINKCH]	Link channel number when linking at end of minor loop (current iteration)
Link at end of major loop	CSR[MAJOR_ELINK]	Enable channel-to-channel linking on major loop completion
	CSR[MAJOR_LINKCH]	Link channel number when linking at end of major loop

6.4.7 Dynamic programming

This section provides recommended methods to change the programming model during channel execution.

6.4.7.1 Dynamically changing the channel priority

The following two options are recommended for dynamically changing channel priority levels:

1. Switch to Round-Robin Channel Arbitration mode, change the channel priorities, then switch back to Fixed Arbitration mode
2. Disable all the channels, change the channel priorities, then enable the appropriate channels.

6.4.7.2 Dynamic channel linking

Dynamic channel linking is the process of setting TCDn_CSR[MAJORELINK] during channel execution (see the diagram in TCD structure). This field is read from the TCD local memory at the end of channel execution, thus enabling you to enable the feature during channel execution.

Because you can change the configuration during execution, a coherency model is needed. Consider the scenario where you attempt to execute a dynamic channel link by enabling TCDn_CSR[MAJORELINK] at the same time the eDMA engine is retiring the channel. TCDn_CSR[MAJORELINK] would be set in the programmer's model, but it would be unclear whether the actual link was made before the channel retired.

The following coherency model is recommended when executing a dynamic channel link request.

1. Write one to TCDn_CSR[MAJORELINK].
2. Read back TCDn_CSR[MAJORELINK].
3. Test the TCDn_CSR[MAJORELINK] request status:
   - If TCDn_CSR[MAJORELINK] = 1, the dynamic link attempt was successful.
   - If TCDn_CSR[MAJORELINK] = 0, the attempted dynamic link did not succeed (the channel was already retiring).

For this request, the TCD local memory controller forces TCDn_CSR[MAJORELINK] to zero on any writes to a channel's TCD.word7 after that channel's TCD.done bit is set, indicating the major loop is complete.

NOTE

You must clear TCDn_CSR[DONE] before writing TCDn_CSR[MAJORELINK]. The eDMA engine automatically clears TCDn_CSR[DONE] after a channel begins execution.

6.4.7.3 Dynamic scatter/gather

Scatter/gather is the process of automatically loading a new TCD into a channel. It enables a DMA channel to use multiple TCDs; this enables a DMA channel to scatter the DMA data to multiple destinations or gather it from multiple sources. When scatter/gather is enabled and the channel has finished its major loop, a new TCD is fetched from system memory and loaded into that channel's descriptor location in eDMA programmer's model, thus replacing the current descriptor.

Because you are able to change the configuration during execution, a coherency model is needed. Consider the scenario where you attempt to execute a dynamic scatter/gather operation by enabling the TCDn_CSR[ESG] bit at the same time the eDMA engine is retiring the channel. The ESG bit would be set in the programmer's model, but it would be unclear whether the actual scatter/gather request was honored before the channel retired.

Two methods for this coherency model are shown in the following subsections. Method 1 has the advantage of reading the MAJORLINKCH field and the ESG bit with a single read. For both dynamic channel linking and scatter/gather requests, the TCD local memory controller forces the TCD MAJOR[ELINK] and ESG bits to zero on any writes to a channel's TCD word 7 if that channel's TCD[DONE] bit is set, indicating the major loop is complete.

NOTE

The user must clear the TCDn_CSR[DONE] bit before writing the MAJORELINK or ESG bits. The TCDn_CSR[DONE] bit is cleared automatically by the eDMA engine after a channel begins execution.

6.4.7.3.1 Method 1 (channel not using major loop channel linking)

For a channel not using major loop channel linking, the coherency model described here may be used for a dynamic scatter/gather request.

When TCDn_CSR[MAJORELINK] is zero, TCDn_CSR[MAJORLINKCH] is not used by the eDMA. In this case, MAJORLINKCH may be used for other purposes. This method uses the MAJORLINKCH field as a TCD identification (ID).

1. When the descriptors are built, write a unique TCD ID in TCDn_CSR[MAJORLINKCH] for each TCD associated with a channel using dynamic scatter/gather.
2. Write one to TCDn_CSR[DREQ].

   Should a dynamic scatter/gather attempt fail, setting the DREQ bit prevents a future hardware activation of the channel. This stops the channel from executing with a destination address (DADDR) that was calculated using a scatter/gather address (written in the next step) instead of a DLAST_SGA final offset value.

3. Write the TCDn_DLASTSGA register with the scatter/gather address.
4. Write one to TCDn_CSR[ESG].
5. Read back the 16-bit TCD control/status field.
6. Test the ESG request status and MAJORLINKCH value in the TCDn_CSR register:

   If ESG = 1, the dynamic link attempt was successful.

   If ESG = 0 and MAJORLINKCH (ID) did not change, the attempted dynamic link did not succeed (the channel was already retiring).

   If ESG = 0 and MAJORLINKCH (ID) changed, the dynamic link attempt was successful (the new TCD's ESG value cleared the ESG bit).

6.4.7.3.2 Method 2 (channel using major loop channel linking)

For a channel using major loop channel linking, the coherency model described here may be used for a dynamic scatter/gather request. This method uses the TCD[DLAST_SGA] field as a TCD identification (ID).

1. Write one to TCDn_CSR[DREQ].

Should a dynamic scatter/gather attempt fail, setting TCDn_CSR[DREQ] prevents a future hardware activation of the channel. This stops the channel from executing with a destination address (DADDR) that was calculated using a scatter/gather address (written in the next step) instead of a DLAST_SGA final offset value.

2. Write the TCDn_DLAST_SGA register with the scatter/gather address.
3. Write one to TCDn_CSR[ESG].
4. Read back TCDn_CSR[ESG].
5. Test the ESG request status:

   If ESG = 1, the dynamic link attempt was successful.

   If ESG = 0, read the 32-bit TCDn_DLAST_SGA field.

   If ESG = 0 and TCDn_DLAST_SGA did not change, the attempted dynamic link did not succeed (the channel was already retiring).

   If ESG = 0 and TCDn_DLAST_SGA changed, the dynamic link attempt was successful (the new TCD's ESG value cleared the ESG bit).

6.4.8 Suspend/resume a DMA channel with active hardware service requests

The DMA enables you to move data from memory or peripheral registers to another location in memory or peripheral registers without CPU interaction. After the DMA and peripherals have been configured and are active, it is rare to suspend a peripheral's service request dynamically. In this scenario, there are certain restrictions to disabling a DMA hardware service request. For coherency, a specific procedure must be followed. This section provides guidance on how to coherently suspend and resume a Direct Memory Access (DMA) channel when the DMA is triggered by a slave module such as the Serial Peripheral Interface (SPI), ADC, or other module.

6.4.8.1 Suspend an active DMA channel

To suspend an active DMA channel:

1. Stop the DMA service request at the peripheral first. Confirm it has been disabled by reading back the appropriate register in the peripheral.
2. Check the DMA's Hardware Request Status register (DMA_HRSn) to ensure there is no service request to the DMA channel being suspended. Then disable the hardware service request by clearing the ERQ bit on appropriate DMA channel.

6.4.8.2 Resume a DMA channel

To resume a DMA channel:

1. Enable the DMA service request on the appropriate channel by setting the relevant ERQ bit.
2. Enable the DMA service request at the peripheral.

For example, assume the SPI is set as a master for transmitting data via a DMA service request when the SPI_TXFIFO has an empty slot. The DMA transfers the next command and data to the TXFIFO upon the request. You must suspend the DMA/SPI transfer loop and perform the following steps:

1. Disable the DMA service request at the source by writing zero to SPI_RSER[TFFF_RE]. Confirm that SPI_RSER[TFFF_RE] is zero.
2. Ensure there is no DMA service request from the SPI by verifying that DMA_HRS[HRS*n*] is zero for the appropriate channel. If no service request is present, disable the DMA channel by clearing the channel's ERQ bit. If a service request is present, wait until the request has been processed and the HRS bit reads zero.

6.5 Memory map/register definition

The eDMA programming model consists of registers that provide:

- Control and status functions
- Channel configuration functions
- TCD definition functions

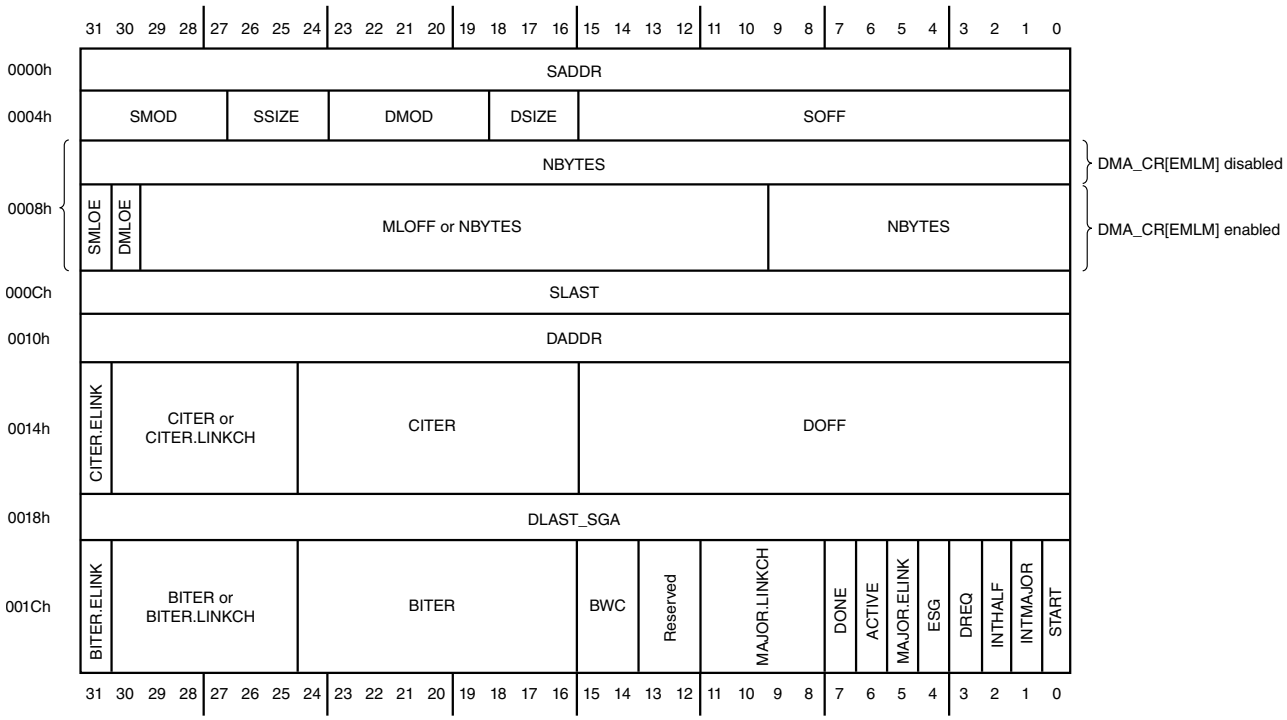
6.5.1 TCD memory

Each channel requires a 32-byte TCD to define the desired data movement operation. The channel descriptors are in local memory in sequential order: channel 0, channel 1, ... channel 31. Each TCD*n* definition comprises 11 registers of 16 or 32 bits.

6.5.2 TCD initialization

Before activating a channel, you must initialize its TCD with the appropriate transfer profile.

6.5.3 TCD structure



6.5.4 Reserved memory and fields

- Reading reserved fields in a register returns the value of zero.
- The eDMA ignores writes to reserved bits in a register.
- Reading or writing a reserved memory location generates a bus error.

6.5.5 DMA register descriptions

6.5.5.1 DMA memory map

DMA base address: 400E_8000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Control (CR)	32	RW	Table 6-8

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
4h	Error Status (ES)	32	RO	0000_0000h
Ch	Enable Request (ERQ)	32	RW	0000_0000h
14h	Enable Error Interrupt (EEI)	32	RW	0000_0000h
18h	Clear Enable Error Interrupt (CEEI)	8	WORZ	00h
19h	Set Enable Error Interrupt (SEEI)	8	WORZ	00h
1Ah	Clear Enable Request (CERQ)	8	WORZ	00h
1Bh	Set Enable Request (SERQ)	8	WORZ	00h
1Ch	Clear DONE Status Bit (CDNE)	8	WORZ	00h
1Dh	Set START Bit (SSRT)	8	WORZ	00h
1Eh	Clear Error (CERR)	8	WORZ	00h
1Fh	Clear Interrupt Request (CINT)	8	WORZ	00h
24h	Interrupt Request (INT)	32	W1C	0000_0000h
2Ch	Error (ERR)	32	W1C	0000_0000h
34h	Hardware Request Status (HRS)	32	RO	0000_0000h
44h	Enable Asynchronous Request in Stop (EARS)	32	RW	0000_0000h
100h	Channel Priority (DCHPRI3)	8	RW	03h
101h	Channel Priority (DCHPRI2)	8	RW	02h
102h	Channel Priority (DCHPRI1)	8	RW	01h
103h	Channel Priority (DCHPRI0)	8	RW	00h
104h	Channel Priority (DCHPRI7)	8	RW	07h
105h	Channel Priority (DCHPRI6)	8	RW	06h
106h	Channel Priority (DCHPRI5)	8	RW	05h
107h	Channel Priority (DCHPRI4)	8	RW	04h
108h	Channel Priority (DCHPRI11)	8	RW	0Bh
109h	Channel Priority (DCHPRI10)	8	RW	0Ah
10Ah	Channel Priority (DCHPRI9)	8	RW	09h
10Bh	Channel Priority (DCHPRI8)	8	RW	08h
10Ch	Channel Priority (DCHPRI15)	8	RW	0Fh
10Dh	Channel Priority (DCHPRI14)	8	RW	0Eh
10Eh	Channel Priority (DCHPRI13)	8	RW	0Dh
10Fh	Channel Priority (DCHPRI12)	8	RW	0Ch
110h	Channel Priority (DCHPRI19)	8	RW	13h
111h	Channel Priority (DCHPRI18)	8	RW	12h
112h	Channel Priority (DCHPRI17)	8	RW	11h
113h	Channel Priority (DCHPRI16)	8	RW	10h
114h	Channel Priority (DCHPRI23)	8	RW	17h
115h	Channel Priority (DCHPRI22)	8	RW	16h
116h	Channel Priority (DCHPRI21)	8	RW	15h
117h	Channel Priority (DCHPRI20)	8	RW	14h
118h	Channel Priority (DCHPRI27)	8	RW	1Bh

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
119h	Channel Priority (DCHPRI26)	8	RW	1Ah
11Ah	Channel Priority (DCHPRI25)	8	RW	19h
11Bh	Channel Priority (DCHPRI24)	8	RW	18h
11Ch	Channel Priority (DCHPRI31)	8	RW	1Fh
11Dh	Channel Priority (DCHPRI30)	8	RW	1Eh
11Eh	Channel Priority (DCHPRI29)	8	RW	1Dh
11Fh	Channel Priority (DCHPRI28)	8	RW	1Ch
1000h - 13E0h	TCD Source Address (TCD0_SADDR - TCD31_SADDR)	32	RW	Table 6-8
1004h - 13E4h	TCD Signed Source Address Offset (TCD0_SOFF - TCD31_SOFF)	16	RW	Table 6-8
1006h - 13E6h	TCD Transfer Attributes (TCD0_ATTR - TCD31_ATTR)	16	RW	Table 6-8
1008h - 13E8h	TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD0_NBYTES_MLNO - TCD31_NBYTES_MLNO)	32	RW	Table 6-8
1008h - 13E8h	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD0_NBYTES_MLOFFNO - TCD31_NBYTES_MLOFFNO)	32	RW	Table 6-8
1008h - 13E8h	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD0_NBYTES_MLOFFYES - TCD31_NBYTES_MLOFFYES)	32	RW	Table 6-8
100Ch - 13ECh	TCD Last Source Address Adjustment (TCD0_SLAST - TCD31_SLAST)	32	RW	Table 6-8
1010h - 13F0h	TCD Destination Address (TCD0_DADDR - TCD31_DADDR)	32	RW	Table 6-8
1014h - 13F4h	TCD Signed Destination Address Offset (TCD0_DOFF - TCD31_DOFF)	16	RW	Table 6-8
1016h - 13F6h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD0_CITER_ELINKNO - TCD31_CITER_ELINKNO)	16	RW	Table 6-8
1016h - 13F6h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD0_CITER_ELINKYES - TCD31_CITER_ELINKYES)	16	RW	Table 6-8
1018h - 13F8h	TCD Last Destination Address Adjustment/Scatter Gather Address (TCD0_DLASTSGA - TCD31_DLASTSGA)	32	RW	Table 6-8
101Ch - 13FCh	TCD Control and Status (TCD0_CSR - TCD31_CSR)	16	RW	Table 6-8
101Eh - 13FEh	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD0_BITER_ELINKNO - TCD31_BITER_ELINKNO)	16	RW	Table 6-8
101Eh - 13FEh	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD0_BITER_ELINKYES - TCD31_BITER_ELINKYES)	16	RW	Table 6-8

6.5.5.2 Control (CR)

6.5.5.2.1 Offset

Register	Offset
CR	0h

6.5.5.2.2 Function

This register defines the basic operating configuration of the eDMA module. eDMA arbitrates channel service requests in two groups of 16 channels each:

- Group 1 contains channels 31-16
- Group 0 contains channels 15-0

You can configure arbitration within a group to use either a fixed-priority or a round-robin scheme. For fixed-priority arbitration, eDMA selects and executes the highest-priority channel that requests service. The channel priority registers assign the priorities (see the Channel Priority (DCHPRI0 - DCHPRI31) registers). For round-robin arbitration, the eDMA engine ignores channel priorities and cycles through channels within each group from high to low channel number without regard to priority.

NOTE

For correct operation, you must write to this register only when the eDMA channels are inactive—that is, when TCD*n*_CSR[ACTIVE] = 0.

The group priorities operate in a similar fashion. In group fixed priority arbitration mode, channel service requests in the highest priority group are executed first, where priority level 1 is the highest and priority level 0 is the lowest. The group priorities are assigned in the GRP*n*PRI fields of the Control register (CR). All group priorities must have unique values prior to any channel service requests occurring; otherwise, a configuration error is reported. For group round robin arbitration, eDMA ignores the group priorities and the groups are cycled through (from high to low group number) without regard to priority.

Minor loop offsets are address-offset values to be added to the final source address (TCDn_SADDR) or destination address (TCDn_DADDR) when the minor loop completes. When you enable minor loop offsets, eDMA adds the minor loop offset (MLOFF) value to the final source address (TCDn_SADDR), to the final destination address (TCDn_DADDR), or to both, before it writes the addresses back into the TCD. If the major loop is complete, eDMA ignores the minor loop offset, and uses the major loop address offsets (TCDn_SLAST and TCDn_DLAST_SGA) to compute the next TCDn_SADDR and TCDn_DADDR values.

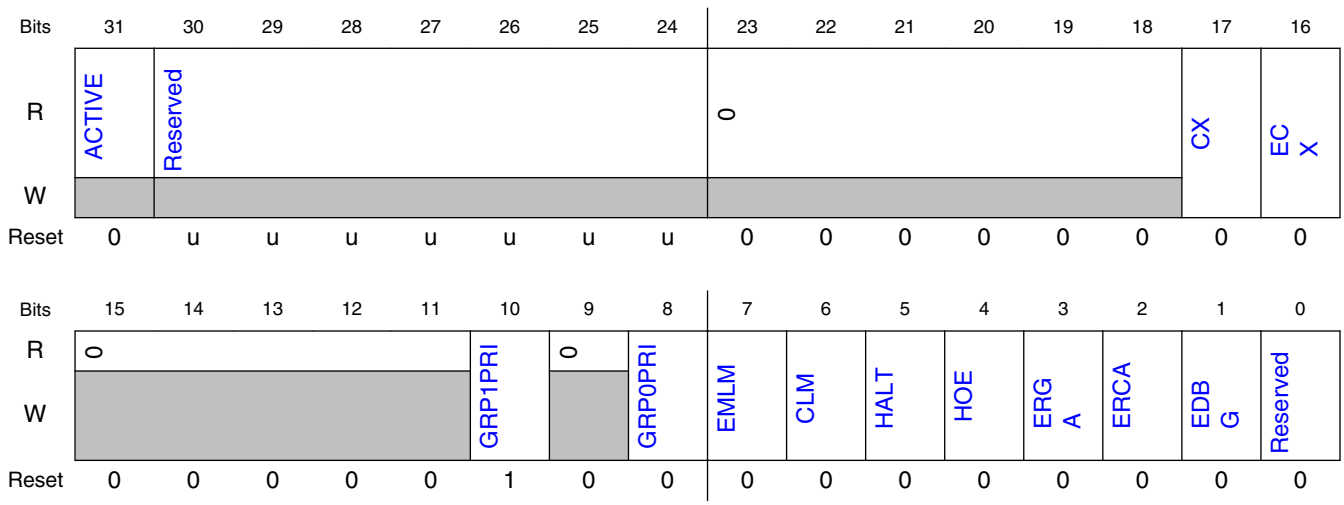
Enabling minor loop mapping (EMLM = 1) redefines TCDn word2. eDMA uses a portion of TCDn word2 for multiple fields:

- A source enable field (SMLOE) to specify the minor loop offset is to be applied to the source address (TCDn_SADDR) when the minor loop completes
- A destination enable field (DMLOE) to specify the minor loop offset to be applied to the destination address (TCDn_DADDR) when the minor loop completes
- The sign extended minor loop offset value (MLOFF).

eDMA uses the same offset value (MLOFF) for both source and destination minor loop offsets. When you enable either minor loop offset (SMLOE = 1 or DMLOE = 1), the NBYTES field reduces in size to 10 bits. When you disable both minor loop offsets (SMLOE = 0 and and DMLOE = 0), the NBYTES field is a 30-bit vector.

When you disable minor loop mapping (EMLM = 0), the NBYTES field contains all 32 bits of TCDn word2.

6.5.5.2.3 Diagram



6.5.5.2.4 Fields

Field	Function
31 ACTIVE	eDMA Active Status 0b - eDMA is idle 1b - eDMA is executing a channel
30-24 —	Reserved
23-18 —	Reserved
17 CX	Cancel Transfer When you write 1 to this field, the following actions take place:

Table continues on the next page...

Field	Function
	• Stop the executing channel • Force the minor loop to finish. The cancellation takes effect after the last write of the current read/write sequence. This field is automatically written with 0 after the cancellation completes. The cancellation retires the channel normally as if the minor loop completed. 0b - Normal operation 1b - Cancel the remaining data transfer
16 ECX	Error Cancel Transfer When you write a 1 to this field, the following actions take place: • Stop the executing channel • Force the minor loop to finish. The cancellation takes effect after the last write of the current read/write sequence. This field is automatically reset to 0 after the cancellation completes. In addition to cancelling the transfer, eDMA: • Treats the cancel as an error condition • Updates the Error Status register (DMAx_ES) • Optionally generates an error interrupt. 0b - Normal operation 1b - Cancel the remaining data transfer
15-11 —	Reserved
10 GRP1PRI	Channel Group 1 Priority Group 1 priority level when fixed priority group arbitration is enabled.
9 —	Reserved
8 GRP0PRI	Channel Group 0 Priority Group 0 priority level when fixed priority group arbitration is enabled.
7 EMLM	Enable Minor Loop Mapping When the value of this field is 0, TCDn.word2 is a 32-bit NBYTES field. When the value of this field is 1, TCDn.word2 includes: • Individual enable fields • An offset field • The NBYTES field. The individual enable fields allow the minor loop offset to be applied to the source address, the destination address, or both. The NBYTES field reduces in size when either offset is enabled. 0b - Disabled 1b - Enabled
6 CLM	Continuous Link Mode When the value of this field is 0, a minor loop channel link made to itself goes through channel arbitration before being activated again. When the value of this field is 1, a minor loop channel link made to itself does not go through channel arbitration before being activated again. When the minor loop completes, the channel activates again if that channel has a minor loop channel link enabled and the link channel is itself. This effectively applies the minor loop offsets and restarts the next minor loop. **NOTE:** Do not use continuous link mode with a channel linking to itself if there is only one minor loop iteration per service request, for example, if the channel's NBYTES value is the same as either the source or destination size. The same data transfer profile can be achieved by simply increasing the NBYTES value, which provides more efficient, faster processing. 0b - Continuous link mode is off

Table continues on the next page...

Field	Function
	1b - Continuous link mode is on
5 HALT	Halt eDMA Operations When this field is 1 the following actions take place: • eDMA stalls the start of any new channels • Executing channels are allowed to complete. When you write 0 to this field, channel execution resumes. 0b - Normal operation 1b - eDMA operations halted
4 HOE	Halt On Error When this field is 1, any error causes the eDMA engine to write 1 to the HALT field. Subsequently, the eDMA engine ignores all service requests until you write 0 to the HALT field. 0b - Normal operation 1b - Error causes HALT field to be automatically set to 1
3 ERGA	Enable Round Robin Group Arbitration When you write 1 to this field, eDMA uses round robin arbitration for selection among the groups. Otherwise, eDMA uses fixed priority arbitration. 0b - Fixed priority arbitration 1b - Round robin arbitration
2 ERCA	Enable Round Robin Channel Arbitration When you write 1 to this field, eDMA uses round robin arbitration for channel selection. Otherwise, eDMA uses fixed priority arbitration for channel selection. 0b - Fixed priority arbitration within each group 1b - Round robin arbitration within each group
1 EDBG	Enable Debug When this field is 0 and the chip enters Debug mode, eDMA continues operation. When this field is 1, entry of the chip into Debug mode causes the eDMA to stall the start of a new channel. Executing channels are allowed to complete. Channel execution resumes when the chip exits Debug mode or you write 0 to this field. 0b - When the chip is in Debug mode, the eDMA continues to operate. 1b - When the chip is in debug mode, the DMA stalls the start of a new channel. Executing channels are allowed to complete.
0 —	Reserved

6.5.5.3 Error Status (ES)

6.5.5.3.1 Offset

Register	Offset
ES	4h

6.5.5.3.2 Function

The ES register provides information concerning the most-recently recorded channel error. Channel errors can be caused by:

- A configuration error, that is:
  - An illegal setting in the transfer-control descriptor
  - An illegal priority register setting in fixed arbitration
- An error termination to a bus master read or write cycle
- A cancel transfer with error field that is 1 when a transfer is canceled via the corresponding cancel transfer control field

See Fault reporting and handling for more details.

6.5.5.3.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	VLD	0														ECX
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	GPE	CPE	0	ERRCHN					SAE	SOE	DAE	DOE	NCE	SGE	SBE	DBE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6.5.5.3.4 Fields

Field	Function
31 VLD	Logical OR of all ERR status fields 0b - No ERR fields are 1 1b - At least one ERR field has a value of 1, indicating a valid error exists that has not been cleared
30-17 —	Reserved
16 ECX	Transfer Canceled 0b - No canceled transfers 1b - The most-recently recorded entry was a canceled transfer initiated by the error cancel transfer field
15 GPE	Group Priority Error 0b - No group priority error. 1b - The most-recently recorded error was a configuration error among the group priorities. All group priorities are not unique.
14 CPE	Channel Priority Error 0b - No channel priority error. 1b - The most-recently recorded error was a configuration error in the channel priorities within a group. Channel priorities within a group are not unique.

Table continues on the next page...

Field	Function
13 —	Reserved
12-8 ERRCHN	Error Channel Number or Canceled Channel Number The channel number of the most-recently recorded error, excluding GPE and CPE errors or most-recently recorded error canceled transfer.
7 SAE	Source Address Error 0b - No source address configuration error. 1b - The most-recently recorded error was a configuration error detected in the TCDn_SADDR field. TCDn_SADDR is inconsistent with TCDn_ATTR[SSIZE].
6 SOE	Source Offset Error 0b - No source offset configuration error. 1b - The most-recently recorded error was a configuration error detected in the TCDn_SOFF field. TCDn_SOFF is inconsistent with TCDn_ATTR[SSIZE].
5 DAE	Destination Address Error 0b - No destination address configuration error. 1b - The most-recently recorded error was a configuration error detected in the TCDn_DADDR field. TCDn_DADDR is inconsistent with TCDn_ATTR[DSIZE].
4 DOE	Destination Offset Error 0b - No destination offset configuration error. 1b - The most-recently recorded error was a configuration error detected in the TCDn_DOFF field. TCDn_DOFF is inconsistent with TCDn_ATTR[DSIZE].
3 NCE	NBYTES/CITER Configuration Error 0b - No NBYTES/CITER configuration error. 1b - The most-recently recorded error was a configuration error detected in the TCDn_NBYTES or TCDn_CITER fields. TCDn_NBYTES is not a multiple of TCDn_ATTR[SSIZE] and TCDn_ATTR[DSIZE], or TCDn_CITER[CITER] = 0, or TCDn_CITER[ELINK] is not equal to TCDn_BITER[ELINK].
2 SGE	Scatter/Gather Configuration Error When 1, this field indicates the most-recently recorded error was a configuration error detected in the TCDn_DLASTSGA field. eDMA checks This field at the beginning of a scatter/gather operation after major loop completion if TCDn_CSR[ESG] is enabled. TCDn_DLASTSGA is not on a 32-byte boundary. 0b - No scatter/gather configuration error. 1b - The most-recently recorded error was a configuration error detected in the TCDn_DLASTSGA field.
1 SBE	Source Bus Error 0b - No source bus error. 1b - The most-recently recorded error was a bus error on a source read.
0 DBE	Destination Bus Error 0b - No destination bus error. 1b - The most-recently recorded error was a bus error on a destination write.

6.5.5.4 Enable Request (ERQ)

6.5.5.4.1 Offset

Register	Offset
ERQ	Ch

6.5.5.4.2 Function

The ERQ register provides a bit map for the 32 channels to enable the request signal for each channel. The state of any given channel enable is directly affected by writes to this register; it is also affected by writes to the SERQ and CERQ registers. These registers are provided so the request enable for a single channel can easily be modified without needing to perform a read-modify-write sequence to this register.

DMA request input signals and this enable request field must be set to 1 before a channel's hardware service request is accepted. The state of the DMA enable request field does not affect a channel service request made explicitly through software or a linked channel request.

NOTE

Disable a channel's hardware service request at the source before writing 0 to the channel's ERQ field.

6.5.5.4.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ERQ31	ERQ30	ERQ29	ERQ28	ERQ27	ERQ26	ERQ25	ERQ24	ERQ23	ERQ22	ERQ21	ERQ20	ERQ19	ERQ18	ERQ17	ERQ16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ERQ15	ERQ14	ERQ13	ERQ12	ERQ11	ERQ10	ERQ9	ERQ8	ERQ7	ERQ6	ERQ5	ERQ4	ERQ3	ERQ2	ERQ1	ERQ0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6.5.5.4.4 Fields

Field	Function
31 ERQ31	Enable DMA Request 31 0b - The DMA request signal for channel 31 is disabled 1b - The DMA request signal for channel 31 is enabled

Table continues on the next page...

Field	Function
30 ERQ30	Enable DMA Request 30 0b - The DMA request signal for channel 30 is disabled 1b - The DMA request signal for channel 30 is enabled
29 ERQ29	Enable DMA Request 29 0b - The DMA request signal for channel 29 is disabled 1b - The DMA request signal for channel 29 is enabled
28 ERQ28	Enable DMA Request 28 0b - The DMA request signal for channel 28 is disabled 1b - The DMA request signal for channel 28 is enabled
27 ERQ27	Enable DMA Request 27 0b - The DMA request signal for channel 27 is disabled 1b - The DMA request signal for channel 27 is enabled
26 ERQ26	Enable DMA Request 26 0b - The DMA request signal for channel 26 is disabled 1b - The DMA request signal for channel 26 is enabled
25 ERQ25	Enable DMA Request 25 0b - The DMA request signal for channel 25 is disabled 1b - The DMA request signal for channel 25 is enabled
24 ERQ24	Enable DMA Request 24 0b - The DMA request signal for channel 24 is disabled 1b - The DMA request signal for channel 24 is enabled
23 ERQ23	Enable DMA Request 23 0b - The DMA request signal for channel 23 is disabled 1b - The DMA request signal for channel 23 is enabled
22 ERQ22	Enable DMA Request 22 0b - The DMA request signal for channel 22 is disabled 1b - The DMA request signal for channel 22 is enabled
21 ERQ21	Enable DMA Request 21 0b - The DMA request signal for channel 21 is disabled 1b - The DMA request signal for channel 21 is enabled
20 ERQ20	Enable DMA Request 20 0b - The DMA request signal for channel 20 is disabled 1b - The DMA request signal for channel 20 is enabled
19 ERQ19	Enable DMA Request 19 0b - The DMA request signal for channel 19 is disabled 1b - The DMA request signal for channel 19 is enabled
18 ERQ18	Enable DMA Request 18 0b - The DMA request signal for channel 18 is disabled 1b - The DMA request signal for channel 18 is enabled
17 ERQ17	Enable DMA Request 17 0b - The DMA request signal for channel 17 is disabled 1b - The DMA request signal for channel 17 is enabled
16 ERQ16	Enable DMA Request 16 0b - The DMA request signal for channel 16 is disabled 1b - The DMA request signal for channel 16 is enabled
15 ERQ15	Enable DMA Request 15 0b - The DMA request signal for channel 15 is disabled 1b - The DMA request signal for channel 15 is enabled
14 ERQ14	Enable DMA Request 14 0b - The DMA request signal for channel 14 is disabled

Table continues on the next page...

Field	Function
	1b - The DMA request signal for channel 14 is enabled
13 ERQ13	Enable DMA Request 13 0b - The DMA request signal for channel 13 is disabled 1b - The DMA request signal for channel 13 is enabled
12 ERQ12	Enable DMA Request 12 0b - The DMA request signal for channel 12 is disabled 1b - The DMA request signal for channel 12 is enabled
11 ERQ11	Enable DMA Request 11 0b - The DMA request signal for channel 11 is disabled 1b - The DMA request signal for channel 11 is enabled
10 ERQ10	Enable DMA Request 10 0b - The DMA request signal for channel 10 is disabled 1b - The DMA request signal for channel 10 is enabled
9 ERQ9	Enable DMA Request 9 0b - The DMA request signal for channel 9 is disabled 1b - The DMA request signal for channel 9 is enabled
8 ERQ8	Enable DMA Request 8 0b - The DMA request signal for channel 8 is disabled 1b - The DMA request signal for channel 8 is enabled
7 ERQ7	Enable DMA Request 7 0b - The DMA request signal for channel 7 is disabled 1b - The DMA request signal for channel 7 is enabled
6 ERQ6	Enable DMA Request 6 0b - The DMA request signal for channel 6 is disabled 1b - The DMA request signal for channel 6 is enabled
5 ERQ5	Enable DMA Request 5 0b - The DMA request signal for channel 5 is disabled 1b - The DMA request signal for channel 5 is enabled
4 ERQ4	Enable DMA Request 4 0b - The DMA request signal for channel 4 is disabled 1b - The DMA request signal for channel 4 is enabled
3 ERQ3	Enable DMA Request 3 0b - The DMA request signal for channel 3 is disabled 1b - The DMA request signal for channel 3 is enabled
2 ERQ2	Enable DMA Request 2 0b - The DMA request signal for channel 2 is disabled 1b - The DMA request signal for channel 2 is enabled
1 ERQ1	Enable DMA Request 1 0b - The DMA request signal for channel 1 is disabled 1b - The DMA request signal for channel 1 is enabled
0 ERQ0	Enable DMA Request 0 0b - The DMA request signal for channel 0 is disabled 1b - The DMA request signal for channel 0 is enabled

6.5.5.5 Enable Error Interrupt (EEI)

6.5.5.5.1 Offset

Register	Offset
EEI	14h

6.5.5.5.2 Function

The EEI register provides a bit map for the 32 channels to enable the error interrupt signal for each channel. The state of any given channel's error interrupt enable is directly affected by writes to this register; it is also affected by writes to the SEEI and CEEI registers. These registers are provided so that the error interrupt enable for a single channel can easily be modified without the need to perform a read-modify-write sequence to the EEI register.

The DMA error indicator and the error interrupt enable field must be set to 1 before an error interrupt request for a given channel is sent to the interrupt controller.

6.5.5.5.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	EEI31	EEI30	EEI29	EEI28	EEI27	EEI26	EEI25	EEI24	EEI23	EEI22	EEI21	EEI20	EEI19	EEI18	EEI17	EEI16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	EEI15	EEI14	EEI13	EEI12	EEI11	EEI10	EEI 9	EEI 8	EEI 7	EEI 6	EEI 5	EEI 4	EEI 3	EEI 2	EEI 1	EEI 0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6.5.5.5.4 Fields

Field	Function
31 EEI31	Enable Error Interrupt 31 0b - An error on channel 31 does not generate an error interrupt 1b - An error on channel 31 generates an error interrupt request
30 EEI30	Enable Error Interrupt 30 0b - An error on channel 30 does not generate an error interrupt 1b - An error on channel 30 generates an error interrupt request
29 EEI29	Enable Error Interrupt 29 0b - An error on channel 29 does not generate an error interrupt 1b - An error on channel 29 generates an error interrupt request

Table continues on the next page...

Field	Function
28 EEI28	Enable Error Interrupt 28 0b - An error on channel 28 does not generate an error interrupt 1b - An error on channel 28 generates an error interrupt request
27 EEI27	Enable Error Interrupt 27 0b - An error on channel 27 does not generate an error interrupt 1b - An error on channel 27 generates an error interrupt request
26 EEI26	Enable Error Interrupt 26 0b - An error on channel 26 does not generate an error interrupt 1b - An error on channel 26 generates an error interrupt request
25 EEI25	Enable Error Interrupt 25 0b - An error on channel 25 does not generate an error interrupt 1b - An error on channel 25 generates an error interrupt request
24 EEI24	Enable Error Interrupt 24 0b - An error on channel 24 does not generate an error interrupt 1b - An error on channel 24 generates an error interrupt request
23 EEI23	Enable Error Interrupt 23 0b - An error on channel 23 does not generate an error interrupt 1b - An error on channel 23 generates an error interrupt request
22 EEI22	Enable Error Interrupt 22 0b - An error on channel 22 does not generate an error interrupt 1b - An error on channel 22 generates an error interrupt request
21 EEI21	Enable Error Interrupt 21 0b - An error on channel 21 does not generate an error interrupt 1b - An error on channel 21 generates an error interrupt request
20 EEI20	Enable Error Interrupt 20 0b - An error on channel 20 does not generate an error interrupt 1b - An error on channel 20 generates an error interrupt request
19 EEI19	Enable Error Interrupt 19 0b - An error on channel 19 does not generate an error interrupt 1b - An error on channel 19 generates an error interrupt request
18 EEI18	Enable Error Interrupt 18 0b - An error on channel 18 does not generate an error interrupt 1b - An error on channel 18 generates an error interrupt request
17 EEI17	Enable Error Interrupt 17 0b - An error on channel 17 does not generate an error interrupt 1b - An error on channel 17 generates an error interrupt request
16 EEI16	Enable Error Interrupt 16 0b - An error on channel 16 does not generate an error interrupt 1b - An error on channel 16 generates an error interrupt request
15 EEI15	Enable Error Interrupt 15 0b - An error on channel 15 does not generate an error interrupt 1b - An error on channel 15 generates an error interrupt request
14 EEI14	Enable Error Interrupt 14 0b - An error on channel 14 does not generate an error interrupt 1b - An error on channel 14 generates an error interrupt request
13 EEI13	Enable Error Interrupt 13 0b - An error on channel 13 does not generate an error interrupt 1b - An error on channel 13 generates an error interrupt request
12 EEI12	Enable Error Interrupt 12 0b - An error on channel 12 does not generate an error interrupt

Table continues on the next page...

Field	Function
	1b - An error on channel 12 generates an error interrupt request
11 EEI11	Enable Error Interrupt 11 0b - An error on channel 11 does not generate an error interrupt 1b - An error on channel 11 generates an error interrupt request
10 EEI10	Enable Error Interrupt 10 0b - An error on channel 10 does not generate an error interrupt 1b - An error on channel 10 generates an error interrupt request
9 EEI9	Enable Error Interrupt 9 0b - An error on channel 9 does not generate an error interrupt 1b - An error on channel 9 generates an error interrupt request
8 EEI8	Enable Error Interrupt 8 0b - An error on channel 8 does not generate an error interrupt 1b - An error on channel 8 generates an error interrupt request
7 EEI7	Enable Error Interrupt 7 0b - An error on channel 7 does not generate an error interrupt 1b - An error on channel 7 generates an error interrupt request
6 EEI6	Enable Error Interrupt 6 0b - An error on channel 6 does not generate an error interrupt 1b - An error on channel 6 generates an error interrupt request
5 EEI5	Enable Error Interrupt 5 0b - An error on channel 5 does not generate an error interrupt 1b - An error on channel 5 generates an error interrupt request
4 EEI4	Enable Error Interrupt 4 0b - An error on channel 4 does not generate an error interrupt 1b - An error on channel 4 generates an error interrupt request
3 EEI3	Enable Error Interrupt 3 0b - An error on channel 3 does not generate an error interrupt 1b - An error on channel 3 generates an error interrupt request
2 EEI2	Enable Error Interrupt 2 0b - An error on channel 2 does not generate an error interrupt 1b - An error on channel 2 generates an error interrupt request
1 EEI1	Enable Error Interrupt 1 0b - An error on channel 1 does not generate an error interrupt 1b - An error on channel 1 generates an error interrupt request
0 EEI0	Enable Error Interrupt 0 0b - An error on channel 0 does not generate an error interrupt 1b - An error on channel 0 generates an error interrupt request

6.5.5.6 Clear Enable Error Interrupt (CEEI)

6.5.5.6.1 Offset

Register	Offset
CEEI	18h

6.5.5.6.2 Function

The CEEI provides a simple memory-mapped mechanism to write 0 to a given field in the EEI register to disable the error interrupt for a given channel. The data value on a register write causes the corresponding field in the EEI register to be written to 0. Writing 1 to the CAEE field provides a global clear to 0 function, forcing the EEI contents to be written to 0, disabling all DMA request inputs.

If the NOP field is written with 1, the command is ignored. This enables you to write 1 to a single, byte-wide register with a 32-bit write that does not affect the other registers addressed in the write. In such a case the other three bytes of the word must all have their NOP field set to 1 so that these registers are not affected by the write.

Reads of this register return all zeroes.

6.5.5.6.3 Diagram

Bits	7	6	5	4	3	2	1	0
R	0	0		0				
W	NOP	CAEE	0	CEEI				
Reset	0	0	0	0	0	0	0	0

6.5.5.6.4 Fields

Field	Function
7 NOP	No Op Enable 0b - Normal operation 1b - No operation, ignore the other fields in this register
6 CAEE	Clear All Enable Error Interrupts 0b - Write 0 only to the EEI field specified in the CEEI field 1b - Write 0 to all fields in EEI
5 —	Reserved
4-0 CEEI	Clear Enable Error Interrupt Writes 0 to the corresponding field in EEI

6.5.5.7 Set Enable Error Interrupt (SEEI)

6.5.5.7.1 Offset

Register	Offset
SEEI	19h

6.5.5.7.2 Function

The SEEI register provides a simple memory-mapped mechanism to write 1 to a given field in the EEI register to enable the error interrupt for a given channel. The data value on a register write causes the corresponding field in the EEI to be written to 1. Writing 1 to the SAEE field provides a global set to 1 function, forcing the entire EEI register contents to be written with 1.

If the NOP field is 1, the command is ignored. This enables you to write 1 to a single, byte-wide register with a 32-bit write that does not affect the other registers addressed in the write. In such a case the other three bytes of the word must all have their NOP field set to 1 so that these registers are not affected by the write.

Reads of this register return all zeroes.

6.5.5.7.3 Diagram

Bits	7	6	5	4	3	2	1	0
R	0	0		0				
W	NOP	SAEE	0	SEEI				
Reset	0	0	0	0	0	0	0	0

6.5.5.7.4 Fields

Field	Function
7 NOP	No Op Enable 0b - Normal operation 1b - No operation, ignore the other fields in this register
6 SAEE	Set All Enable Error Interrupts 0b - Write 1 only to the EEI field specified in the SEEI field 1b - Writes 1 to all fields in EEI
5 —	Reserved
4-0 SEEI	Set Enable Error Interrupt Writes 1 to the corresponding field in EEI

6.5.5.8 Clear Enable Request (CERQ)

6.5.5.8.1 Offset

Register	Offset
CERQ	1Ah

6.5.5.8.2 Function

The CERQ provides a simple memory-mapped mechanism to write 0 to a given field in the ERQ register to disable the DMA request for a given channel. The data value on a register write causes the corresponding field in the ERQ register to be written with 0. Setting the CAER field provides a global clear to 0 function, forcing the entire contents of the ERQ register to be written with 0, disabling all DMA request inputs.

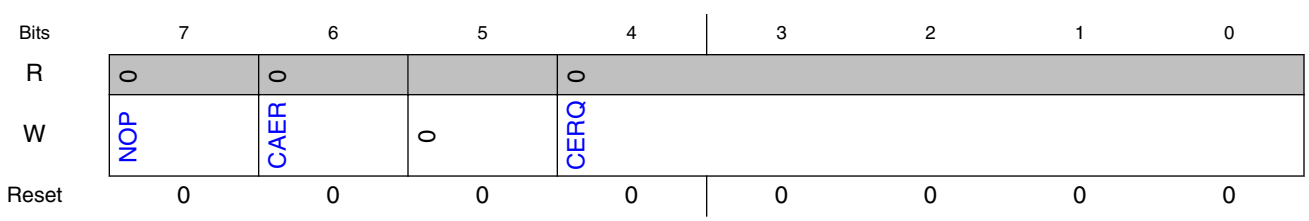
If the NOP field is 1, the command is ignored. This enables you to write 1 to a single, byte-wide register with a 32-bit write that does not affect the other registers addressed in the write. In such a case the other three bytes of the word must all have their NOP field written with 1 so that these registers are not affected by the write.

Reads of this register return all zeroes.

NOTE

Disable a channel's hardware service request at the source before writing 0 to the channel's ERQ field.

6.5.5.8.3 Diagram



6.5.5.8.4 Fields

Field	Function
7	No Op Enable 0b - Normal operation

Table continues on the next page...

Field	Function
NOP	1b - No operation, ignore the other fields in this register
6 CAER	Clear All Enable Requests 0b - Write 0 to only the ERQ field specified in the CERQ field 1b - Write 0 to all fields in ERQ
5 —	Reserved
4-0 CERQ	Clear Enable Request Writes 0 to the corresponding field in ERQ.

6.5.5.9 Set Enable Request (SERQ)

6.5.5.9.1 Offset

Register	Offset
SERQ	1Bh

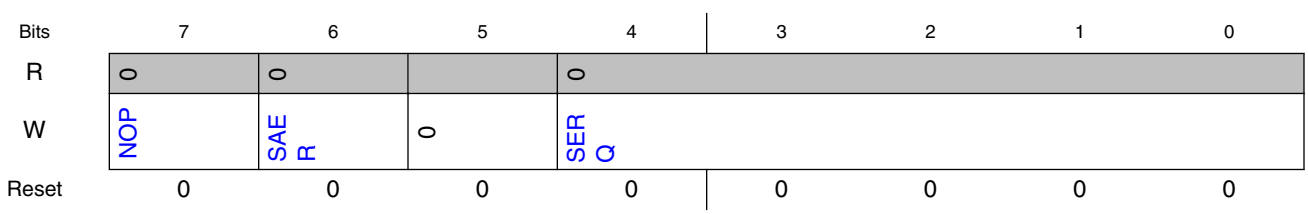
6.5.5.9.2 Function

The SERQ provides a simple memory-mapped mechanism to write 1 to a given field in the ERQ register to enable the DMA request for a given channel. The data value on a register write causes the corresponding field in the ERQ register to be set. Writing 1 to the SAER field provides a global set to 1 function, forcing the entire contents of ERQ register to be 1.

If the NOP field is 1, the command is ignored. This enables you to write 1 to a single, byte-wide register with a 32-bit write that does not affect the other registers addressed in the write. In such a case the other three bytes of the word must all have their NOP field written with 1 so that these registers are not affected by the write.

Reads of this register returns all zeroes.

6.5.5.9.3 Diagram



6.5.5.9.4 Fields

Field	Function
7 NOP	No Op Enable 0b - Normal operation 1b - No operation, ignore the other fields in this register
6 SAER	Set All Enable Requests 0b - Write 1 to only the ERQ field specified in the SERQ field 1b - Write 1 to all fields in ERQ
5 —	Reserved
4-0 SERQ	Set Enable Request Writes 1 to the corresponding field in ERQ.

6.5.5.10 Clear DONE Status Bit (CDNE)

6.5.5.10.1 Offset

Register	Offset
CDNE	1Ch

6.5.5.10.2 Function

The CDNE provides a simple memory-mapped mechanism to write 0 to the DONE field in the TCD of the given channel. The data value on a register write causes the DONE field in the corresponding TCD to be written with 0. Writing 1 to the CADN field provides a global clear function, forcing all DONE fields to be written with 0.

If the NOP field is 1, the command is ignored. This enables you to write 1 to a single, byte-wide register with a 32-bit write that does not affect the other registers addressed in the write. In such a case the other three bytes of the word must all have their NOP field written with 1 so that these registers are not affected by the write.

Reads of this register return all zeroes.

6.5.5.10.3 Diagram

Bits	7	6	5	4	3	2	1	0
R	0	0		0				
W	NOP	CADN	0	CDNE				
Reset	0	0	0	0	0	0	0	0

6.5.5.10.4 Fields

Field	Function
7 NOP	No Op Enable 0b - Normal operation 1b - No operation; all other fields in this register are ignored.
6 CADN	Clears All DONE fields 0b - Writes 0 to only the TCDn_CSR[DONE] field specified in the CDNE field 1b - Writes 0 to all bits in TCDn_CSR[DONE]
5 —	Reserved
4-0 CDNE	Clear DONE field Writes 0 to the corresponding field in TCDn_CSR[DONE]

6.5.5.11 Set START Bit (SSRT)

6.5.5.11.1 Offset

Register	Offset
SSRT	1Dh

6.5.5.11.2 Function

The SSRT register provides a simple memory-mapped mechanism to write 1 to the START field in the TCD of the given channel. The data value on a register write causes the START field in the corresponding TCD to be written with 1. Writing 1 to the SAST field provides a global set to 1 function, forcing all START fields to be written with 1.

If the NOP field is 1, the command is ignored. This enables you to write 1 to a single, byte-wide register with a 32-bit write that does not affect the other registers addressed in the write. In such a case the other three bytes of the word must all have their NOP field written with 1 so that these registers are not affected by the write.

Reads of this register return all zeroes.

6.5.5.11.3 Diagram

Bits	7	6	5	4	3	2	1	0
R	0	0		0				
W	NOP	SAST	0	SSRT				
Reset	0	0	0	0	0	0	0	0

6.5.5.11.4 Fields

Field	Function
7 NOP	No Op Enable 0b - Normal operation 1b - No operation; all other fields in this register are ignored.
6 SAST	Set All START fields (activates all channels) 0b - Write 1 to only the TCDn_CSR[START] field specified in the SSRT field 1b - Write 1 to all bits in TCDn_CSR[START]
5 —	Reserved
4-0 SSRT	Set START field Sets the corresponding field in TCDn_CSR[START]

6.5.5.12 Clear Error (CERR)

6.5.5.12.1 Offset

Register	Offset
CERR	1Eh

6.5.5.12.2 Function

The CERR provides a simple memory-mapped mechanism to write 0 to a given field in the ERR register to disable the error condition field for a given channel. The given value on a register write causes the corresponding field in the ERR register to be written with 0. Writing 1 to the CAEI field provides a global clear to 0 function, forcing the ERR register contents to be written with 0, clearing all channel error indicators. If the NOP field is 1, the command is ignored. This enables you to write multiple-byte registers as a 32-bit word. Reads of this register return all zeroes.

6.5.5.12.3 Diagram

Bits	7	6	5	4	3	2	1	0
R	0	0		0				
W	NOP	CAEI	0	CERR				
Reset	0	0	0	0	0	0	0	0

6.5.5.12.4 Fields

Field	Function
7 NOP	No Op Enable 0b - Normal operation 1b - No operation; all other fields in this register are ignored.
6 CAEI	Clear All Error Indicators 0b - Write 0 to only the ERR field specified in the CERR field 1b - Write 0 to all fields in ERR
5 —	Reserved
4-0 CERR	Clear Error Indicator Writes 0 to the corresponding field in ERR

6.5.5.13 Clear Interrupt Request (CINT)

6.5.5.13.1 Offset

Register	Offset
CINT	1Fh

6.5.5.13.2 Function

The CINT register provides a simple, memory-mapped mechanism to clear a given field in the INT register to disable the interrupt request for a given channel. The given value on a register write causes the corresponding field in the INT register to be cleared. Setting the CAIR field provides a global clear function, forcing the entire contents of the INT to be cleared, disabling all DMA interrupt requests.

If the NOP field is 1, the command is ignored. This enables you to set a single, byte-wide register with a 32-bit write that does not affect the other registers addressed in the write. In such a case the other three bytes of the word would all have their NOP field set to 1 so that these registers are not affected by the write.

Reads of this register return all zeroes.

6.5.5.13.3 Diagram

Bits	7	6	5	4	3	2	1	0
R	0	0		0				
W	NOP	CAIR	0	CINT				
Reset	0	0	0	0	0	0	0	0

6.5.5.13.4 Fields

Field	Function
7 NOP	No Op Enable 0b - Normal operation 1b - No operation; all other fields in this register are ignored.
6 CAIR	Clear All Interrupt Requests 0b - Clear only the INT field specified in the CINT field 1b - Clear all bits in INT
5 —	Reserved
4-0 CINT	Clear Interrupt Request Clears the corresponding field in INT

6.5.5.14 Interrupt Request (INT)

6.5.5.14.1 Offset

Register	Offset
INT	24h

6.5.5.14.2 Function

The INT register provides a bit map for the 32 channels signaling the presence of an interrupt request for each channel. Depending on the appropriate bit setting in the transfer-control descriptors, the eDMA engine generates an interrupt on data transfer completion. The outputs of this register are directly routed to the interrupt controller. During the interrupt-service routine associated with any given channel, it is the software's responsibility to write 0 to the appropriate bit, negating the interrupt request. Typically, a write to the CINT register in the interrupt service routine is used for this purpose.

The state of any given channel's interrupt request is directly affected by writes to this register; it is also affected by writes to the CINT register. On writes to INT, a 1 in any bit position clears the corresponding channel's interrupt request. A 0 in any bit position has no effect on the corresponding channel's current interrupt status. The CINT register is provided so the interrupt request for a single channel can easily be cleared without the need to perform a read-modify-write sequence to the INT register.

6.5.5.14.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	INT31	INT30	INT29	INT28	INT27	INT26	INT25	INT24	INT23	INT22	INT21	INT20	INT19	INT18	INT17	INT16
W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	INT15	INT14	INT13	INT12	INT11	INT10	INT9	INT8	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0
W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6.5.5.14.4 Fields

Field	Function
31 INT31	Interrupt Request 31 0b - The interrupt request for channel 31 is cleared 1b - The interrupt request for channel 31 is active
30 INT30	Interrupt Request 30 0b - The interrupt request for channel 30 is cleared 1b - The interrupt request for channel 30 is active
29 INT29	Interrupt Request 29 0b - The interrupt request for channel 29 is cleared 1b - The interrupt request for channel 29 is active
28 INT28	Interrupt Request 28 0b - The interrupt request for channel 28 is cleared 1b - The interrupt request for channel 28 is active
27 INT27	Interrupt Request 27 0b - The interrupt request for channel 27 is cleared 1b - The interrupt request for channel 27 is active
26 INT26	Interrupt Request 26 0b - The interrupt request for channel 26 is cleared 1b - The interrupt request for channel 26 is active
25 INT25	Interrupt Request 25 0b - The interrupt request for channel 25 is cleared 1b - The interrupt request for channel 25 is active
24 INT24	Interrupt Request 24 0b - The interrupt request for channel 24 is cleared 1b - The interrupt request for channel 24 is active
23 INT23	Interrupt Request 23 0b - The interrupt request for channel 23 is cleared 1b - The interrupt request for channel 23 is active
22 INT22	Interrupt Request 22 0b - The interrupt request for channel 22 is cleared 1b - The interrupt request for channel 22 is active
21 INT21	Interrupt Request 21 0b - The interrupt request for channel 21 is cleared 1b - The interrupt request for channel 21 is active
20 INT20	Interrupt Request 20 0b - The interrupt request for channel 20 is cleared 1b - The interrupt request for channel 20 is active
19 INT19	Interrupt Request 19 0b - The interrupt request for channel 19 is cleared 1b - The interrupt request for channel 19 is active
18 INT18	Interrupt Request 18 0b - The interrupt request for channel 18 is cleared 1b - The interrupt request for channel 18 is active
17 INT17	Interrupt Request 17 0b - The interrupt request for channel 17 is cleared 1b - The interrupt request for channel 17 is active
16 INT16	Interrupt Request 16 0b - The interrupt request for channel 16 is cleared 1b - The interrupt request for channel 16 is active

Table continues on the next page...

Field	Function
15 INT15	Interrupt Request 15 0b - The interrupt request for channel 15 is cleared 1b - The interrupt request for channel 15 is active
14 INT14	Interrupt Request 14 0b - The interrupt request for channel 14 is cleared 1b - The interrupt request for channel 14 is active
13 INT13	Interrupt Request 13 0b - The interrupt request for channel 13 is cleared 1b - The interrupt request for channel 13 is active
12 INT12	Interrupt Request 12 0b - The interrupt request for channel 12 is cleared 1b - The interrupt request for channel 12 is active
11 INT11	Interrupt Request 11 0b - The interrupt request for channel 11 is cleared 1b - The interrupt request for channel 11 is active
10 INT10	Interrupt Request 10 0b - The interrupt request for channel 10 is cleared 1b - The interrupt request for channel 10 is active
9 INT9	Interrupt Request 9 0b - The interrupt request for channel 9 is cleared 1b - The interrupt request for channel 9 is active
8 INT8	Interrupt Request 8 0b - The interrupt request for channel 8 is cleared 1b - The interrupt request for channel 8 is active
7 INT7	Interrupt Request 7 0b - The interrupt request for channel 7 is cleared 1b - The interrupt request for channel 7 is active
6 INT6	Interrupt Request 6 0b - The interrupt request for channel 6 is cleared 1b - The interrupt request for channel 6 is active
5 INT5	Interrupt Request 5 0b - The interrupt request for channel 5 is cleared 1b - The interrupt request for channel 5 is active
4 INT4	Interrupt Request 4 0b - The interrupt request for channel 4 is cleared 1b - The interrupt request for channel 4 is active
3 INT3	Interrupt Request 3 0b - The interrupt request for channel 3 is cleared 1b - The interrupt request for channel 3 is active
2 INT2	Interrupt Request 2 0b - The interrupt request for channel 2 is cleared 1b - The interrupt request for channel 2 is active
1 INT1	Interrupt Request 1 0b - The interrupt request for channel 1 is cleared 1b - The interrupt request for channel 1 is active
0 INT0	Interrupt Request 0 0b - The interrupt request for channel 0 is cleared 1b - The interrupt request for channel 0 is active

6.5.5.15 Error (ERR)

6.5.5.15.1 Offset

Register	Offset
ERR	2Ch

6.5.5.15.2 Function

The ERR register provides a bit map for the 32 channels, signaling the presence of an error for each channel. The eDMA engine signals the occurrence of an error condition by setting the appropriate field in this register. The outputs of this register are enabled by the contents of the EEI register, then logically summed across groups of 16 and 32 channels to form several group error interrupt requests, which are then routed to the interrupt controller. During the execution of the interrupt service routine associated with any DMA errors, it is software's responsibility to reset the appropriate bit to 0, negating the error-interrupt request. Typically, a write to the CERR in the interrupt service routine is used for this purpose. The normal DMA channel completion indicators (setting the TCD DONE field to 1 and the possible generation of an interrupt request) are not affected when an error is detected.

The contents of this register can also be polled because a non-zero value indicates the presence of a channel error regardless of the state of the EEI fields. The state of any given channel's error indicators is affected by writes to this register; it is also affected by writes to the CERR. On writes to the ERR, a 1 in any bit position clears the corresponding channel's error status. A 0 in any bit position has no effect on the corresponding channel's current error status. The CERR is provided so the error indicator for a single channel can easily be reset to 0.

6.5.5.15.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ERR31	ERR30	ERR29	ERR28	ERR27	ERR26	ERR25	ERR24	ERR23	ERR22	ERR21	ERR20	ERR19	ERR18	ERR17	ERR16
W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ERR15	ERR14	ERR13	ERR12	ERR11	ERR10	ERR9	ERR8	ERR7	ERR6	ERR5	ERR4	ERR3	ERR2	ERR1	ERR0
W	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6.5.5.15.4 Fields

Field	Function
31 ERR31	Error In Channel 31 0b - No error in this channel has occurred 1b - An error in this channel has occurred
30 ERR30	Error In Channel 30 0b - No error in this channel has occurred 1b - An error in this channel has occurred
29 ERR29	Error In Channel 29 0b - No error in this channel has occurred 1b - An error in this channel has occurred
28 ERR28	Error In Channel 28 0b - No error in this channel has occurred 1b - An error in this channel has occurred
27 ERR27	Error In Channel 27 0b - No error in this channel has occurred 1b - An error in this channel has occurred
26 ERR26	Error In Channel 26 0b - No error in this channel has occurred 1b - An error in this channel has occurred
25 ERR25	Error In Channel 25 0b - No error in this channel has occurred 1b - An error in this channel has occurred
24 ERR24	Error In Channel 24 0b - No error in this channel has occurred 1b - An error in this channel has occurred
23 ERR23	Error In Channel 23 0b - No error in this channel has occurred 1b - An error in this channel has occurred

Table continues on the next page...

Field	Function
22 ERR22	Error In Channel 22 0b - No error in this channel has occurred 1b - An error in this channel has occurred
21 ERR21	Error In Channel 21 0b - No error in this channel has occurred 1b - An error in this channel has occurred
20 ERR20	Error In Channel 20 0b - No error in this channel has occurred 1b - An error in this channel has occurred
19 ERR19	Error In Channel 19 0b - No error in this channel has occurred 1b - An error in this channel has occurred
18 ERR18	Error In Channel 18 0b - No error in this channel has occurred 1b - An error in this channel has occurred
17 ERR17	Error In Channel 17 0b - No error in this channel has occurred 1b - An error in this channel has occurred
16 ERR16	Error In Channel 16 0b - No error in this channel has occurred 1b - An error in this channel has occurred
15 ERR15	Error In Channel 15 0b - No error in this channel has occurred 1b - An error in this channel has occurred
14 ERR14	Error In Channel 14 0b - No error in this channel has occurred 1b - An error in this channel has occurred
13 ERR13	Error In Channel 13 0b - No error in this channel has occurred 1b - An error in this channel has occurred
12 ERR12	Error In Channel 12 0b - No error in this channel has occurred 1b - An error in this channel has occurred
11 ERR11	Error In Channel 11 0b - No error in this channel has occurred 1b - An error in this channel has occurred
10 ERR10	Error In Channel 10 0b - No error in this channel has occurred 1b - An error in this channel has occurred
9 ERR9	Error In Channel 9 0b - No error in this channel has occurred 1b - An error in this channel has occurred
8 ERR8	Error In Channel 8 0b - No error in this channel has occurred 1b - An error in this channel has occurred
7 ERR7	Error In Channel 7 0b - No error in this channel has occurred 1b - An error in this channel has occurred
6 ERR6	Error In Channel 6 0b - No error in this channel has occurred

Table continues on the next page...

Field	Function
	1b - An error in this channel has occurred
5 ERR5	Error In Channel 5 0b - No error in this channel has occurred 1b - An error in this channel has occurred
4 ERR4	Error In Channel 4 0b - No error in this channel has occurred 1b - An error in this channel has occurred
3 ERR3	Error In Channel 3 0b - No error in this channel has occurred 1b - An error in this channel has occurred
2 ERR2	Error In Channel 2 0b - No error in this channel has occurred 1b - An error in this channel has occurred
1 ERR1	Error In Channel 1 0b - No error in this channel has occurred 1b - An error in this channel has occurred
0 ERR0	Error In Channel 0 0b - No error in this channel has occurred 1b - An error in this channel has occurred

6.5.5.16 Hardware Request Status (HRS)

6.5.5.16.1 Offset

Register	Offset
HRS	34h

6.5.5.16.2 Function

The HRS register provides a bit map for the DMA channels, signaling the presence of a hardware request for each channel. The hardware request status bits reflect the current state of the register and qualified (via the ERQ fields) DMA request signals, as seen by the DMA's arbitration logic. This view into the hardware request signals may be used for debug purposes.

NOTE

These bits reflect the state of the request as seen by the arbitration logic. Therefore, this status is affected by the ERQ bits.

Each HRS field for its respective channel is 1 when a hardware request is present on the channel. After the request is completed and channel is free, the HRS field is automatically changed to 0 by hardware.

6.5.5.16.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	HRS31	HRS30	HRS29	HRS28	HRS27	HRS26	HRS25	HRS24	HRS23	HRS22	HRS21	HRS20	HRS19	HRS18	HRS17	HRS16
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HRS15	HRS14	HRS13	HRS12	HRS11	HRS10	HRS 9	HRS 8	HRS 7	HRS 6	HRS 5	HRS 4	HRS 3	HRS 2	HRS 1	HRS 0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6.5.5.16.4 Fields

Field	Function
31 HRS31	Hardware Request Status Channel 31 0b - A hardware service request for channel 31 is not present 1b - A hardware service request for channel 31 is present
30 HRS30	Hardware Request Status Channel 30 0b - A hardware service request for channel 30 is not present 1b - A hardware service request for channel 30 is present
29 HRS29	Hardware Request Status Channel 29 0b - A hardware service request for channel 29 is not preset 1b - A hardware service request for channel 29 is present
28 HRS28	Hardware Request Status Channel 28 0b - A hardware service request for channel 28 is not present 1b - A hardware service request for channel 28 is present
27 HRS27	Hardware Request Status Channel 27 0b - A hardware service request for channel 27 is not present 1b - A hardware service request for channel 27 is present
26 HRS26	Hardware Request Status Channel 26 0b - A hardware service request for channel 26 is not present 1b - A hardware service request for channel 26 is present
25 HRS25	Hardware Request Status Channel 25 0b - A hardware service request for channel 25 is not present 1b - A hardware service request for channel 25 is present
24 HRS24	Hardware Request Status Channel 24 0b - A hardware service request for channel 24 is not present 1b - A hardware service request for channel 24 is present

Table continues on the next page...

Field	Function
23 HRS23	Hardware Request Status Channel 23 0b - A hardware service request for channel 23 is not present 1b - A hardware service request for channel 23 is present
22 HRS22	Hardware Request Status Channel 22 0b - A hardware service request for channel 22 is not present 1b - A hardware service request for channel 22 is present
21 HRS21	Hardware Request Status Channel 21 0b - A hardware service request for channel 21 is not present 1b - A hardware service request for channel 21 is present
20 HRS20	Hardware Request Status Channel 20 0b - A hardware service request for channel 20 is not present 1b - A hardware service request for channel 20 is present
19 HRS19	Hardware Request Status Channel 19 0b - A hardware service request for channel 19 is not present 1b - A hardware service request for channel 19 is present
18 HRS18	Hardware Request Status Channel 18 0b - A hardware service request for channel 18 is not present 1b - A hardware service request for channel 18 is present
17 HRS17	Hardware Request Status Channel 17 0b - A hardware service request for channel 17 is not present 1b - A hardware service request for channel 17 is present
16 HRS16	Hardware Request Status Channel 16 0b - A hardware service request for channel 16 is not present 1b - A hardware service request for channel 16 is present
15 HRS15	Hardware Request Status Channel 15 0b - A hardware service request for channel 15 is not present 1b - A hardware service request for channel 15 is present
14 HRS14	Hardware Request Status Channel 14 0b - A hardware service request for channel 14 is not present 1b - A hardware service request for channel 14 is present
13 HRS13	Hardware Request Status Channel 13 0b - A hardware service request for channel 13 is not present 1b - A hardware service request for channel 13 is present
12 HRS12	Hardware Request Status Channel 12 0b - A hardware service request for channel 12 is not present 1b - A hardware service request for channel 12 is present
11 HRS11	Hardware Request Status Channel 11 0b - A hardware service request for channel 11 is not present 1b - A hardware service request for channel 11 is present
10 HRS10	Hardware Request Status Channel 10 0b - A hardware service request for channel 10 is not present 1b - A hardware service request for channel 10 is present
9 HRS9	Hardware Request Status Channel 9 0b - A hardware service request for channel 9 is not present 1b - A hardware service request for channel 9 is present
8 HRS8	Hardware Request Status Channel 8 0b - A hardware service request for channel 8 is not present 1b - A hardware service request for channel 8 is present
7 HRS7	Hardware Request Status Channel 7 0b - A hardware service request for channel 7 is not present

Table continues on the next page...

Field	Function
	1b - A hardware service request for channel 7 is present
6 HRS6	Hardware Request Status Channel 6 0b - A hardware service request for channel 6 is not present 1b - A hardware service request for channel 6 is present
5 HRS5	Hardware Request Status Channel 5 0b - A hardware service request for channel 5 is not present 1b - A hardware service request for channel 5 is present
4 HRS4	Hardware Request Status Channel 4 0b - A hardware service request for channel 4 is not present 1b - A hardware service request for channel 4 is present
3 HRS3	Hardware Request Status Channel 3 0b - A hardware service request for channel 3 is not present 1b - A hardware service request for channel 3 is present
2 HRS2	Hardware Request Status Channel 2 0b - A hardware service request for channel 2 is not present 1b - A hardware service request for channel 2 is present
1 HRS1	Hardware Request Status Channel 1 0b - A hardware service request for channel 1 is not present 1b - A hardware service request for channel 1 is present
0 HRS0	Hardware Request Status Channel 0 0b - A hardware service request for channel 0 is not present 1b - A hardware service request for channel 0 is present

6.5.5.17 Enable Asynchronous Request in Stop (EARS)

6.5.5.17.1 Offset

Register	Offset
EARS	44h

6.5.5.17.2 Function

The EARS register is used to enable or disable the DMA requests in Enable Request (ERQ) by AND'ing the bits of these two registers.

6.5.5.17.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	EDREQ_31	EDREQ_30	EDREQ_29	EDREQ_28	EDREQ_27	EDREQ_26	EDREQ_25	EDREQ_24	EDREQ_23	EDREQ_22	EDREQ_21	EDREQ_20	EDREQ_19	EDREQ_18	EDREQ_17	EDREQ_16
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	EDREQ_15	EDREQ_14	EDREQ_13	EDREQ_12	EDREQ_11	EDREQ_10	EDREQ_9	EDREQ_8	EDREQ_7	EDREQ_6	EDREQ_5	EDREQ_4	EDREQ_3	EDREQ_2	EDREQ_1	EDREQ_0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6.5.5.17.4 Fields

Field	Function
31 EDREQ_31	Enable asynchronous DMA request in stop mode for channel 31. 0b - Disable asynchronous DMA request for channel 31 1b - Enable asynchronous DMA request for channel 31
30 EDREQ_30	Enable asynchronous DMA request in stop mode for channel 30. 0b - Disable asynchronous DMA request for channel 30 1b - Enable asynchronous DMA request for channel 30
29 EDREQ_29	Enable asynchronous DMA request in stop mode for channel 29. 0b - Disable asynchronous DMA request for channel 29 1b - Enable asynchronous DMA request for channel 29
28 EDREQ_28	Enable asynchronous DMA request in stop mode for channel 28. 0b - Disable asynchronous DMA request for channel 28 1b - Enable asynchronous DMA request for channel 28
27 EDREQ_27	Enable asynchronous DMA request in stop mode for channel 27. 0b - Disable asynchronous DMA request for channel 27 1b - Enable asynchronous DMA request for channel 27
26 EDREQ_26	Enable asynchronous DMA request in stop mode for channel 26. 0b - Disable asynchronous DMA request for channel 26 1b - Enable asynchronous DMA request for channel 26
25 EDREQ_25	Enable asynchronous DMA request in stop mode for channel 25. 0b - Disable asynchronous DMA request for channel 25 1b - Enable asynchronous DMA request for channel 25
24 EDREQ_24	Enable asynchronous DMA request in stop mode for channel 24. 0b - Disable asynchronous DMA request for channel 24 1b - Enable asynchronous DMA request for channel 24
23 EDREQ_23	Enable asynchronous DMA request in stop mode for channel 23. 0b - Disable asynchronous DMA request for channel 23 1b - Enable asynchronous DMA request for channel 23
22	Enable asynchronous DMA request in stop mode for channel 22. 0b - Disable asynchronous DMA request for channel 22

Table continues on the next page...

Field	Function
EDREQ_22	1b - Enable asynchronous DMA request for channel 22
21 EDREQ_21	Enable asynchronous DMA request in stop mode for channel 21. 0b - Disable asynchronous DMA request for channel 21 1b - Enable asynchronous DMA request for channel 21
20 EDREQ_20	Enable asynchronous DMA request in stop mode for channel 20. 0b - Disable asynchronous DMA request for channel 20 1b - Enable asynchronous DMA request for channel 20
19 EDREQ_19	Enable asynchronous DMA request in stop mode for channel 19. 0b - Disable asynchronous DMA request for channel 19 1b - Enable asynchronous DMA request for channel 19
18 EDREQ_18	Enable asynchronous DMA request in stop mode for channel 18. 0b - Disable asynchronous DMA request for channel 18 1b - Enable asynchronous DMA request for channel 18
17 EDREQ_17	Enable asynchronous DMA request in stop mode for channel 17. 0b - Disable asynchronous DMA request for channel 17 1b - Enable asynchronous DMA request for channel 17
16 EDREQ_16	Enable asynchronous DMA request in stop mode for channel 16. 0b - Disable asynchronous DMA request for channel 16 1b - Enable asynchronous DMA request for channel 16
15 EDREQ_15	Enable asynchronous DMA request in stop mode for channel 15. 0b - Disable asynchronous DMA request for channel 15 1b - Enable asynchronous DMA request for channel 15
14 EDREQ_14	Enable asynchronous DMA request in stop mode for channel 14. 0b - Disable asynchronous DMA request for channel 14 1b - Enable asynchronous DMA request for channel 14
13 EDREQ_13	Enable asynchronous DMA request in stop mode for channel 13. 0b - Disable asynchronous DMA request for channel 13 1b - Enable asynchronous DMA request for channel 13
12 EDREQ_12	Enable asynchronous DMA request in stop mode for channel 12. 0b - Disable asynchronous DMA request for channel 12 1b - Enable asynchronous DMA request for channel 12
11 EDREQ_11	Enable asynchronous DMA request in stop mode for channel 11. 0b - Disable asynchronous DMA request for channel 11 1b - Enable asynchronous DMA request for channel 11
10 EDREQ_10	Enable asynchronous DMA request in stop mode for channel 10. 0b - Disable asynchronous DMA request for channel 10 1b - Enable asynchronous DMA request for channel 10
9 EDREQ_9	Enable asynchronous DMA request in stop mode for channel 9. 0b - Disable asynchronous DMA request for channel 9 1b - Enable asynchronous DMA request for channel 9
8 EDREQ_8	Enable asynchronous DMA request in stop mode for channel 8. 0b - Disable asynchronous DMA request for channel 8 1b - Enable asynchronous DMA request for channel 8
7 EDREQ_7	Enable asynchronous DMA request in stop mode for channel 7. 0b - Disable asynchronous DMA request for channel 7 1b - Enable asynchronous DMA request for channel 7
6 EDREQ_6	Enable asynchronous DMA request in stop mode for channel 6. 0b - Disable asynchronous DMA request for channel 6 1b - Enable asynchronous DMA request for channel 6
5	Enable asynchronous DMA request in stop mode for channel 5.

Table continues on the next page...

Field	Function
EDREQ_5	0b - Disable asynchronous DMA request for channel 5 1b - Enable asynchronous DMA request for channel 5
4 EDREQ_4	Enable asynchronous DMA request in stop mode for channel 4. 0b - Disable asynchronous DMA request for channel 4 1b - Enable asynchronous DMA request for channel 4
3 EDREQ_3	Enable asynchronous DMA request in stop mode for channel 3. 0b - Disable asynchronous DMA request for channel 3 1b - Enable asynchronous DMA request for channel 3
2 EDREQ_2	Enable asynchronous DMA request in stop mode for channel 2. 0b - Disable asynchronous DMA request for channel 2 1b - Enable asynchronous DMA request for channel 2
1 EDREQ_1	Enable asynchronous DMA request in stop mode for channel 1. 0b - Disable asynchronous DMA request for channel 1 1b - Enable asynchronous DMA request for channel 1
0 EDREQ_0	Enable asynchronous DMA request in stop mode for channel 0. 0b - Disable asynchronous DMA request for channel 0 1b - Enable asynchronous DMA request for channel 0

6.5.5.18 Channel Priority (DCHPRI0 - DCHPRI31)

6.5.5.18.1 Offset

For n = 0 to 31:

Register	Offset
DCHPRIn	100h + (n + 3 - 2 × (n mod 4))

6.5.5.18.2 Function

When fixed-priority channel arbitration is enabled (CR[ERCA] = 0), the contents of these registers define the unique priorities associated with each channel within a group. The channel priorities are evaluated by numeric value; for example, 0 is the lowest priority, 1 is the next higher priority, then 2, 3, and so on. Software must program the channel priorities with unique values; otherwise, a configuration error is reported. The range of the priority value is limited to the values of 0 through 15. When read, the GRPPRI bits of the DCHPRIn register reflect the current priority level of the group of channels in which the corresponding channel resides. GRPPRI bits are not affected by writes to the DCHPRIn registers. The group priority is assigned in the DMA control register.

6.5.5.18.3 Diagram

Bits	7	6	5	4	3	2	1	0
R	ECP	DPA	GRPPRI		CHPRI			
W								
Reset	See Register reset values.							

6.5.5.18.4 Register reset values

Register	Reset value
DCHPRI0	00h
DCHPRI1	01h
DCHPRI2	02h
DCHPRI3	03h
DCHPRI4	04h
DCHPRI5	05h
DCHPRI6	06h
DCHPRI7	07h
DCHPRI8	08h
DCHPRI9	09h
DCHPRI10	0Ah
DCHPRI11	0Bh
DCHPRI12	0Ch
DCHPRI13	0Dh
DCHPRI14	0Eh
DCHPRI15	0Fh
DCHPRI16	10h
DCHPRI17	11h
DCHPRI18	12h
DCHPRI19	13h
DCHPRI20	14h
DCHPRI21	15h
DCHPRI22	16h
DCHPRI23	17h
DCHPRI24	18h
DCHPRI25	19h
DCHPRI26	1Ah
DCHPRI27	1Bh
DCHPRI28	1Ch
DCHPRI29	1Dh

Table continues on the next page...

Register	Reset value
DCHPRI30	1Eh
DCHPRI31	1Fh

6.5.5.18.5 Fields

Field	Function
7 ECP	Enable Channel Preemption. This field resets to 0. 0b - Channel n cannot be suspended by a higher priority channel's service request 1b - Channel n can be temporarily suspended by the service request of a higher priority channel
6 DPA	Disable Preempt Ability. This field resets to 0. 0b - Channel n can suspend a lower priority channel 1b - Channel n cannot suspend any channel, regardless of channel priority
5-4 GRPPRI	Channel n Current Group Priority Group priority assigned to this channel group when fixed-priority arbitration is enabled. This field is read-only; writes are ignored.
3-0 CHPRI	Channel n Arbitration Priority Channel priority when fixed-priority arbitration is enabled.

6.5.5.19 TCD Source Address (TCD0_SADDR - TCD31_SADDR)

6.5.5.19.1 Offset

For n = 0 to 31:

Register	Offset
TCDn_SADDR	1000h + (n × 20h)

6.5.5.19.2 Function

This register contains the source address of the transfer.

6.5.5.19.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SADDR															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SADDR															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

6.5.5.19.4 Fields

Field	Function
31-0 SADDR	Source Address Memory address pointing to the source data.

6.5.5.20 TCD Signed Source Address Offset (TCD0_SOFF - TCD31_SOFF)

6.5.5.20.1 Offset

For n = 0 to 31:

Register	Offset
TCDn_SOFF	1004h + (n × 20h)

6.5.5.20.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SOFF															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

6.5.5.20.3 Fields

Field	Function
15-0 SOFF	Source address signed offset Sign-extended offset applied to the current source address to form the next-state value as each source read is completed.

6.5.5.21 TCD Transfer Attributes (TCD0_ATTR - TCD31_ATTR)

6.5.5.21.1 Offset

For n = 0 to 31:

Register	Offset
TCDn_ATTR	1006h + (n × 20h)

6.5.5.21.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SMOD					SSIZE			DMOD					DSIZE		
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

6.5.5.21.3 Fields

Field	Function
15-11 SMOD	Source Address Modulo Any non-zero value in this field defines a specific address range specified to be the value after SADDR + SOFF calculation is performed on the original register value. Setting this field provides the ability to implement a circular data queue easily. For data queues requiring power-of-2 size bytes, the queue should start at a 0-modulo-size address and the SMOD field should be set to the appropriate value for the queue, freezing the desired number of upper address bits. The value programmed into this field specifies the number of lower address bits allowed to change. For a circular queue application, the SOFF is typically set to the transfer size to implement post-increment addressing with the SMOD function constraining the addresses to a 0-modulo-size range. 00000b - Source address modulo feature is disabled 00001-11111b - Value defines address range used to set up circular data queue
10-8 SSIZE	Source data transfer size **NOTE:** 1. Using a reserved value causes a configuration error. 2. The eDMA defaults to privileged data access for all transactions.

Table continues on the next page...

Field	Function
	000b - 8-bit 001b - 16-bit 010b - 32-bit 011b - 64-bit 100b - Reserved 101b - 32-byte burst (4 beats of 64 bits) 110b - Reserved 111b - Reserved
7-3 DMOD	Destination Address Modulo See the SMOD definition.
2-0 DSIZE	Destination data transfer size See the SSIZE definition.

6.5.5.22 TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD0_NBYTES_MLNO - TCD31_NBYTES_MLNO)

6.5.5.22.1 Offset

For n = 0 to 31:

Register	Offset
TCDn_NBYTES_MLNO	1008h + (n × 20h)

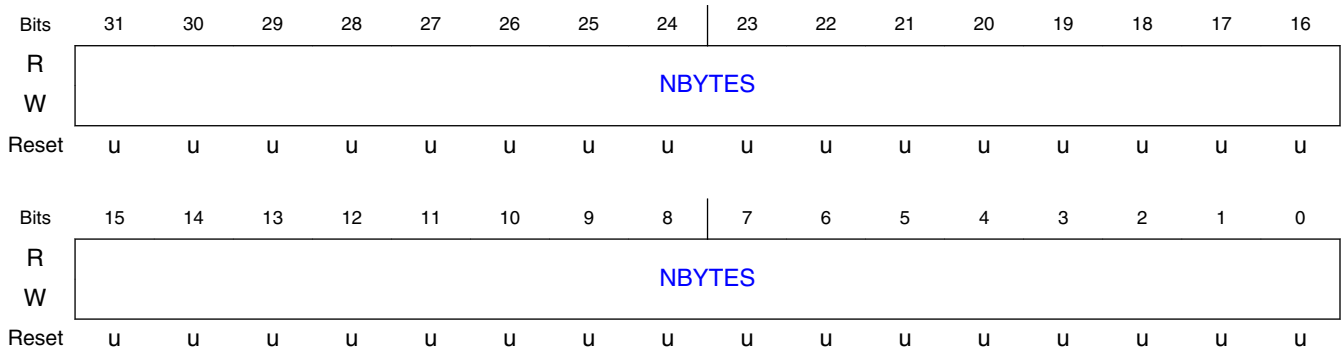
6.5.5.22.2 Function

This register, or one of the next two registers (TCD_NBYTES_MLOFFNO, TCD_NBYTES_MLOFFYES), that defines the number of bytes to transfer per request. Which register to use depends on whether minor loop mapping is disabled, is enabled but not used for this channel, or is enabled and used.

TCD word 2 is defined as follows if minor loop mapping is disabled (CR[EMLM] = 0).

If minor loop mapping is enabled, see the TCD_NBYTES_MLOFFNO and TCD_NBYTES_MLOFFYES register descriptions for the definition of TCD word 2.

6.5.5.22.3 Diagram



6.5.5.22.4 Fields

Field	Function
31-0 NBYTES	Minor Byte Transfer Count Number of bytes to be transferred in each service request of the channel. As a channel activates, the appropriate TCD contents load into the eDMA engine, and the appropriate reads and writes are performed until the minor byte transfer count has transferred. This is an indivisible operation and cannot be halted. It can, however, be stalled by using the bandwidth control field, or via preemption. After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, and the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed. **NOTE:** An NBYTES value of 0x0000_0000 is interpreted as a 4 GB transfer.

6.5.5.23 TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD0_NBYTES_MLOFFNO - TCD31_NBYTES_MLOFFNO)

6.5.5.23.1 Offset

For n = 0 to 31:

Register	Offset
TCDn_NBYTES_MLOFFNO	1008h + (n × 20h)

6.5.5.23.2 Function

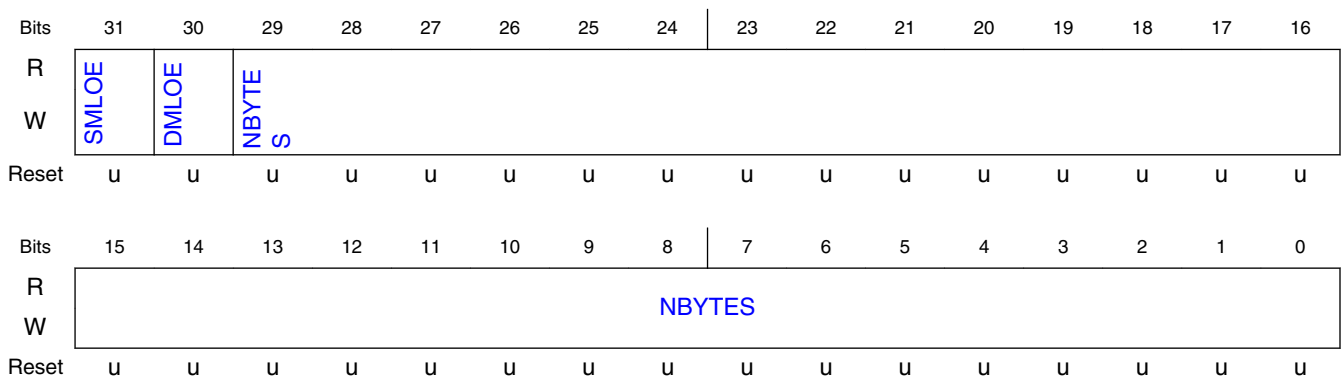
One of three registers (this register, TCD_NBYTES_MLNO, or TCD_NBYTES_MLOFFYES), that defines the number of bytes to transfer per request. Which register to use depends on whether minor loop mapping is disabled, is enabled but not used for this channel, or is enabled and used.

TCD word 2 is defined as follows if:

- Minor loop mapping is enabled (CR[EMLM] = 1) and
- SMLOE = 0 and DMLOE = 0

If minor loop mapping is enabled and SMLOE = 1 or DMLOE = 1, refer to the TCD_NBYTES_MLOFFYES register description. If minor loop mapping is disabled, refer to the TCD_NBYTES_MLNO register description.

6.5.5.23.3 Diagram



6.5.5.23.4 Fields

Field	Function
31 SMLOE	Source Minor Loop Offset Enable Specifies whether the minor loop offset is applied to the source address when the minor loop completes. 0b - The minor loop offset is not applied to the SADDR 1b - The minor loop offset is applied to the SADDR
30 DMLOE	Destination Minor Loop Offset Enable Specifies whether the minor loop offset is applied to the destination address when the minor loop completes. 0b - The minor loop offset is not applied to the DADDR 1b - The minor loop offset is applied to the DADDR
29-0 NBYTES	Minor Byte Transfer Count Number of bytes to be transferred in each service request of the channel.

Field	Function
	As a channel activates, the appropriate TCD contents load into the eDMA engine, and the appropriate reads and writes are performed until the minor byte transfer count has transferred. This is an indivisible operation and cannot be halted. It can, however, be stalled by using the bandwidth control field, or via preemption. After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, and the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed.

6.5.5.24 TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD0_NBYTES_MLOFFYES - TCD31_NBYTES_MLOFFYES)

6.5.5.24.1 Offset

For n = 0 to 31:

Register	Offset
TCDn_NBYTES_MLOFFYES	1008h + (n × 20h)

6.5.5.24.2 Function

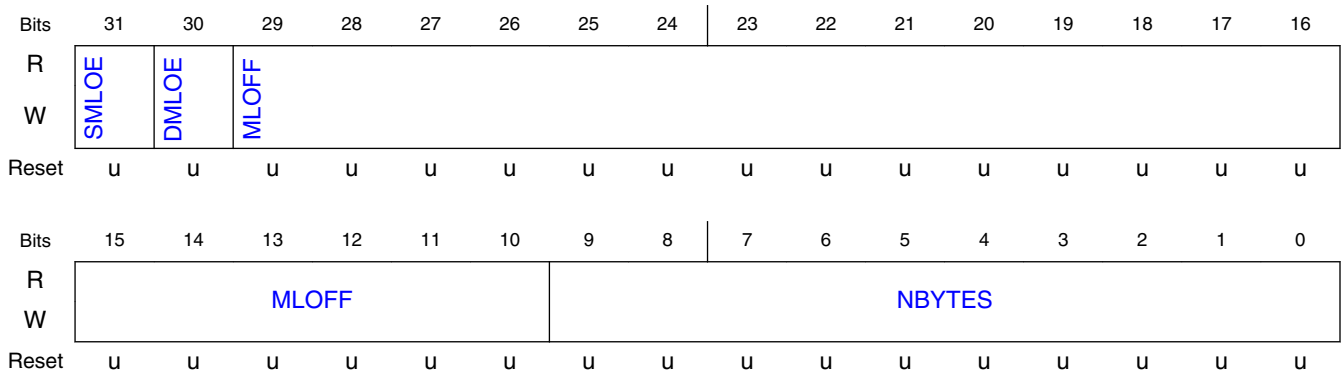
One of three registers (this register, TCD_NBYTES_MLNO, or TCD_NBYTES_MLOFFNO), that defines the number of bytes to transfer per request. Which register to use depends on whether minor loop mapping is disabled, is enabled but not used for this channel, or is enabled and used.

TCD word 2 is defined as follows if:

- Minor loop mapping is enabled (CR[EMLM] = 1) and
- Minor loop offset is enabled (SMLOE or DMLOE = 1)

If minor loop mapping is enabled and SMLOE = 0 and DMLOE = 0, refer to the TCD_NBYTES_MLOFFNO register description. If minor loop mapping is disabled, refer to the TCD_NBYTES_MLNO register description.

6.5.5.24.3 Diagram



6.5.5.24.4 Fields

Field	Function
31 SMLOE	Source Minor Loop Offset Enable Specifies whether the minor loop offset is applied to the source address when the minor loop completes. 0b - The minor loop offset is not applied to the SADDR 1b - The minor loop offset is applied to the SADDR
30 DMLOE	Destination Minor Loop Offset Enable Specifies whether the minor loop offset is applied to the destination address when the minor loop completes. 0b - The minor loop offset is not applied to the DADDR 1b - The minor loop offset is applied to the DADDR
29-10 MLOFF	If SMLOE = 1 or DMLOE = 1, this field represents a sign-extended offset applied to the source or destination address to form the next-state value after the minor loop completes.
9-0 NBYTES	Minor Byte Transfer Count Number of bytes to be transferred in each service request of the channel. As a channel activates, the appropriate TCD contents load into the eDMA engine, and the appropriate reads and writes are performed until the minor byte transfer count has transferred. This is an indivisible operation and cannot be halted. It can, however, be stalled by using the bandwidth control field, or via preemption. After the minor count is exhausted, the SADDR and DADDR values are written back into the TCD memory, and the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed.

6.5.5.25 TCD Last Source Address Adjustment (TCD0_SLAST - TCD31_SLAST)

6.5.5.25.1 Offset

For n = 0 to 31:

Register	Offset
TCDn_SLAST	100Ch + (n × 20h)

6.5.5.25.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SLAST															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SLAST															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

6.5.5.25.3 Fields

Field	Function
31-0 SLAST	Last Source Address Adjustment Adjustment value added to the source address at the completion of the major iteration count. This value can be applied to restore the source address to the initial value, or adjust the address to reference the next data structure. This register uses two's complement notation; the overflow bit is discarded.

6.5.5.26 TCD Destination Address (TCD0_DADDR - TCD31_DADDR)

6.5.5.26.1 Offset

For n = 0 to 31:

Register	Offset
TCDn_DADDR	1010h + (n × 20h)

6.5.5.26.2 Function

This register contains the destination address of the transfer.

6.5.5.26.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DADDR															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DADDR															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

6.5.5.26.4 Fields

Field	Function
31-0 DADDR	Destination Address Memory address pointing to the destination data.

6.5.5.27 TCD Signed Destination Address Offset (TCD0_DOFF - TCD31_DOFF)

6.5.5.27.1 Offset

For n = 0 to 31:

Register	Offset
TCDn_DOFF	1014h + (n × 20h)

6.5.5.27.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DOFF															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

6.5.5.27.3 Fields

Field	Function
15-0 DOFF	Destination Address Signed Offset Sign-extended offset applied to the current destination address to form the next-state value as each destination write is completed.

6.5.5.28 TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD0_CITER_ELINKNO - TCD31_CITER_ELINKNO)

6.5.5.28.1 Offset

For n = 0 to 31:

Register	Offset
TCDn_CITER_ELINKNO	1016h + (n × 20h)

6.5.5.28.2 Function

This register contains the minor-loop channel-linking configuration and the channel's current iteration count. It is the same register as TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD0_CITER_ELINKYES - TCD31_CITER_ELINKYES), but its fields are defined differently based on the state of the ELINK field. If the ELINK field is 0, this register is defined as follows.

6.5.5.28.3 Diagram

6.5.5.28.4 Fields

Field	Function
15 ELINK	Enable channel-to-channel linking on minor-loop complete As the channel completes the minor loop, this field enables linking to another channel, defined by the LINKCH field. The link target channel initiates a channel service request via an internal mechanism that sets TCDn_CSR[START] of the specified channel. If channel linking is disabled, the CITER value is extended to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **NOTE:** This field must be equal to BITER[ELINK]; otherwise, a configuration error is reported. 0b - Channel-to-channel linking is disabled 1b - Channel-to-channel linking is enabled
14-0 CITER	Current Major Iteration Count This field is the current major loop count for the channel. It is decremented each time the minor loop is completed and updated in the transfer control descriptor memory. After the major iteration count is exhausted, the channel performs a number of operations, for example, final source and destination address calculations. It optionally generates an interrupt to signal channel completion before reloading the CITER field from the Beginning Iteration Count (BITER) field. **NOTE:** 1. When the CITER field is initially loaded by software, it must be set to the same value as that contained in the BITER field. 2. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

6.5.5.29 TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD0_CITER_ELINKYES - TCD31_CITER_ELINKYES)

6.5.5.29.1 Offset

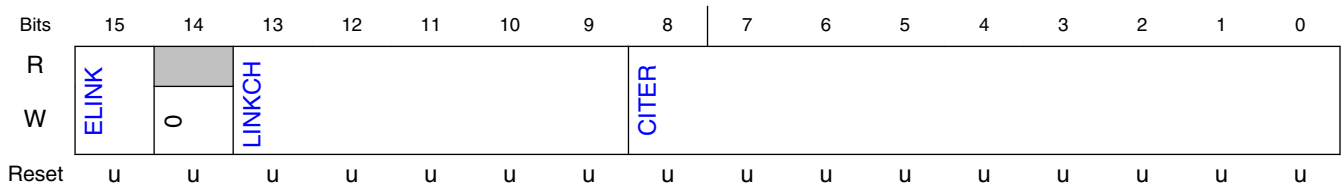
For n = 0 to 31:

Register	Offset
TCDn_CITER_ELINKYES	1016h + (n × 20h)

6.5.5.29.2 Function

This register contains the minor-loop channel-linking configuration and the channel's current iteration count. It is the same register as TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD0_CITER_ELINKNO - TCD31_CITER_ELINKNO), but its fields are defined differently based on the state of the ELINK field. If the ELINK field is 1, this register is defined as follows.

6.5.5.29.3 Diagram



6.5.5.29.4 Fields

Field	Function
15 ELINK	Enable channel-to-channel linking on minor-loop complete As the channel completes the minor loop, this field enables linking to another channel, defined by the LINKCH field. The link target channel initiates a channel service request via an internal mechanism that sets TCDn_CSR[START] of the specified channel. If channel linking is disabled, the CITER value is extended to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **NOTE:** This field must be equal to BITER[ELINK]; otherwise, a configuration error is reported. 0b - Channel-to-channel linking is disabled 1b - Channel-to-channel linking is enabled
14 —	Reserved
13-9 LINKCH	Minor Loop Link Channel Number If channel-to-channel linking is enabled (ELINK = 1), then after the minor loop is exhausted, the eDMA engine initiates a channel service request to the channel defined by this field, by setting that channel's TCDn_CSR[START].
8-0 CITER	Current Major Iteration Count This field is the current major loop count for the channel. It is decremented each time the minor loop is completed and updated in the transfer control descriptor memory. After the major iteration count is exhausted, the channel performs a number of operations, for example, final source and destination address calculations. It optionally generates an interrupt to signal channel completion before reloading the CITER field from the Beginning Iteration Count (BITER) field. **NOTE:** 1. When the CITER field is initially loaded by software, it must be set to the same value as that contained in the BITER field. 2. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

6.5.5.30 TCD Last Destination Address Adjustment/Scatter Gather Address (TCD0_DLASTSGA - TCD31_DLASTSGA)

6.5.5.30.1 Offset

For n = 0 to 31:

Register	Offset
TCDn_DLASTSGA	1018h + (n × 20h)

6.5.5.30.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DLASTSGA															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DLASTSGA															
Reset	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u	u

6.5.5.30.3 Fields

Field	Function
31-0 DLASTSGA	Destination last address adjustment, or next memory address TCD for channel (scatter/gather) If (TCDn_CSR[ESG] = 0) then: • This is the adjustment value added to the destination address at the completion of the major iteration count. This value can apply to restore the destination address to the initial value or adjust the address to reference the next data structure. • This field uses two's complement notation for the final destination address adjustment. Otherwise: • This address points to the beginning of a 0-modulo 32-byte region containing the next TCD to be loaded into this channel. This channel reload is performed as the major iteration count completes. The scatter/gather address must be 0-modulo 32-byte; otherwise a configuration error is reported.

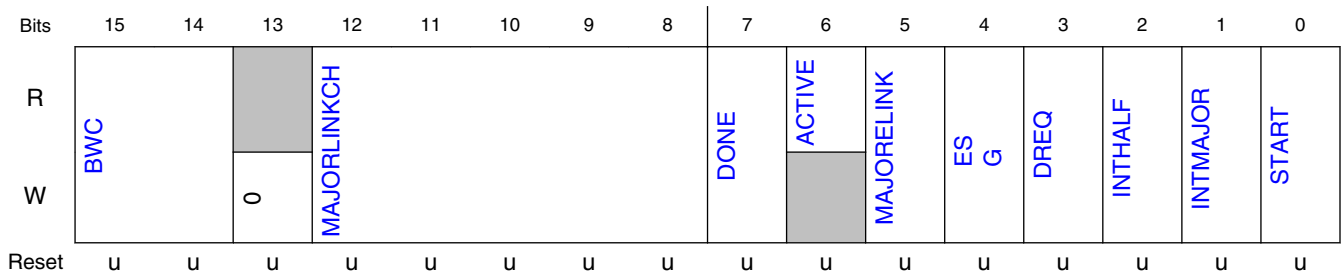
6.5.5.31 TCD Control and Status (TCD0_CSR - TCD31_CSR)

6.5.5.31.1 Offset

For n = 0 to 31:

Register	Offset
TCDn_CSR	101Ch + (n × 20h)

6.5.5.31.2 Diagram



6.5.5.31.3 Fields

Field	Function
15-14 BWC	Bandwidth Control Throttles the amount of bus bandwidth consumed by the eDMA. Generally, as the eDMA processes the minor loop, it continuously generates read/write sequences until the minor count is exhausted. This field forces the eDMA to stall after the completion of each read/write access to control the bus request bandwidth seen by the crossbar switch. **NOTE:** If the source and destination sizes are equal, this field is ignored between the first and second transfers and after the last write of each minor loop. This behavior is a side effect of reducing start-up latency. 00b - No eDMA engine stalls 01b - Reserved 10b - eDMA engine stalls for 4 cycles after each R/W 11b - eDMA engine stalls for 8 cycles after each R/W
13 —	Reserved
12-8 MAJORLINKCH	Major Loop Link Channel Number If (MAJORELINK = 0) then: • No channel-to-channel linking, or chaining, is performed after the major loop counter is exhausted. Otherwise: • After the major loop counter is exhausted, the eDMA engine initiates a channel service request at the channel defined by this field by setting that channel's START bit.
7 DONE	Channel Done This field indicates whether the eDMA has completed the major loop. The eDMA engine sets the value of this field to 1 when the CITER count reaches zero. The value of this field is reset to 0 by the hardware (when the channel is activated) or by software. **NOTE:** This field must be 0 to write the MAJORELINK or ESG fields.
6	Channel Active

Table continues on the next page...

Field	Function
ACTIVE	This field indicates whether the channel is currently in execution. The eDMA sets the value of this field to 1 when channel service begins, and resets it to 0 as the minor loop completes or when any error condition is detected.
5 MAJORELINK	Enable channel-to-channel linking on major loop complete As the channel completes the major loop, this field controls linking to another channel, defined by MAJORLINKCH. The link target channel initiates a channel service request via an internal mechanism that sets TCDn_CSR[START] of the specified channel. **NOTE:** To support the dynamic linking coherency model, this field is forced to zero when written to when TCDn_CSR[DONE] is set. 0b - Channel-to-channel linking is disabled 1b - Channel-to-channel linking is enabled
4 ESG	Enable Scatter/Gather Processing As the channel completes the major loop, this field controls scatter/gather processing in the current channel. If enabled, the eDMA engine uses DLASTSGA as a memory pointer to a 0-modulo 32-bit address containing a 32-byte data structure loaded as the TCD into local memory. **NOTE:** To support the dynamic scatter/gather coherency model, this field is forced to zero when written to when TCDn_CSR[DONE] is set. 0b - The current channel's TCD is normal format 1b - The current channel's TCD specifies a scatter gather format
3 DREQ	Disable Request If the value of this field is 1, eDMA hardware automatically writes 0 to the corresponding ERQ field when the current major iteration count reaches zero. 0b - The channel's ERQ field is not affected 1b - The channel's ERQ field value changes to 0 when the major loop is complete
2 INTHALF	Enable an interrupt when major counter is half complete. If the value of this field is 1, the channel generates an interrupt request by setting the appropriate field in the INT register when the current major iteration count reaches the halfway point. Specifically, the comparison performed by the eDMA engine is (CITER == (BITER >> 1)). This halfway point interrupt request is provided to support double-buffered, also known as ping-pong, schemes or other types of data movement where the processor needs an early indication of the transfer's progress. **NOTE:** If BITER = 1, do not use INTHALF. Use INTMAJOR instead. 0b - Half-point interrupt is disabled 1b - Half-point interrupt is enabled
1 INTMAJOR	Enable an interrupt when major iteration count completes. If the value of this field is 1, the channel generates an interrupt request by setting the appropriate field in the INT when the current major iteration count reaches zero. 0b - End of major loop interrupt is disabled 1b - End of major loop interrupt is enabled
0 START	Channel Start If the value of this field is 1, the channel is requesting service. eDMA hardware automatically writes 0 to this field after the channel begins execution. 0b - Channel is not explicitly started 1b - Channel is explicitly started via a software initiated service request

6.5.5.32 TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD0_BITER_ELINKNO - TCD31_BITER_ELINKNO)

6.5.5.32.1 Offset

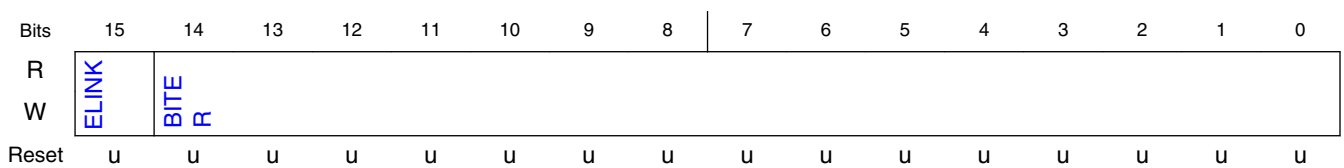
For n = 0 to 31:

Register	Offset
TCDn_BITER_ELINKNO	101Eh + (n × 20h)

6.5.5.32.2 Function

If TCDn_BITER[ELINK] is 0, the TCDn_BITER register is defined as follows.

6.5.5.32.3 Diagram



6.5.5.32.4 Fields

Field	Function
15 ELINK	Enables channel-to-channel linking on minor loop complete As the channel completes the minor loop, this field enables linking to another channel, defined by BITER[LINKCH]. The link target channel initiates a channel service request via an internal mechanism that sets TCDn_CSR[START] of the specified channel. If channel linking is disabled, the BITER value extends to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. 0b - Channel-to-channel linking is disabled 1b - Channel-to-channel linking is enabled
14-0 BITER	Starting Major Iteration Count As the TCD is first loaded by software, this 9-bit (ELINK = 1) or 15-bit (ELINK = 0) field must be equal to the value in the CITER field. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. **NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

6.5.5.33 TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD0_BITER_ELINKYES - TCD31_BITER_ELINKYES)

6.5.5.33.1 Offset

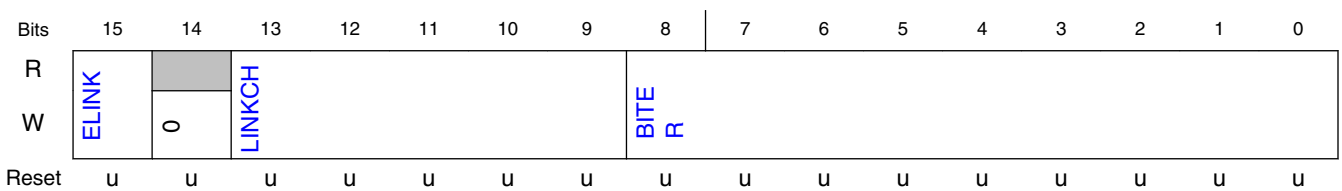
For n = 0 to 31:

Register	Offset
TCDn_BITER_ELINKYES	101Eh + (n × 20h)

6.5.5.33.2 Function

If TCDn_BITER[ELINK] is 1, the TCDn_BITER register is defined as follows.

6.5.5.33.3 Diagram



6.5.5.33.4 Fields

Field	Function
15 ELINK	Enables channel-to-channel linking on minor loop complete As the channel completes the minor loop, this field enables linking to another channel, defined by BITER[LINKCH]. The link target channel initiates a channel service request via an internal mechanism that sets TCDn_CSR[START] of the specified channel. If channel linking disables, the BITER value extends to 15 bits in place of a link channel number. If the major loop is exhausted, this link mechanism is suppressed in favor of the MAJORELINK channel linking. **NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. 0b - Channel-to-channel linking is disabled 1b - Channel-to-channel linking is enabled
14 —	Reserved

Table continues on the next page...

Field	Function
13-9 LINKCH	Link Channel Number If channel-to-channel linking is enabled (ELINK = 1), then after the minor loop is exhausted, the eDMA engine initiates a channel service request at the channel defined by this field, by setting that channel's TCDn_CSR[START]. **NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field.
8-0 BITER	Starting major iteration count As the TCD is first loaded by software, this 9-bit (ELINK = 1) or 15-bit (ELINK = 0) field must be equal to the value in the CITER field. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. **NOTE:** When the software loads the TCD, this field must be set equal to the corresponding CITER field; otherwise, a configuration error is reported. As the major iteration count is exhausted, the contents of this field are reloaded into the CITER field. If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

Chapter 7
System Security

7.1 Chapter overview

This chapter provides an overview of the following chip security components, explaining the purpose and features of each of them.

- System Boot ROM including High Assurance Boot (HAB)
- Secure Non-Volatile Storage (SNVS) with the security monitor, key storage, and real-time clock
- Data co-processor (DCP) with cryptographic acceleration, hardware encryption, and hash algorithm engine for AES128 and SHA-1/256
- True Random Number Generator (TRNG)
- On-chip One-Time Programmable Element Controller (OCOTP_CTRL) with on-chip electrical fuse arrays
- Central Security Unit (CSU)
- System JTAG Controller (SJC) with secure debug
- Bus Encryption Engine (BEE) for on-the-fly FlexSPI(QSPI) Flash decryption

The detailed description of each component can be found in the Security Reference Manual for this chip.

7.2 Feature summary

This figure shows a simplified diagram of the security subsystem:

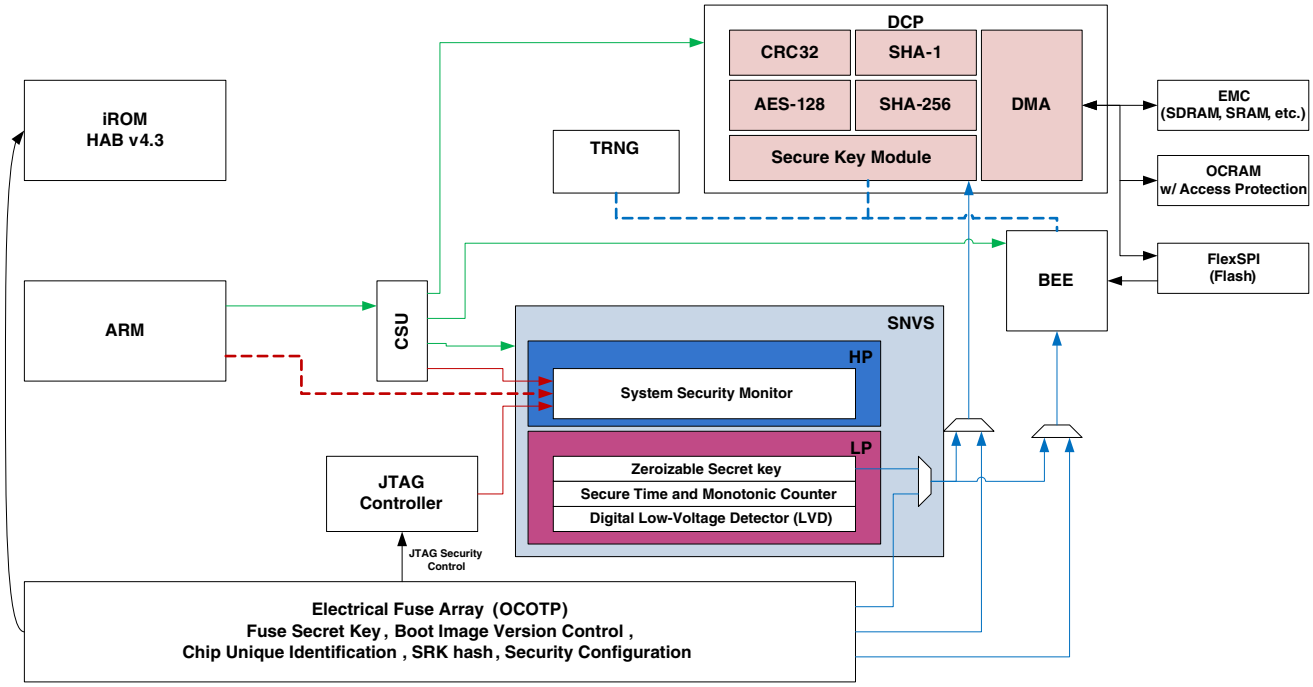


Figure 7-1. Security subsystem (simplified)

All platforms built using this chip share a general need for security, though the specific security requirements vary greatly from platform to platform. For example, portable consumer devices need to protect a different type and cost of assets than automotive or industrial platforms. Each market must be protected against different kinds of attacks. The platform designers need an appropriate set of countermeasures to meet the security needs of their specific platform.

To help the platform designers to meet the requirements of each market, the chip incorporates a range of security features. Most of these features provide protection against specific kinds of attack, and can be configured for different levels according to the required degree of protection. These features are designed to work together or independently. They can be also integrated with the appropriate software to create defensive layers. In addition, the chip includes a general-purpose accelerator that enhances the performance of selected industry-standard cryptographic algorithms.

The security features include:

- Secure High-Assurance Boot
  - Security library embedded in the immutable on-chip ROM
  - Authenticated boot, which protects against unauthorized software
    - Verification of the code signature during boot
    - RSA-1024/2048/3072/4096 keys anchored to the OTP fingerprint (SHA-256)
  - Encrypted boot which protects the software confidentiality

    - Runs every time the chip is reset
    - Image version control/image revocation (on-chip OTP-based)
- Secure storage
    - Off-chip storage protection using AES-128 and the chip's unique hardware-only key
- Hardware cryptographic accelerators
    - Symmetric: AES-128
    - Hash message digest: SHA-1, SHA-256
- Standalone True Random Number Generator (SA-TRNG) to act as entropy generator
- Secure debugging
    - Configurable protection against unauthorized JTAG manipulation
    - Three security levels + a complete JTAG disable
    - Support for JTAG port secure reopening for field return debugging
- Electrical fuses (OTP Memory)
- Hardware bus encryption
    - AES-128 encryption, supporting ECB and CTR modes
    - Non-secured access filtering

7.3 High-Assurance Boot (HAB)

The HAB, which is the high-assurance boot feature in the system boot ROM, detects and prevents the execution of unauthorized software (malware) during the boot sequence.

When the unauthorized software is permitted to gain control of the boot sequence, it can be used for a variety of goals, such as exposing stored secrets, circumventing access controls to sensitive data, services, or networks, or for re-purposing the platform. The unauthorized software can enter the platform during upgrades or re-provisioning, or when booting from the USB connections or removable devices.

The HAB protects against unauthorized software by:

- Using digital signatures to recognize the authentic software. This enables you to boot the device to a known initial state and run the software signed by the device manufacturer.
- Using code encryption to protect the confidential software during off-chip storage. When activated, the HAB decrypts the software loaded into the RAM before execution.

7.3.1 HAB process flow

The following figure shows the flow for creating and verifying digital signatures. The top half of this figure shows the signing process, which is performed off-chip. The bottom half shows the verification process performed on-chip during every system boot.

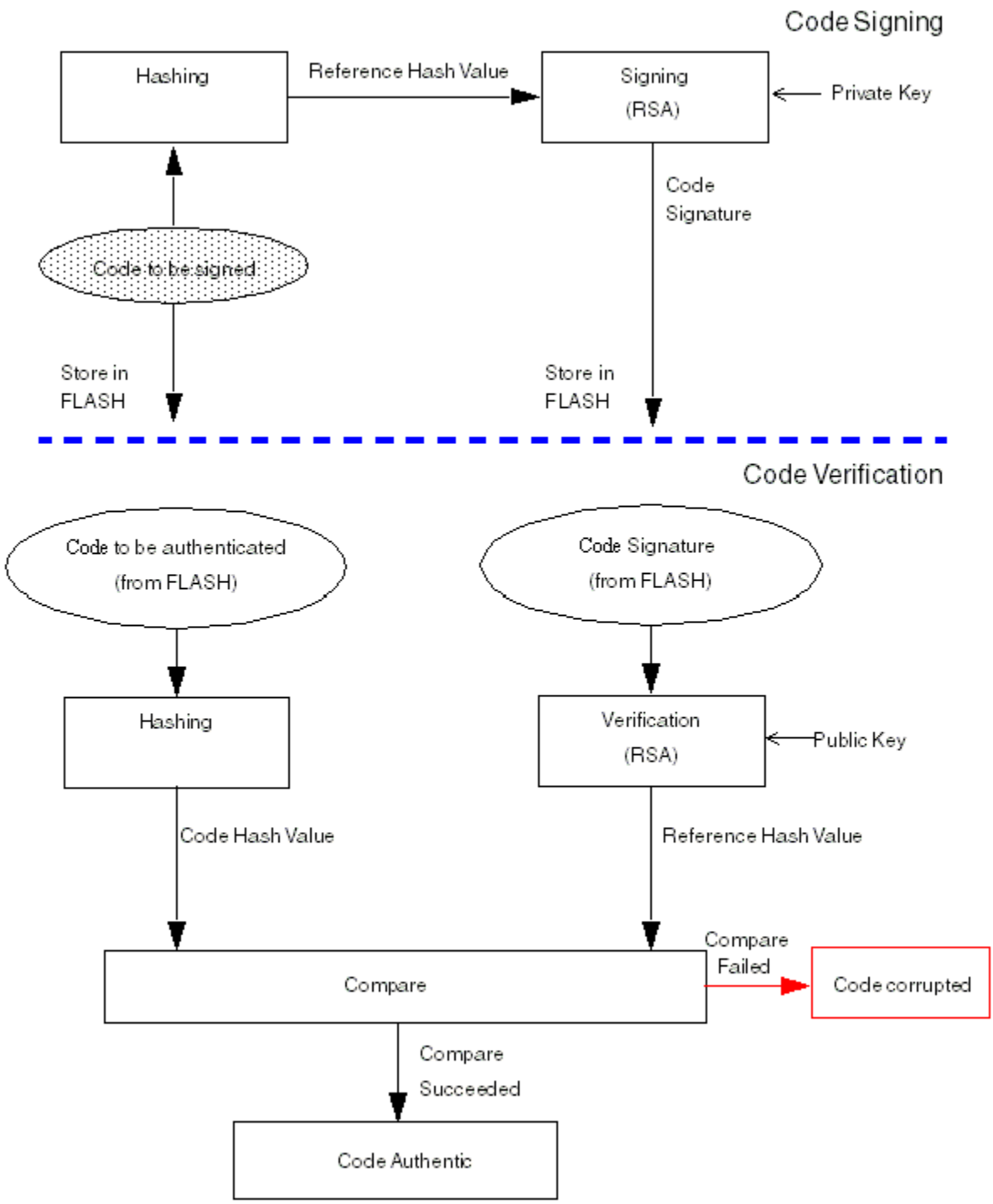


Figure 7-2. Code signing and authentication processes

The original software is programmed into the flash memory (or any other boot device) along with the signature. The HAB uses a public key to recover the reference hash value from the signature; it then compares the reference hash value to the current hash value calculated from the software in the flash. If the contents of the flash are modified either intentionally or unintentionally, the two hash values do not match and the verification fails.

7.3.2 HAB feature summary

The HAB features:

- Enforced internal boot via on-chip masked ROM
- Authentication of software loaded from any boot device (including the USB download)
- Authenticated decryption of software loaded from any boot device (including USB download) using the AES keys (128-bit)
- CMS PKCS#1 signature verification using RSA public key (1024, 2048, 3072 or 4096) and the SHA256 hash algorithm
- Public Key Infrastructure (PKI) support using X.509v3 certificates
- Root public key fingerprint in the manufacturer-programmable on-chip fuses
- Multiple root public keys with revocation by fuses
- Initialization of other security components
- Supports fall-through USB downloader if the primary or recovery boot fails
- Open configuration for development purposes and non-secure platforms
- Closed configuration for shipping secure platforms

On the chip, the HAB is integrated with these security features:

- The HAB initializes the SNVS security monitor state machine. A successful secure boot with the HAB is required for the platform software to gain access to use the DCP master secret key selected by SNVS.
- The HAB reads the root public key fingerprint, revocation mask, and security configuration from the OCOTP_CTRL.
- The HAB initializes the CSU.
- The HAB can use the DCP to accelerate hash calculations.

7.4 Secure Non-Volatile Storage (SNVS) module

- Provides a non-volatile real-time clock maintained by a coin-cell battery during system power down for use in both the secure and non-secure platforms.

- Protects the secure real-time clock against rollback attacks in time-sensitive protocols such as DRM and PKI
- Deters replay attacks in time-independent protocols such as certificate or firmware revocations
- Handles security violation detection and reporting, to defend sensitive data and operations against compromise, both at run-time and during system power-down
- Controls the access to the OTP master secret key used by the DCP to protect confidential data in the off-chip storage
- Provides non-volatile highly protected storage for an alternative master secret key

7.4.1 SNVS architecture

The SNVS is partitioned into two sections: a low-power part (SNVS_LP) and a high-power part (SNVS_HP).

The SNVS_LP block is in the always-powered-up domain. It is isolated from the rest of the logic by isolation cells which are library-instantiated cells that insure that the powered-up logic is not corrupted when the power goes down in the rest of the chip.

The SNVS_LP has these functional units:

- Zeroizable Master Key
- Secure non-rollover real-time counter with alarm
- Non-rollover monotonic counter
- Digital Low-Voltage Detector (LVD)
- General-purpose register
- Control and status registers

The SNVS_HP is in the chip power-supply domain. The SNVS_HP provides an interface between the SNVS_LP and the rest of the system. The access to the SNVS_LP registers can be gained through the SNVS_HP only when it is powered up according to the access permission policy.

The SNVS_HP has these functional units:

- IP bus interface
- SNVS_LP interface
- System Security Monitor (SSM)
- Zeroizable Master Key programming mechanism
- Master Key control block
- Non-secure real-time counter with alarm
- Control and status registers

7.5 Data Co-Processor (DCP)

For security purposes, the Data Co-Processor (DCP) provides hardware acceleration for the cryptographic algorithms. The features of DCP include:

- Encryption Algorithms: AES-128 (ECB and CBC modes)
- Hashing Algorithms: SHA-1 and SHA-256
- CRC-32
- Key selection from the SNVS, DCP internal key storage, or general memory
- Internal Memory for storing up to four AES-128 keys—when a key is written to a key slot it can be read only by the DCP AES-128 engine
- IP slave interface
- DMA

7.6 Standalone True Random Number Generator (TRNG)

The SA-TRNG is hardware accelerator module that generates a 512-bit entropy as needed by an entropy consuming module or by other post processing functions. A typical entropy consumer is a pseudo random number generator (PRNG) which can be implemented to achieve both true randomness and cryptographic strength random numbers using the SA-TRNG output as its entropy seed. The PRNG is not part of this module.

7.7 On-Chip OTP Controller (OCOTP_CTRL)

The OCOTP_CTRL provides the primary user-visible mechanism for interfacing with the on-chip fuses. These fuses' uses include:

- Unique chip identifiers
- Mask revision numbers
- Cryptographic keys
- Security configuration
- Boot characteristics
- Various control signals requiring permanent non-volatility

The OCOTP_CTRL provides:

- Shadow cache of fuse values, loaded at reset, before the system boot
- Ability to read and override the fuse values in the shadow cache (does not affect the fuse element)

- Ability to read the fuses directly (ignoring the shadow cache)
- Ability to write (program) the fuses by software or JTAG
- Fuses and shadow cache bits enforce read-protect, override-protect, and write-protect
- Lock fuses for selected fuse fields
- Scan protection
- Volatile software-accessible signals which can be used for software control of hardware elements (not requiring non-volatility).

7.8 Central Security Unit (CSU)

Central Security Unit (CSU) sets access control policies between the bus masters and bus slaves, enabling the peripherals to be separated into distinct security domains. This protects against the indirect unauthorized access to data which occurs when the software programs a DMA bus master to access addresses that the software itself is prohibited from accessing directly. Configuring the DMA bus master privileges in the CSU consistently with the software privileges defends against such indirect unauthorized access.

The CSU provides:

- Configuration of peripheral access permissions for peripherals that are unable to control their own access permissions
- Configuration of bus master privileges for bus masters that are unable to control their own privileges
- Optional locking of the individual CSU settings until the next power-on reset

7.9 System JTAG Controller (SJC)

The JTAG port provides debug access to hardware blocks, including the Arm processor and the system bus. This enables program control and manipulation as well as visibility to the chip peripherals and memory.

The JTAG port must be accessible during initial platform development, manufacturing tests, and general troubleshooting. Given its capabilities, JTAG manipulation is a known attack vector for accessing sensitive data and gaining control over software execution. The System JTAG Controller (SJC) protects against the whole range of attacks based on unauthorized JTAG manipulation. It also provides a JTAG port that conforms to the IEEE 1149.1 and IEEE 1149.6 (AC) standards for BSR (boundary-scan) testing.

The SJC provides these security levels:

- The JTAG Disabled-JTAG use is permanently blocked.
- The No-Debug-All security sensitive JTAG features are permanently blocked.
- The Secure JTAG-JTAG use is restricted (as in the No-Debug level) unless a secret-key challenge/response protocol is successfully executed.
- The JTAG Enabled-JTAG use is unrestricted.

The security levels are selected via the eFuse configuration.

7.9.1 Scan protection

The chip includes further scan protection logic for those SJC modes where the JTAG use is allowed. This ensures that the access to critical security values is protected as follows:

- The chip is reset when entering the scan mode.
- All modules are reset two clock cycles before receiving the scan-enable indication.
- The chip cannot exit the scan mode without a reset.
- The security modules (including SNVS, CSU, and OCOTP_CTRL) have an additional scan-protection logic to protect the sensitive internal data and functionality.

See the "System JTAG Controller (SJC)" chapter in the Security Reference Manual for more information on the SJC.

7.10 Bus Encryption Engine (BEE)

The Bus Encryption Engine (BEE) is implemented as an on-the-fly decryption engine, which is used for decrypting cypher context of FlexSPI. The main features of the BEE are:

- Standard AXI interconnection
- On-the-fly AES-128 decryption, supporting ECB and CTR modes
- Aliased memory space support. Address remapping for up to two individual regions
- Independent AES Key management for those two individual regions
- Bus access pattern optimization with the aid of the local store and forward buffer
- Non-secured access filtering based on the security label of the access
- Illegal access check and filtering

See the "Bus Encryption Engine (BEE)" chapter in the Security Reference Manual for more information on BEE.

Chapter 8
System Debug

8.1 Overview

This section describes the hardware and software debug and application development features and resources of the chip. It describes the following:

- Core/platform-specific resources
- Chip-wide resources
- Interface to the external debug and development tools

The debug and trace architecture is designed around the following:

- Arm Cortex M7 debug and trace components, adapted to SoC (for core debug)
- JTAG port used to interact with core under the debug by means of SJC, the system JTAG controller port
- DAP, the debug access port that supports the interface to the Arm RealView Debugging tools and other third-party tools
- TPIU, a trace port interface unit that efficiently accesses the program trace information from the system

8.2 Chip and Arm Platform Debug Architecture

The Arm Debug architecture is based on the Cortex M7 debug and trace components by Arm.

Its architecture maintains the traditional requirements of debug and trace:

- To access the debug functionality without software interaction
- To connect to a running system without performing a reset

Full access to the processor debug capability is available by the Arm debug register map through the Advanced Peripheral Bus (APB) slave port. The core includes a Processor Debug Unit which stops program execution, examines and alters the processor and coprocessor state, examines and alters the memory and input/output peripheral state, and restarts the processor core.

8.2.1 Debug Features

- EmbeddedICE-RT logic
  - Support for both the monitor-mode and halt-mode debugging:
  - Core run/halt control, debug status/control
  - Breakpoint/watchpoint control
  - Core-mapped and memory-mapped resource examination/modification
- Data communication channel between the Arm core and the host debugger via JTAG or SWD and the Debug Access Port (DAP) module

8.2.2 Debug system components

The Arm Cortex M7 debug and trace components include:

- Cortex-M7 DAP (CM7DAP)
- ETM (Embedded Trace Macrocell) supporting instruction trace
- ITM (Instrumentation Trace Macrocell)
- Cortex-M7 TPIU (Trace Port Interface)
- Cross Triggering logic for event routing (CTIs)
- Timestamp Generator (TSGEN)

Other related IPs and functionality:

- Flash Patch and Breakpoint unit (FPB)

8.2.2.1 Cortex-M7 DAP (CM7DAP)

The Cortex-M7 DAP (CM7DAP) is a component that provides an interface to allow off-chip debug tools to access the Cortex-M7 processor. It is a low gate-count DAP implementation. The key features of the CM7DAP are:

- Supports JTAG and Serial Wire Debug (SWD) Data Link protocols. Protocol is selected by fuse at reset

- Includes an AHB-AP, intended to be directly connected to the Cortex-M7 processor AHBD slave port
- Implements the Minimal Debug Port programmers model, see the ARM® Debug Interface Architecture Specification ADIv5.0 to ADIv5.2

For more information, see Arm Cortex M7 Integration and Implementation Manual.

8.2.2.2 Cortex-M7 Trace Port Interface (TPIU)

The Cortex-M7 TPIU is a component that bridges between the on-chip trace data from the Embedded Trace Macrocell (ETM) and the Instrumentation Trace Macrocell (ITM), with separate IDs, to a data stream. The Cortex-M7 TPIU encapsulates IDs where required, and the data stream is then captured by an external Trace Port Analyzer (TPA).

The Cortex-M7 TPIU is a variant of the CoreSight SoC-400 TPIU that is specially designed for low-cost debug.

The output of the TPIU is connected via external pins (MPS of TRACEDATA (ARM_TRACEn), which can be 1, 2, or 4 bits). The corresponding IOMUXC need be configured to select ARM_TRACEn function first. The TPIU port size will be updated automatically according to the number of ARM_TRACEn selected by IOMUXC. The system may utilize double data rate pins to either use a lower clock speed than that of the 32-bit ATB interface, or use fewer than 4 data pins for the output, based on the ability of the technology used. Given the speed of the ATB, 4 data pins with double data rate is recommended, with the external interface running at half the speed of the ATB. The speed of the external interface is from TRACECLKIN (TRACE_CLK_ROOT). TRACECLK (ARM_TRACE_CLK) is equal to TRACECLKIN/2, and is divided in the TPIU to clock trace data at the trace capture unit.

For more information, see Arm Cortex M7 Integration and Implementation Manual.

8.2.2.3 Embedded Trace Macrocell (ETM)

Instruction trace, also known as ETM (Embedded Trace Macrocell) trace, is a continuously collected sequence of every executed instruction for a selected portion of the application. ETM generates trace packets and sends them to the trace bus. ETM does not actually output every address or instruction that the processor has reached or executed; it usually generates compressed information about the program flow and

outputs full addresses only if needed (for example, if a branch has taken place). Because the debugger knows the application code image, the debugger can then reconstruct the full instruction sequence from the trace data.

For more information about ETM, refer to *Arm® CoreSight™ ETM-M7 Revision r0p1 Technical Reference Manual*.

8.2.2.4 Instrumentation Trace Macrocell

The ITM (Instrumentation Trace Macrocell) generates trace information as packets. There are four sources that can generate packets. If multiple sources generate packets at the same time, the ITM arbitrates the order in which packets are output. The four sources in decreasing order of priority are:

- Synchronization: DWT provides periodic requests to make ITM generate synchronization packet. Trace capture hardware uses it to identify the alignment of packet bytes in the bit-stream.
- Software Trace: Application software can write console messages directly to ITM stimulus ports, and output them to the host as trace packets.
- Hardware trace: The DWT generates these packets, and the ITM outputs them.
- Timestamping: ITM can generate timestamp packets that are inserted in to the trace stream, to help the host debugger to find out the timing of events. Timestamps are generated relative to packets. The ITM receives a 64-bit counter to generate the timestamp.

Trace data from ITM will be forwarded to TPIU and streamed out via the trace port.

For more information about ITM, refer to *Armv7-M Architecture Reference Manual*.

8.2.2.5 Cross-Trigger Interface (CTI)

The Cross-Trigger Interface (CTI) component is provided by Arm. A brief description of the CTI is provided below.

The CTI enables the debug logic and ETM to interact with each other. The tables below show how the CTI triggers are connected to the processor and ETM. For more information, refer to Arm Cortex-M7 Processor Revision r1p2 Technical Reference Manual.

Table 8-1. Trigger Signals to CTI

Signal	Description	Connection	Acknowledge, handshake
CTITRIGIN[7]	ETM Event Output 3	ETM to CTI	Pulsed
CTITRIGIN[6]	ETM Event Output 2		
CTITRIGIN[5]	ETM Event Output 1		
CTITRIGIN[4]	ETM Event Output 0		
CTITRIGIN[3]	DWT Comparator Output 2	Processor to CTI	
CTITRIGIN[2]	DWT Comparator Output 1		
CTITRIGIN[1]	DWT Comparator Output 0		
CTITRIGIN[0]	Processor Halted		

Table 8-2. Trigger Signals from CTI

Signal	Description	Connection	Acknowledge, handshake
CTITRIGOUT[7]	Processor Restart	CTI to Processor	Processor Restarted
CTITRIGOUT[6]	ETM Event Input 3	CTI to ETM	Pulsed
CTITRIGOUT[5]	ETM Event Input 2		
CTITRIGOUT[4]	ETM Event Input 1		
CTITRIGOUT[3]	ETM Event Input 0		
CTITRIGOUT[2]	Interrupt Request 1	CTI to System	Acknowledged by writing to the CTINTACK register in ISR
CTITRIGOUT[1]	Interrupt Request 0		
CTITRIGOUT[0]	Processor Debug Request	CTI to Processor	Acknowledged by the debugger writing to the CTINTACK register

8.2.2.6 Timestamp Generator (TSGEN)

The timestamp generator generates the timestamp value that is distributed over the rest of the timestamp interconnect. For this chip the timestamp value is distributed to ITM and ETM. The pselread APB interface is disabled in this chip so the registers in PSELREAD region are not applicable.

For more information about TSGEN, refer to Arm CoreSight SoC-400 Revision: r3p2 Technical Reference Manual

8.2.3 Chip-Specific SJC Features

NOTE

The application note AN12419: Secure JTAG for i.MXRT10xx describes how the Secure JTAG on the i.MX RT10xx MCU family can be used.

8.2.3.1 JTAG Disable Mode

In addition to different JTAG security modes that are implemented internally in the JTAG Controller, there is an option to disable the SJC functionality by e-fuse configuration. This creates additional JTAG mode "JTAG Disabled" with highest level of JTAG protection. In this mode all JTAG features are disabled.

Specifically, the following debug features are disabled in addition to the features that were already disabled in "No Debug" JTAG mode:

- Non-Secure JTAG control registers (PLL configuration, Deterministic Reset, PLL bypass)
- Non-Secure JTAG status registers (Core status)
- Chip Identification Code (IDCODE)

8.2.3.2 JTAG ID

Table 8-3. i.MX JTAG ID

Device	Silicon revision	JTAG ID
i.MX RT1060	Rev 1.0	088C_501Dh

8.2.3.3 JTAG-to-SWD change sequence

1. Send more than 50 TCK cycles with TMS (SWD_IO) = 1
2. Send the 16-bit sequence on TMS (SWD_IO) = 0111_1001_1110_1111 (MSB transmitted first)
3. Send more than 50 TCK cycles with TMS (SWD_IO) = 1

NOTE

See the ARM documentation for the CoreSight DAP Lite for restrictions.

8.2.4 System JTAG controller main features

- IEEE P1149.1, 1149.6 (standard JTAG) interface to off-chip test and development equipment
  - Includes an SJC-only mode for true IEEE P1149.1 compliance, used primarily for board-level implementation of boundary scan.
  - Supports IEEE P1149.6 extensions to the JTAG standard for AC testing of selected I/O signals.
- Debug-related control and status; putting selected cores into reset and/or debug mode and monitoring individual core status signals by means of JTAG
- System status, such as the state of the PLLs (locked or not locked)
- levels of security, ranging from no security to no JTAG accessibility to the chip

NOTE

The security levels are detailed in the Security Reference Manual (SRM).

8.2.5 TAP port

The SJC supports the following standard JTAG pins:

- TRSTB
- TDI
- TDO
- TCK
- TMS

8.2.6 SJC main blocks

- Interface to the outside world via the standard JTAG pins
- Interface to the external Debug_Event pin
- A master TAP controller which implements the standard JTAG state machine
- Implementation of the mandatory and optional IEEE P1149.1 (JTAG) instructions
  - Mandatory: "EXTEST", "SAMPLE/PRELOAD", and "BYPASS"
  - Optional: "ID_CODE" (SOC JTAG ID register), "HIGHZ"
- The ExtraDebug registers, which implement a variety of control and status features
  - Three 32-bit insecure general purpose status registers
  - Two 32-bit secure status registers - one predefined, one general purpose.
  - Control and status registers for debug, core, charge pump, and PLL.
- Four levels of fuse-defined security, ranging from no security to no access.

Both predefined and user-defined control and status functions are supported by the SJC.

8.3 Miscellaneous

The Miscellaneous function described in this section provide useful general capabilities.

8.3.1 Clock/Reset/Power

CDBGPWRUPREQ and CDBGPWRUPACK are the handshake signals between the DAP and the clock control module to ensure debug power and clocks are turned on. If the debug components are always powered on, the handshake becomes a mechanism to turn debug clocks on.

The debug components can receive resets from the following sources:

- Debug Reset (CDBGRSTREQ bit within the SWJ-DP CTRL/STAT register of the DAP) in the TCLK domain. This allows the debug tools to reset the debug logic.
- System POR reset

Chapter 9
System Boot

9.1 Chip-specific Boot Information

This device has various peripherals supported by the ROM bootloader.

Table 9-1. ROM Bootloader Peripheral PinMux

Peripheral	Instance	Port (IO function)	PAD	Mode	Use
LPUART	1	LPUART1_TX	GPIO_AD_B0_12 pin 24	ALT2	Can be used for serial downloader mode. Refer to Serial Downloader for more information.
		LPUART1_RX	GPIO_AD_B0_13 pin 25	ALT2	
LPSPI	1	LPSPI1_SCK	GPIO_SD_B0_00	ALT4	Serial NOR/EEPROM connected to one of the LPSPI ports can be used as a recovery device. Refer to Recovery devices for more information. **Note:** recovery device boot is disabled by default. Fuses must be blown to enable and configure this option.
		LPSPI1_SDO	GPIO_SD_B0_02	ALT4	
		LPSPI1_SDI	GPIO_SD_B0_03	ALT4	
		LPSPI1_PCS0	GPIO_SD_B0_01	ALT4	
	2	LPSPI2_SCK	GPIO_SD_B1_07	ALT4	
		LPSPI2_SDO	GPIO_SD_B1_08	ALT4	
		LPSPI2_SDI	GPIO_SD_B1_09	ALT4	
		LPSPI2_PCS0	GPIO_SD_B1_06	ALT4	
	3	LPSPI3_SCK	GPIO_AD_B0_00	ALT7	
		LPSPI3_SDO	GPIO_AD_B0_01	ALT7	
		LPSPI3_SDI	GPIO_AD_B0_02 pin 1	ALT7	
		LPSPI3_PCS0	GPIO_AD_B0_03 pin	ALT7	
	4	LPSPI4_SCK	GPIO_B0_03 pin 13	ALT3	
		LPSPI4_SDO	GPIO_B0_02 pin 11	ALT3	
		LPSPI4_SDI	GPIO_B0_01 pin 12	ALT3	
		LPSPI4_PCS0	GPIO_B0_00 pin 10	ALT3	
SEMC NAND	N/A	SEMC_DATA00	GPIO_EMC_00	ALT0	Parallel NAND flash connected to the SEMC is a primary boot option. Refer to Parallel NAND flash Boot over SEMC for more information.
		SEMC_DATA01	GPIO_EMC_01	ALT0	
		SEMC_DATA02	GPIO_EMC_02	ALT0	
		SEMC_DATA03	GPIO_EMC_03	ALT0	
		SEMC_DATA04	GPIO_EMC_04 pin 2	ALT0	
		SEMC_DATA05	GPIO_EMC_05 pin 3	ALT0	

Table continues on the next page...

Table 9-1. ROM Bootloader Peripheral PinMux (continued)

Peripheral	Instance	Port (IO function)	PAD	Mode	Use
		SEMC_DATA06	GPIO_EMC_06 pin 4	ALT0	**Note:** By default SEMC_CSX0 is used as the chip select. Other chip selects can be used, but fuses must be blown to select these configurations (GPIO overrides cannot be used to configure this option).
		SEMC_DATA07	GPIO_EMC_07 pin 33	ALT0	
		SEMC_DATA08	GPIO_EMC_30	ALT0	
		SEMC_DATA09	GPIO_EMC_31 pin 29	ALT0	
		SEMC_DATA10	GPIO_EMC_32 pin 28	ALT0	
		SEMC_DATA11	GPIO_EMC_33	ALT0	
		SEMC_DATA12	GPIO_EMC_34	ALT0	
		SEMC_DATA13	GPIO_EMC_35	ALT0	
		SEMC_DATA14	GPIO_EMC_36 pin 31	ALT0	
		SEMC_DATA15	GPIO_EMC_37 pin 30	ALT0	
		SEMC_ADDR09	GPIO_EMC_18	ALT0	
		SEMC_ADDR11	GPIO_EMC_19	ALT0	
		SEMC_ADDR12	GPIO_EMC_20	ALT0	
		SEMC_BA1	GPIO_EMC_22	ALT0	
		SEMC_RDY	GPIO_EMC_40	ALT0	
		SEMC_CSX0	GPIO_EMC_41	ALT0	
		SEMC_CSX1	GPIO_B0_00 pin	ALT6	
		SEMC_CSX2	GPIO_B0_01 pin 12	ALT6	
		SEMC_CSX3	GPIO_B0_02 pin 11	ALT6	
		SEMC_ADDR08	GPIO_EMC_17	ALT0	
SEMC NOR	N/A	SEMC_DATA00	GPIO_EMC_00	ALT0	Parallel NOR flash connected to the SEMC is a primary boot option. Refer to Parallel NOR flash Boot over SEMC for more information. **Note:** By default SEMC_CSX0 is used as the chip select. Other chip selects can be used, but fuses must be blown to select these configurations (GPIO overrides cannot be used to configure this option).
		SEMC_DATA01	GPIO_EMC_01	ALT0	
		SEMC_DATA02	GPIO_EMC_02	ALT0	
		SEMC_DATA03	GPIO_EMC_03	ALT0	
		SEMC_DATA04	GPIO_EMC_04 pin 2	ALT0	
		SEMC_DATA05	GPIO_EMC_05 pin 3	ALT0	
		SEMC_DATA06	GPIO_EMC_06 pin 4	ALT0	
		SEMC_DATA07	GPIO_EMC_07 pin 33	ALT0	
		SEMC_DATA08	GPIO_EMC_30	ALT0	
		SEMC_DATA09	GPIO_EMC_31 pin 29	ALT0	
		SEMC_DATA10	GPIO_EMC_32 pin 28	ALT0	
		SEMC_DATA11	GPIO_EMC_33	ALT0	
		SEMC_DATA12	GPIO_EMC_34	ALT0	
		SEMC_DATA13	GPIO_EMC_35	ALT0	
		SEMC_DATA14	GPIO_EMC_36 pin 31	ALT0	
		SEMC_DATA15	GPIO_EMC_37 pin 30	ALT0	
		SEMC_ADDR00	GPIO_EMC_09	ALT0	
		SEMC_ADDR01	GPIO_EMC_10	ALT0	
		SEMC_ADDR02	GPIO_EMC_11	ALT0	

Table continues on the next page...

Table 9-1. ROM Bootloader Peripheral PinMux (continued)

Peripheral	Instance	Port (IO function)	PAD	Mode	Use
		SEMC_ADDR03	GPIO_EMC_12	ALT0	
		SEMC_ADDR04	GPIO_EMC_13	ALT0	
		SEMC_ADDR05	GPIO_EMC_14	ALT0	
		SEMC_ADDR06	GPIO_EMC_15	ALT0	
		SEMC_ADDR07	GPIO_EMC_16	ALT0	
		SEMC_ADDR11	GPIO_EMC_19	ALT0	
		SEMC_ADDR12	GPIO_EMC_20	ALT0	
		SEMC_BA0	GPIO_EMC_21	ALT0	
		SEMC_BA1	GPIO_EMC_22	ALT0	
		SEMC_CSX0	GPIO_EMC_41	ALT0	
		SEMC_CSX1	GPIO_B0_00 pin 10	ALT6	
		SEMC_CSX2	GPIO_B0_01 pin 12	ALT6	
		SEMC_CSX3	GPIO_B0_02 pin 11	ALT6	
		SEMC_ADDR08	GPIO_EMC_17	ALT0	
SD	1	USDHC1_CD_B	GPIO_SD_B1_12	ALT6	eMMC/MMC or SD/eSD connected to one of the USDHC ports is a primary boot option. Refer to Expansion device for more information on USDHC boot.
		USDHC1_VSELECT	GPIO_B1_14	ALT6	
		USDHC1_RESET_B	GPIO_B1_15	ALT6	
		USDHC1_CMD	GPIO_SD_B0_00	ALT0	
		USDHC1_CLK	GPIO_SD_B0_01	ALT0	
		USDHC1_DATA0	GPIO_SD_B0_02	ALT0	
		USDHC1_DATA1	GPIO_SD_B0_03	ALT0	
		USDHC1_DATA2	GPIO_SD_B0_04	ALT0	
		USDHC1_DATA3	GPIO_SD_B0_05	ALT0	
	2	USDHC2_RESET_B	GPIO_SD_B1_06	ALT0	
		USDHC2_CMD	GPIO_SD_B1_05	ALT0	
		USDHC2_CLK	GPIO_SD_B1_04	ALT0	
		USDHC2_DATA0	GPIO_SD_B1_03	ALT0	
		USDHC2_DATA1	GPIO_SD_B1_02	ALT0	
		USDHC2_DATA2	GPIO_SD_B1_01	ALT0	
		USDHC2_DATA3	GPIO_SD_B1_00	ALT0	
		USDHC2_DATA4	GPIO_SD_B1_08	ALT0	
		USDHC2_DATA5	GPIO_SD_B1_09	ALT0	
		USDHC2_DATA6	GPIO_SD_B1_10	ALT0	
		USDHC2_DATA7	GPIO_SD_B1_11	ALT0	
FlexSPI NOR Flash - QSPI	1	*FLEXSPI_B_DATA3*	GPIO_SD_B1_00	ALT1	QSPI memory attached to FlexSPI is a primary boot option. Refer to Serial NOR Flash Boot via FlexSPI for more information. The ROM will read the 512-byte FlexSPI NOR configuration parameters
		FLEXSPI_B_DATA2	GPIO_SD_B1_01	ALT1	
		FLEXSPI_B_DATA1	GPIO_SD_B1_02	ALT1	
		FLEXSPI_B_DATA0	GPIO_SD_B1_03	ALT1	
		FLEXSPI_B_SCLK	GPIO_SD_B1_01	ALT1	

Table continues on the next page...

Table 9-1. ROM Bootloader Peripheral PinMux (continued)

Peripheral	Instance	Port (IO function)	PAD	Mode	Use
		FLEXSPI_B_DQS	GPIO_SD_B0_05	ALT4	described in FlexSPI Serial NOR Flash Boot Operation using the non-italicized pins. **Note:** ROM can configure the italicized signals based on the FlexSPI NOR configuration parameters provided.
		FLEXSPI_B_SS0_B	GPIO_SD_B0_04	ALT4	
		FLEXSPI_B_SS1_B	GPIO_SD_B0_01	ALT6	
		FLEXSPI_A_DQS	GPIO_SD_B1_05	ALT1	
		FLEXSPI_A_SS0_B	GPIO_SD_B1_06	ALT1	
		FLEXSPI_A_SS1_B	GPIO_SD_B0_00	ALT6	
		FLEXSPI_A_SCLK	GPIO_SD_B1_07	ALT1	
		FLEXSPI_A_DATA0	GPIO_SD_B1_08	ALT1	
		FLEXSPI_A_DATA1	GPIO_SD_B1_09	ALT1	
		FLEXSPI_A_DATA2	GPIO_SD_B1_10	ALT1	
		FLEXSPI_A_DATA3	GPIO_SD_B1_11	ALT1	
FlexSPI NOR - QSPI - 2nd Option	1	FLEXSPI_A_SS0_B	GPIO_AD_B1_15 pin 27	ALT0	QSPI memory attached to FlexSPI is a primary boot option. Refer to Serial NOR Flash Boot via FlexSPI for more information. The ROM will read the 512-byte FlexSPI NOR configuration parameters described in FlexSPI Serial NOR Flash Boot Operation using the non-italicized pins. **Note:** These pins are a secondary pinout option for FlexSPI serial NOR flash boot.
		FLEXSPI_A_SCLK	GPIO_AD_B1_14 pin 26	ALT0	
		FLEXSPI_A_DQS	GPIO_AD_B1_09 pin 23	ALT0	
		FLEXSPI_A_DATA0	GPIO_AD_B1_13 pin 39 (T4.1)	ALT0	
		FLEXSPI_A_DATA1	GPIO_AD_B1_12 pin 38 (T4.1)	ALT0	
		FLEXSPI_A_DATA2	GPIO_AD_B1_11 pin 21	ALT0	
		FLEXSPI_A_DATA3	GPIO_AD_B1_10 pin	ALT0	
FlexSPI NOR Flash - Octal	1	FLEXSPI_B_DATA3	GPIO_SD_B1_00	ALT1	Octal serial NOR flash memory attached to FlexSPI is a primary boot option. Refer to Serial NOR Flash Boot via FlexSPI for more information. The ROM will read the 512-byte FlexSPI NOR configuration parameters described in FlexSPI Serial NOR Flash Boot Operation using the non-italicized pins. For 8-bit wide memories the FLEXSPI_B_DATA[3:0] pins are combined with the FLEXSPI_A_DATA[3:0] lines to get the full 8-bit port. **Note:** ROM can configure the italicized signals based on the FlexSPI NOR configuration parameters provided.
		FLEXSPI_B_DATA2	GPIO_SD_B1_01	ALT1	
		FLEXSPI_B_DATA1	GPIO_SD_B1_02	ALT1	
		FLEXSPI_B_DATA0	GPIO_SD_B1_03	ALT1	
		FLEXSPI_B_SCLK	GPIO_SD_B1_01	ALT1	
		FLEXSPI_B_DQS	GPIO_SD_B0_05	ALT4	
		FLEXSPI_B_SS0_B	GPIO_SD_B0_04	ALT4	
		FLEXSPI_B_SS1_B	GPIO_SD_B0_01	ALT6	
		FLEXSPI_A_DQS	GPIO_SD_B1_05	ALT1	
		FLEXSPI_A_SS0_B	GPIO_SD_B1_06	ALT1	
		FLEXSPI_A_SS1_B	GPIO_SD_B0_00	ALT6	
		FLEXSPI_A_SCLK	GPIO_SD_B1_07	ALT1	
		FLEXSPI_A_DATA0	GPIO_SD_B1_08	ALT1	
		FLEXSPI_A_DATA1	GPIO_SD_B1_09	ALT1	
		FLEXSPI_A_DATA2	GPIO_SD_B1_10	ALT1	
		FLEXSPI_A_DATA3	GPIO_SD_B1_11	ALT1	
FlexSPI NAND Flash	1	FLEXSPI_A_DQS	GPIO_SD_B1_05	ALT1	Serial NAND memory attached to FlexSPI is a primary boot option. Refer to Serial NAND Flash Boot over FlexSPI for more information.
		FLEXSPI_A_SS0_B	GPIO_SD_B1_06	ALT1	
		FLEXSPI_A_SCLK	GPIO_SD_B1_07	ALT1	

Table continues on the next page...

Table 9-1. ROM Bootloader Peripheral PinMux (continued)

Peripheral	Instance	Port (IO function)	PAD	Mode	Use
		FLEXSPI_A_DATA0	GPIO_SD_B1_08	ALT1	
		FLEXSPI_A_DATA1	GPIO_SD_B1_09	ALT1	
		FLEXSPI_A_DATA2	GPIO_SD_B1_10	ALT1	
		FLEXSPI_A_DATA3	GPIO_SD_B1_11	ALT1	
FlexSPI RESET		GPIO1_IO29	GPIO_AD_B1_13 pin 39 (T4.1)	ALT5	

NOTE

ROM does not support boot from FLEXSPI_B port directly. ROM always seeks a valid Flash Configuration Block from the FLEXSPI_A port and then re-configures the FLEXSPI controller using the valid parameters in the block read-out. This reconfiguration can include, but is not limited to, FLEXSPI_B port support.

9.2 Overview

The boot process begins at any Reset where the hardware reset logic forces the Arm core to begin execution starting from the on-chip boot ROM.

The boot ROM code uses the state of the internal register BOOT_MODE[1:0] as well as the state of various eFuses and/or GPIO settings to determine the boot flow behavior of the device.

The main features of the ROM include:

- Support for booting from various boot devices
- Serial downloader support (USBOTG and UART)
- Device Configuration Data (DCD) and plugin
- Digital signature and encryption based High-Assurance Boot (HAB)
- Wake-up from the low-power modes
- Encrypted execute-in-place (XIP) on Serial NOR via FlexSPI interface powered by:
  - Bus Encryption Engine (BEE)
  - Data Co-Processor (DCP)

The boot ROM supports boot device as below:

- Serial NOR Flash via FlexSPI
- Serial NAND Flash via FlexSPI
- Parallel NOR Flash via SEMC
- RAWNAND Flash via SEMC

- SD/MMC via uSDHC
- SPI NOR/EEPROM via LPSPI

The boot ROM uses the state of the BOOT_MODE and eFuses to determine the boot device. For development purposes, the eFuses used to determine the boot device may be overridden using the GPIO pin inputs.

The boot ROM code also allows the downloading of programs to be run on the device. An example is a provisioning program that can make further use of the serial connection to provide a boot device with a new image. Typically, the provisioning program is downloaded to the internal RAM and allows to program the boot devices, such as the FlexSPI NOR flash. The ROM serial downloader uses a high-speed USB in a non-stream mode connection.

The boot ROM allows waking up from the low-power modes. On reset, the ROM checks the power gating status register. When waking from the low-power mode, the core skips loading an image from the boot device and jumps to the address saved in PERSISTENT_ENTRY0.

The Device Configuration Data (DCD) feature allows the boot ROM code to obtain the SOC configuration data from an external program image residing on the boot device. As an example, the DCD can be used to program the SDRAM controller for optimal settings improving the boot performance. The DCD is restricted to the memory areas and peripheral addresses that are considered essential for the boot purposes (see Write data command).

A key feature of the boot ROM is the ability to perform a secure boot, also known as a High-Assurance Boot (HAB). This is supported by the HAB security library which is a subcomponent of the ROM code. The HAB uses a combination of hardware and software together with the Public Key Infrastructure (PKI) protocol to protect the system from executing unauthorized program images. Before the HAB allows the user image to execute, the image must be signed. The signing process is done during the image build process by the private key holder and the signatures are then included as a part of the final program image. If configured to do so, the ROM verifies the signatures using the public keys included in the program image. In addition to supporting the digital signature verification to authenticate the program images, encrypted boot is also supported. The encrypted boot can be used to prevent the cloning of the program image directly off the boot device. A secure boot with HAB can be performed on all boot devices supported on the chip in addition to the serial downloader. The HAB library in the boot ROM also provides the API functions, allowing the additional boot chain components (e.g., bootloader, application) to extend the secure boot chain. The out-of-fab setting for the SEC_CONFIG is the open configuration, in which the ROM/HAB performs the image authentication, but all authentication errors are ignored and the image is still allowed to execute.

9.3 Boot modes

During reset, the chip checks the power gating controller status register.

During boot, the ROM's behavior is defined by the boot mode pin settings, as described in Boot mode pin settings. When waking up from the low-power boot mode, the core skips the clock settings. The boot ROM checks that the PERSISTENT_ENTRY0 (see Persistent bits) is a pointer to a valid address space (OCRAM). If the PERSISTENT_ENTRY0 is a pointer to a valid range, it starts the execution using the entry point from the PERSISTENT_ENTRY0 register. If the PERSISTENT_ENTRY0 is a pointer to an invalid range, the core performs the system reset.

9.3.1 Boot mode pin settings

The device has four boot modes (one is reserved for NXP use). The boot mode is selected based on the binary value stored in the internal BOOT_MODE register.

The BOOT_MODE register is initialized by sampling the BOOT_MODE0 and BOOT_MODE1 inputs on the rising edge of the POR_B. After these inputs are sampled, their subsequent state does not affect the contents of the internal BOOT_MODE register. The state of the internal BOOT_MODE register may be read from the BMOD[1:0] field of the SRC Boot Mode Register (SRC_SBMR2). The available boot modes are: Boot From Fuses, Serial Downloader, and Internal Boot. See this table for settings:

Table 9-2. Boot MODE pin settings

BOOT_MODE[1:0]	Boot Type
00	Boot From Fuses
01	Serial Downloader
10	Internal Boot
11	Reserved

9.3.2 High-level boot sequence

The figure found here show the high-level boot ROM code flow.

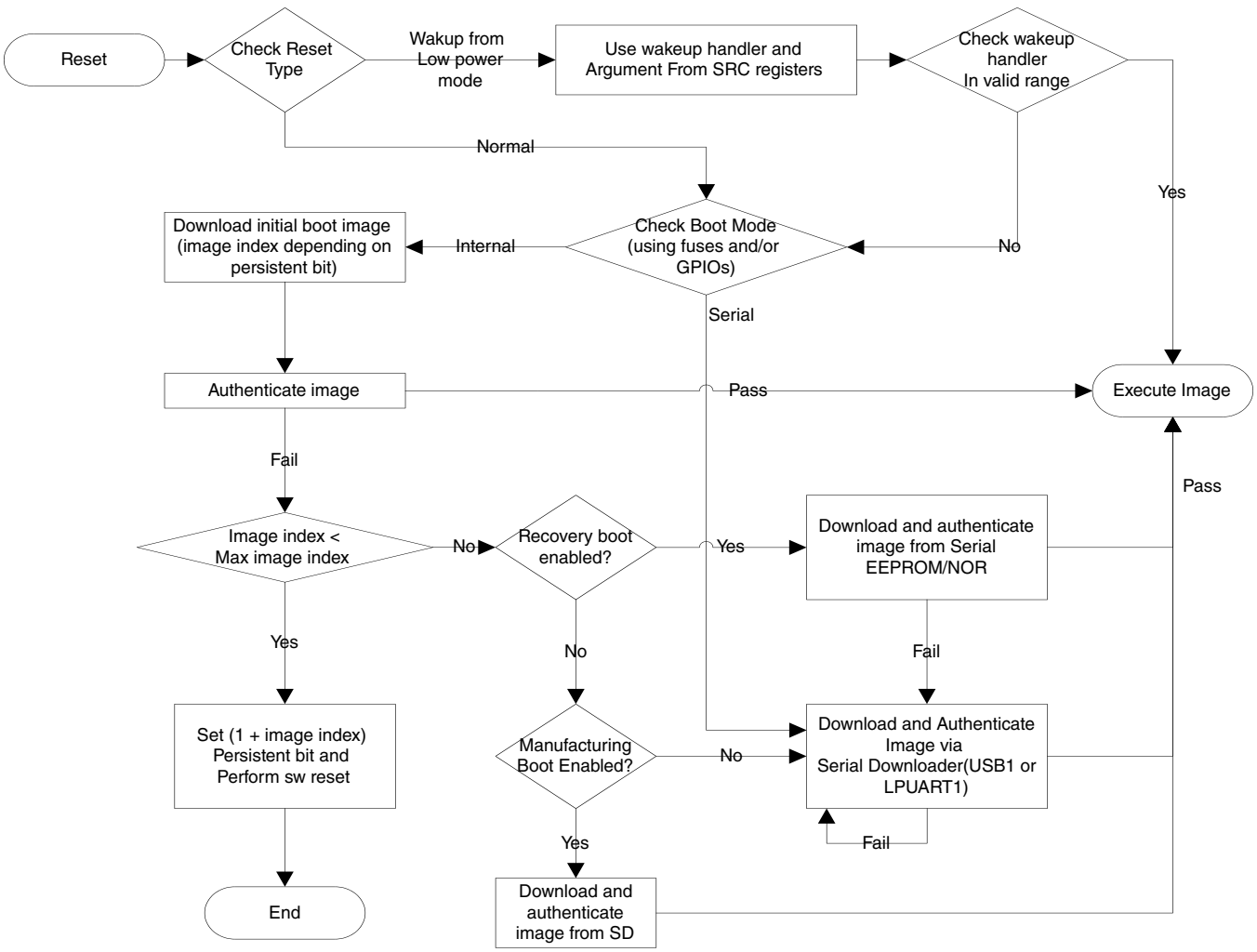


Figure 9-1. Boot flow

9.3.3 Boot From Fuses mode (BOOT_MODE[1:0] = 00b)

A value of 00b in the BOOT_MODE[1:0] register selects the Boot From Fuses mode.

This mode is similar to the Internal Boot mode described in Internal Boot mode (BOOT_MODE[1:0] = 10b) with one difference. In this mode, the override from the GPIO boot pins are ignored. The boot ROM code uses the boot eFuse settings only. This mode also supports a secure boot using HAB.

If set to Boot From Fuses, the boot flow is controlled by the BT_FUSE_SEL eFuse value. If BT_FUSE_SEL = 0, indicating that the boot device (for example, flash) was not programmed yet, the boot flow jumps directly to the Serial Downloader. If BT_FUSE_SEL = 1, the normal boot flow is followed, where the ROM attempts to boot from the selected boot device. BT_FUSE_SEL = 1 is programmed on every Teensy board

The first time a board is used, the default eFuses may be configured incorrectly for the hardware on the platform. In such case, the Boot ROM code may try to boot from a device that does not exist. This may cause an electrical/logic violation on some pads. Using the Boot From Fuses mode addresses this problem.

Setting the BT_FUSE_SEL=0 forces the ROM code to jump directly to the Serial Downloader. This allows a bootloader to be downloaded which can then provision the boot device with a program image and blow the BT_FUSE_SEL and the other boot configuration eFuses. After the reset, the boot ROM code determines that the BT_FUSE_SEL is blown (BT_FUSE_SEL = 1) and the ROM code performs an internal boot according to the new eFUSE settings. This allows the user to set BOOT_MODE[1:0]=00b on a production device and burn the fuses on the same device (by forcing the entry to the Serial Downloader), without changing the value of the BOOT_MODE[1:0] or the pullups/pulldowns on the BOOT_MODE pins.

9.3.4 Serial Downloader (BOOT_MODE[1:0] = 01b)

The Serial Downloader provides a means to download a Program Image to the on-chip RAM over USB or UART serial connection. In this mode, typically a host PC can communicate to the ROM bootloader using serial download protocol. Serial downloader and the protocol are discussed in Serial Downloader.

9.3.5 Internal Boot mode (BOOT_MODE[1:0] = 10b)

A value of 10b in the BOOT_MODE[1:0] register selects the Internal Boot mode. In this mode, the processor continues to execute the boot code from the internal boot ROM.

The boot code performs the hardware initialization, loads the program image from the chosen boot device, performs the image validation using the HAB library (see Boot security settings), and then jumps to an address derived from the program image. If an error occurs during the internal boot, the boot code jumps to the Serial Downloader (see Serial Downloader (BOOT_MODE[1:0] = 01b)). A secure boot using the HAB is possible in all the three boot modes.

When set to the Internal Boot, the boot flow may be controlled by a combination of eFuse settings with an option of overriding the fuse settings using the General Purpose I/O (GPIO) pins. The GPIO Boot Select FUSE (BT_FUSE_SEL) determines whether the ROM uses the GPIO pins for a selected number of configuration parameters or eFuses in this mode. BT_FUSE_SEL = 1 is programmed on every Teensy board

- If BT_FUSE_SEL = 1, all boot options are controlled by the eFuses described in Boot eFuse descriptions.
- If BT_FUSE_SEL = 0, the specific boot configuration parameters may be set using the GPIO pins rather than eFuses. The fuses that can be overridden when in this mode are indicated in the GPIO column of Boot eFuse descriptions. GPIO boot overrides provides the details of the GPIO pins.

The use of the GPIO overrides is intended for development since these pads are used for other purposes in the deployed products. NXP recommends controlling the boot configuration by the eFuses in the deployed products and reserving the use of the GPIO mode for the development and testing purposes only.

9.3.6 Boot security settings

The internal boot modes use one of three security configurations.

- Closed: This level is intended for use with shipping-secure products. All HAB functions are executed and the security hardware is initialized (the Security Controller or SNVS enters the Secure state), the DCD is processed if present, and the program image is authenticated by the HAB before its execution. All detected errors are logged, and the boot flow is aborted with the control being passed to the serial downloader. At this level, the execution does not leave the internal ROM unless the target executable image is authenticated.
- Open: This level is intended for use in non-secure products or during the development phases of a secure product. All HAB functions are executed in the same way as for a closed system. The security hardware is initialized (except for the SNVS which is left in the Non-Secure state), the DCD is processed if present, and the program image is authenticated by the HAB before its execution. All detected errors are logged, but have no influence on the boot flow which continues as if the errors did not occur. This configuration is useful for a secure product development because the program image runs even if the authentication data is missing or incorrect, and the error log can be examined to determine the cause of the authentication failure.
- Field Return: This level is intended for the parts returned from the shipped products.

9.4 Device configuration

This section describes the external inputs that control the behavior of the Boot ROM code.

This includes the boot device selection (FlexSPI NOR FlexSPI NAND, Parallel NOR,SD, MMC, and so on), boot device configuration (SD bus width, speed, and so on), and other. In general, the source for this configuration comes from the eFuses embedded inside the chip. However, certain configuration parameters can be sourced from the GPIO pins, allowing further flexibility during the development process.

9.4.1 Boot eFuse descriptions

This table is a comprehensive list of the configuration parameters that the ROM uses.

Table 9-3. Boot eFuse descriptions

Fuse	Configuration	Definition	GPIO[1]	Shipped value	Settings[2]
BT_FUSE_SEL BT_FUSE_SEL = 1 is programmed on every Teensy board	OEM	In the Internal Boot mode BOOT_MODE[1:0] = 10, the BT_FUSE_SEL fuse determines whether the boot settings indicated by a Yes in the GPIO column are controlled by the GPIO pins or the eFuse settings in the On-Chip OTP Controller (OCOTP). In the Boot From Fuse mode BOOT_MODE[1:0] = 00, the BT_FUSE_SEL fuse indicates whether the bit configuration eFuses are programmed.	NA	0	If BOOT_MODE[1:0] = 10b: • 0 - The bits of the SBMR are overridden by the GPIO pins. • 1 - The specific bits of the SBMR are controlled by the eFuse settings. If BOOT_MODE[1:0] = 00b: • 0—The BOOT configuration eFuses are not programmed yet. The boot flow jumps to the serial downloader. • 1—The BOOT configuration eFuses are programmed. The regular boot flow is performed.
FIELD_RETURN	OEM	Enables the NXP reserved modes.			0—The NXP reserved modes are disabled. 1—The NXP reserved modes are enabled.
UNIQUE_ID[63:0]	NXP	Device Unique ID, 64-bit UID	NA	Unique ID	Settings vary—used by HAB
BOOT_CFG1[7:0]	OEM	Boot configuration 1	Yes	0	Specific to the selected boot mode
BOOT_CFG2[2:0]	OEM	Boot configuration 2	Yes	0	Specific to the selected boot mode
LPB_BOOT	OEM	Low-Power Boot	No	0	Divide (Core / Bus) based on Boot frequencies: 00 - Div by 1

Table continues on the next page...

Table 9-3. Boot eFuse descriptions (continued)

Fuse	Configuration	Definition	GPIO[1]	Shipped value	Settings[2]
					01 - Div by 2 10 - Div by 4 11 - Div by 8
WDOG_ENABLE	OEM	Watchdog reset counter enable	No	0	WDOG1 enable: 0—The watchdog reset counter is disabled during the serial downloader. 1—The watchdog reset counter is enabled during the serial downloader.
PAD_SETTINGS	OEM	Override values for the SD/MMC and NAND boot modes	No	0	Override these IO PAD settings: • PAD_SETTINGS[0]—Slew Rate • PAD_SETTINGS[3:1]—Drive Strength • PAD_SETTINGS[5:4]—Speed Settings

1. This setting is overridden by the GPIO settings when the BT_FUSE_SEL fuse is intact. See GPIO Boot Overrides for the corresponding GPIO pin.
2. 0 = intact fuse and 1= blown fuse

9.4.2 GPIO boot overrides

This table provides a list of the GPIO boot overrides:

BOOT_CFG1 and BOOT_CFG2 pins are not used with Teensy because BT_FUSE_SEL = 1 is programmed

Table 9-4. GPIO Boot Overrides

Package Pin	Direction on reset	eFuse
GPIO_AD_B0_04/BOOT_MODE0	Input	Boot Mode selection
GPIO_AD_B0_05/BOOT_MODE1	Input	
GPIO_B0_04	Input	BOOT_CFG1[0]
GPIO_B0_05	Input	BOOT_CFG1[1]
GPIO_B0_06	Input	BOOT_CFG1[2]
GPIO_B0_07	Input	BOOT_CFG1[3]
GPIO_B0_08	Input	BOOT_CFG1[4]
GPIO_B0_09	Input	BOOT_CFG1[5]
GPIO_B0_10	Input	BOOT_CFG1[6]
GPIO_B0_11	Input	BOOT_CFG1[7]
GPIO_B0_12	Input	BOOT_CFG2[0]
GPIO_B0_13	Input	BOOT_CFG2[1]
GPIO_B0_14	Input	BOOT_CFG2[2]

Table continues on the next page...

Table 9-4. GPIO Boot Overrides (continued)

Package Pin	Direction on reset	eFuse
GPIO_B0_15	Input	BOOT_CFG2[3]

NOTE

Refer to the Fusemap chapter for more information on fuses mapped to the GPIO pins.

The input pins above are sampled at boot, and can be used to override the corresponding eFuse values, depending on the setting of the BT_FUSE_SEL fuse.

BT_FUSE_SEL = 1 is programmed on every Teensy board.
These pins do not override eFuse values during boot.

9.4.3 Device Configuration Data (DCD)

The DCD is the configuration information contained in the program image (external to the ROM) that the ROM interprets to configure various on-chip peripherals. See Device Configuration Data (DCD) for more details on DCD.

9.5 Device initialization

This section describes the details of the ROM and provides the initialization details.

This includes details on:

- The ROM memory map
- The RAM memory map
- On-chip blocks that the ROM must use or change the POR register default values
- Clock initialization
- Enabling the L1 I/D cache
- Exception handling and interrupt handling

9.5.1 Internal ROM/RAM memory map

These figures show the internal ROM and RAM memory map:

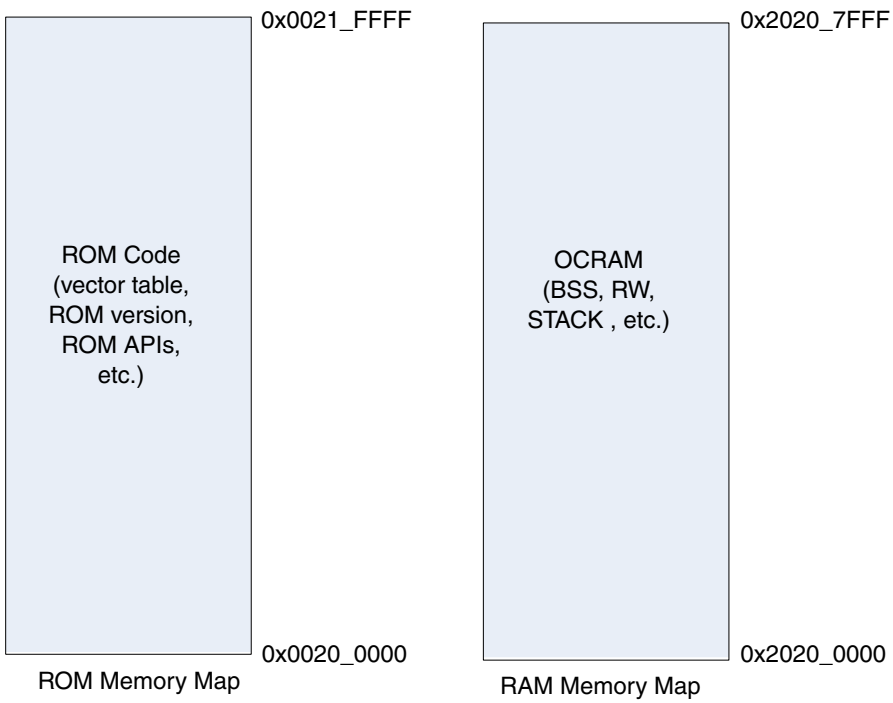


Figure 9-2. Internal ROM and RAM memory map

NOTE

The RAM space occupied by ROM cannot be used as part of the boot image. The entire OCRAM region can be used freely after the boot. The above OCRAM region must be reserved if the user application needs to call the HAB API for image authentication.

9.5.2 Boot block activation

The boot ROM affects a number of different hardware blocks which are activated and play a vital role in the boot flow.

The ROM configures and uses the following blocks (listed in an alphabetical order) during the boot process. Note that the blocks actually used depend on the boot mode and the boot device selection:

Block	Description
CCM	Clock Control Module
FlexSPI	Flexible SPI Interface which supports serial NOR, Serial NAND and serial RAM devices
OCOTP	On Chip One Time Programmable Controller containing the eFuses

Table continues on the next page...

Block	Description
IOMUXC	I/O Multiplexer Control which allows the GPIO use to override the eFuse boot settings
IOMUX GPR	I/O Multiplexer control General Purpose Registers
LPSPI	Low Power SPI interface which supports serial NOR/EEPROM devices
SEMC	Smart External Memory Controller which supports parallel NOR and RAWNAND (SLC) devices
SNVS	Secure Non-Volatile Storage
SRC	System Reset Controller
USB	Used for Serial download of a boot device provisioning program
uSDHC	Ultra-Secure Digital Host Controller
WDOG 1	WatchDog Timer
TRNG	True Random Number Generator
PIT	Periodic Interrupt Timer
Hash-AES	Crypto module

9.5.3 Clocks at boot time

The following table provides the clock frequency configurations for the boot core.

Table 9-5. Normal Frequency Clock Configurations

BOOT_FREQ(0x460[2])	LPB_BOOT(0x470[22:21])	Core Clock Frequency(MHz)
0	0	396
	1	198
	2	99
	3	49.5
1	0	528
	1	264
	2	132
	3	66

Following reset, the Arm core has access to all peripherals. The ROM code will disable the clocks associated with the following Clock Gate Enables. See System Clocks for information about the CCM output clocks' system-level connectivity.

Table 9-6. List of Disabled Clock Gate Enables

Clock Name
CCM_CCGR0_CG6
CCM_CCGR0_CG7
CCM_CCGR0_CG8

Table continues on the next page...

Table 9-6. List of Disabled Clock Gate Enables (continued)

Clock Name
CCM_CCGR0_CG9
CCM_CCGR0_CG10
CCM_CCGR0_CG12
CCM_CCGR0_CG13
CCM_CCGR0_CG14
CCM_CCGR1_CG0
CCM_CCGR1_CG1
CCM_CCGR1_CG2
CCM_CCGR1_CG3
CCM_CCGR1_CG4
CCM_CCGR1_CG5
CCM_CCGR1_CG6
CCM_CCGR1_CG8
CCM_CCGR1_CG12
CCM_CCGR2_CG1
CCM_CCGR2_CG3
CCM_CCGR2_CG4
CCM_CCGR2_CG5
CCM_CCGR2_CG14
CCM_CCGR2_CG15
CCM_CCGR3_CG0
CCM_CCGR3_CG1
CCM_CCGR3_CG3
CCM_CCGR3_CG5
CCM_CCGR3_CG10
CCM_CCGR3_CG11
CCM_CCGR3_CG12
CCM_CCGR3_CG13
CCM_CCGR4_CG8
CCM_CCGR4_CG9
CCM_CCGR4_CG10
CCM_CCGR4_CG11
CCM_CCGR4_CG12
CCM_CCGR4_CG13
CCM_CCGR4_CG14
CCM_CCGR4_CG15
CCM_CCGR5_CG1
CCM_CCGR5_CG3
CCM_CCGR5_CG4

Table continues on the next page...

Table 9-6. List of Disabled Clock Gate Enables (continued)

Clock Name
CCM_CCGR5_CG7
CCM_CCGR5_CG9
CCM_CCGR5_CG10
CCM_CCGR5_CG11
CCM_CCGR5_CG12
CCM_CCGR5_CG13
CCM_CCGR6_CG0
CCM_CCGR6_CG1
CCM_CCGR6_CG2
CCM_CCGR6_CG5
CCM_CCGR6_CG6
CCM_CCGR6_CG7
CCM_CCGR6_CG8
CCM_CCGR6_CG12
CCM_CCGR6_CG13
CCM_CCGR6_CG14
CCM_CCGR6_CG15

This device configures the clock to the following state during boot.

Table 9-7. ROM Clock Setting

Register	Setting
CCM_CACRR	0x00000001
CCM_CBCDR	0x000A8200 (core clock = 396MHz, default boot frequency)
CCM_CSCDR1	0x06490B03
CCM_CBCMR	0x75AE8104
CCM_CSCMR1	0x67930001
CCM_ANALOG_PLL_ARM	0x80002042
CCM_ANALOG_PLL_SYS	0x80002001
CCM_ANALOG_PLL_USB1	0x80003040

NOTE

1. The boot ROM enables MPU protection under HAB-closed mode, to prevent execution of any other supported memory regions than the ROM region, during boot code execution.

This feature puts restrictions on the SW debug on the RT106x chip. The debug tool needs to disable the MPU first, before executing the code on the device under HAB-closed mode.

2. The boot ROM enables LUART1 and USB1 interrupts in serial downloader mode, and it disables these interrupts before jumping to user applications. However, if the boot ROM execution is interrupted by the debug tool, it cannot disable these interrupts; the user application needs to disable these interrupts in the startup codes.
3. The boot ROM configures the PIT channel 0 and channel 1 registers but does not restore them before jumping to boot image. The user application needs to re-configure the PIT channels explicitly instead of relying on the default register values.
4. The PFD_480_PFD1 is selected as the FLEXSPI clock source. The user application needs to take special care on adjusting the PFD_480_PFD1 setting in the XiP boot case because the clock update impacts the FLEXSPI timing.

9.5.4 Enabling Caches

The boot ROM includes a feature that enables the caches to improve the boot speed.

L1 instruction cache is enabled at the start of image download. The L1 data cache is enabled at the start of image authentication. Both the caches are disabled by ROM before leaving the ROM execution.

By default, the L1-ICache and DCache are enabled by ROM. However, there are fuse bits that can be programmed for ROM not to enable the L1 I/DCache at boot.

The Cache features are controlled by the BT_ICACHE_DISABLE fuse and BT_DCACHE_DISABLE fuses. By default, both fuses are not blown meaning the ROM uses the L1-ICache and L1-DCache of the Arm core. This improves the performance of the HAB signature verification software.

9.5.5 Exception handling

The exception vectors (interrupt vectors/vector table) are located at the start of ROM. During the boot phase, CPU loads the SP and PC from exception vectors and then boot to ROM.

After booting, the program image can relocate the vectors as required.

9.5.6 Interrupt handling during boot

No special interrupt-handling routines are required during the boot process. The interrupts are disabled during the boot ROM execution and may be enabled in a later boot stage.

9.5.7 Persistent bits

Some modes of the boot ROM require the registers that keep their values after a warm reset. The SRC General-Purpose registers are used for this purpose.

See this table for persistent bits list and description:

Table 9-8. Persistent bits

Bit name	Bit location	Description
PERSIST_SECONDARY_BOOT	SRC_GPR10[30]	This bit identifies which image must be used—primary and secondary. Used only for eMMC/SD/FlexSPI NOR boot.
PERSISTENT_ENTRY0[31:0]	SRC_GPR1[31:0]	Holds the entry function for the CPU0 to wake up from the low-power mode.
PERSISTENT_ARG0[31:0]	SRC_GPR2[31:0]	Holds the argument of entry function for the CPU0 to wake up from the low-power mode.
PERSIST_REDUNDANT_BOOT	SRC_GPR10[27:26]	This field identifies which image must be used - 0/1/2/3. Used for both SPI NAND and SLC raw NAND devices.

9.6 Boot devices

The chip supports the following boot flash devices:

- Serial NOR flash via FlexSPI Interface
- Serial NAND Flash via FlexSPI Interface

- Parallel NOR flash with the Smart External Memory Controller (SEMC), located on CS0, 16-bit bus width.
- NAND Flash with SEMC interface, located on CS0, 8-bit/16-bit bus width.
- SD/MMC/eSD/SDXC/eMMC4.4 via uSDHC interface, supporting high capacity cards
- Serial NOR/EEPROM boot via LPSPI

The selection of the external boot device type is controlled by the BOOT_CFG1[7:4] eFUSEs. See this table for more details:

Table 9-9. Boot device selection

BOOT_CFG1[7:4]	Boot device
0000b	Serial NOR boot via FlexSPI
01xxb	SD Boot via uSDHC
10xxb	eMMC/MMC boot via uSDHC
001xb	SLC NAND boot via SEMC
0001b	Parallel NOR boot via SEMC
11xxb	Serial NAND boot via FlexSPI

NOTE

A user can use the LPSPI to boot from Serial NOR/EEPROM. Booting from Serial NOR/EEPROM via an LPSPI port is not intended as the primary device, but as a recovery boot device if the primary boot device fails.

In order to enable a recovery boot device, EEPROM_RECOVERY_EN fuse must be set to 1, and other fuses listed in Serial NOR/EEPROM eFUSE configuration must be properly set.

9.6.1 Serial NOR Flash Boot via FlexSPI

9.6.1.1 Serial NOR eFUSE Configuration

Table 9-10. Fuse definition for Serial NOR over FlexSPI

Fuse	Config	Definitions	GPIO	Shipped Value	Settings
BOOT_CFG1[0]	OEM	xSPI FLASH Auto Probe	Yes	0	0 – Disabled 1 – Enabled

Table continues on the next page...

Table 9-10. Fuse definition for Serial NOR over FlexSPI (continued)

Fuse	Config	Definitions	GPIO	Shipped Value	Settings
BOOT_CFG1[1]	OEM	Encrypted XIP	Yes	0	0 – Disabled 1 – Enabled
BOOT_CFG1[3:2]	OEM	xSPI FLASH Auto Probe Type	Yes	0	0 – QuadSPI NOR 1 – MXIC Octal 2 – Micron Octal 3 – Adesto Octal
BOOT_CFG1[7:4]	OEM	Boot device selection	Yes	0	0 – Serial NOR device is selected as boot device
BOOT_CFG2[2:0]	OEM	Flash Type	Yes	0	000b–Device supports 3B (0x03 + 24-bit Address) read by default (command: 0x03, SPI mode) 001b–Device supports 4B (0x13 + 24-bit Address) read by default (command: 0x13, SPI mode) 010b–HyperFlash 1V8 (HyperBus Protocol) 011b–HyperFlash 3V3 (HyperBus protocol) 100b–MXIC Octal DDR (command: 0xEE,0x11, OPI mode) 101b–Micron Octal DDR (command: 0xFD, OPI mode) 110b–Reserved 111b–QSPI device supports 3B read by default (on secondary pinmux option) (command: 0x03, SPI mode)
0x6E0[3:1] (BOOT_CONFIG_ MISC)	OEM	xSPI FLASH Frequency	No	0	0 – 100MHz 1 – 120MHz 2 – 133MHz 3 – 166MHz 5 – 80MHz 6 – 60MHz Others – Reserved
0x6E0[5:4] (BOOT_CONFIG_ MISC)	OEM	Hold time before access the Flash device	No	0	0 – 500us 1 – 1ms 2 – 3ms 3 – 10ms
0x6E0[11:8] (BOOT_CONFIG_ MISC)	OEM	xSPI FLASH Dummy Cycle	No	0	0 – Dummy cycles is auto-probed Others – Actual dummy cycles for Read command

Table continues on the next page...

Table 9-10. Fuse definition for Serial NOR over FlexSPI (continued)

Fuse	Config	Definitions	GPIO	Shipped Value	Settings
0x6E0[15:12] (BOOT_CONFIG_ MISC)	OEM	xSPI FLASH image size	No	0	0 – Image size equals to Secondary Image offset 1 – 1MB 2 – 2MB ... – ... 12 – 12MB 13 – 256KB 14 – 512KB 15 – 768KB
0x6E0[23:16] (BOOT_CONFIG_ MISC)	OEM	xSPI FLASH_SEC_IMAGE_OFFSET	No	0	Offset = 256KB × fuse value

NOTE

If the xSPI FLASH Auto Probe feature is enabled, the following is the logic how this feature works with other fuse combinations:

- Flash Type - If Flash type is 0, the "xSPI FLASH Auto Probe Type" takes effect for the Flash type selection. If Flash Type is greater than 1, the "Flash Type" fuse is used for Flash type selection, ROM will issue specific command to probe the presence of Serial NOR FLASH.
- xSPI FLASH Frequency - This field is used for specifying the Flash working frequency.

9.6.1.2 FlexSPI Serial NOR Flash Boot Operation

The Boot ROM attempts to boot from Serial NOR flash if the BOOT_CFG1 [7:4] fuses are programmed to 0b’0000 as shown in the Serial NOR eFUSE Configuration table, then the ROM will initialize FlexSPI interface. FlexSPI interface initialization is a two-step process.

1. The ROM expects the 512-byte FlexSPI NOR configuration parameters (as explained in the next section) to be present at offset 0 in Serial NOR flash attached to FLEXSPI_A_SS0_B. The ROM can probe the presence of Serial NOR Flash and generate these configuration parameters accordingly via the combination of Fuses in

the table above, or reads these configuration parameters using the read command specified by BOOT_CFG 2[2:0] with serial clock operating at 30 MHz.

2. ROM configures FlexSPI interface with the parameters provided in configuration block read from Serial NOR flash and starts the boot procedure. Refer to Table 9-18 for details regarding FlexSPI configuration parameters and to the FlexSPI NOR boot flow chart for detailed boot flow chart of FlexSPI NOR.

Both booting an XIP and non XIP image are supported from Serial NOR Flash. For XIP boot, the image has to be built for FlexSPI address space; and for non XIP, the image can be built to execute from internal RAM.

NOTE

A non-XIP image's execute address space should NOT be overlapped with the reserved OCRAM region described in the section "Internal ROM/RAM memory map". And it is also highly recommended to avoid using address space near 0x00000000, typically up to 0x00000003.

9.6.1.3 FlexSPI NOR boot flow chart

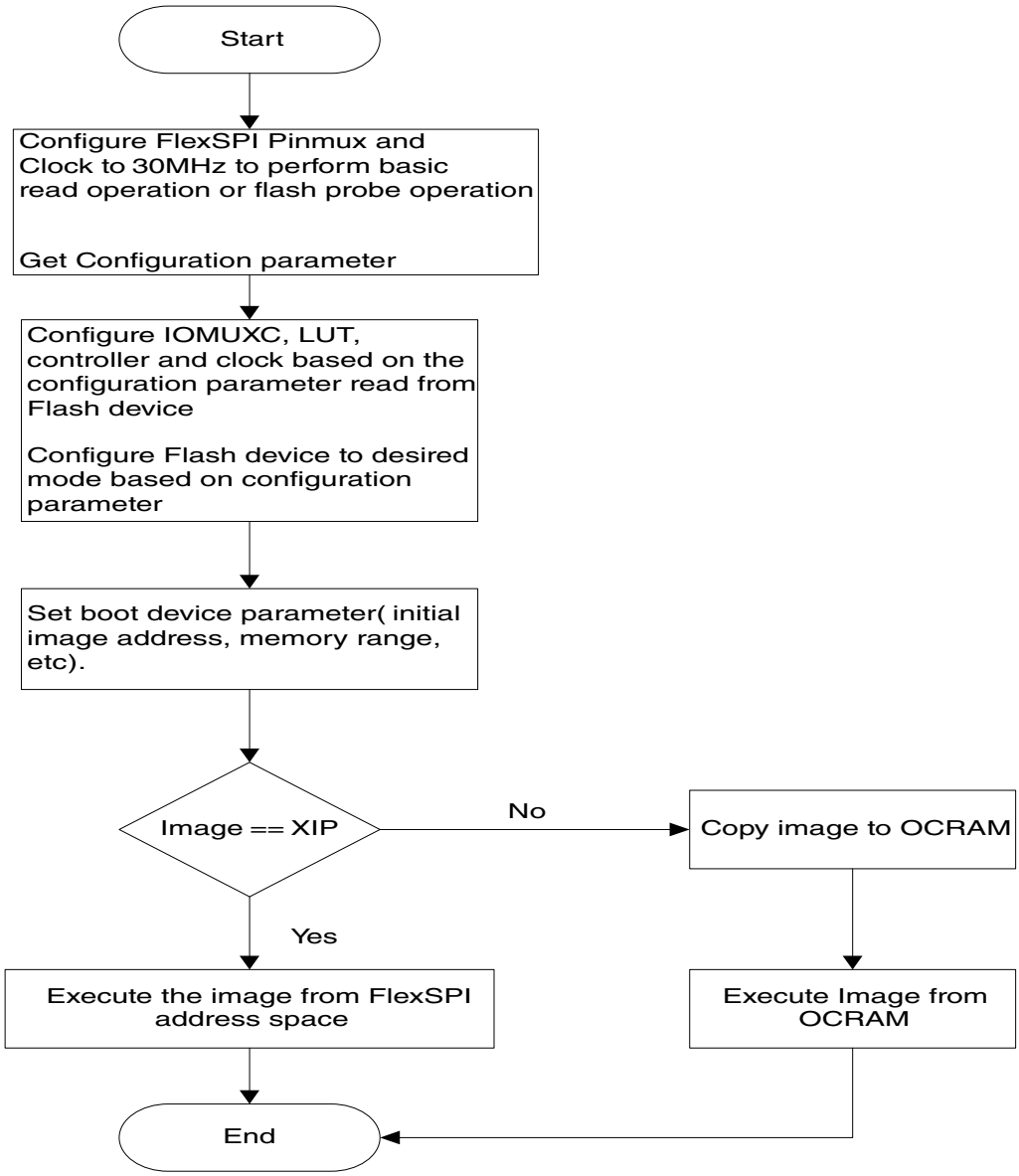


Figure 9-3. FlexSPI NOR boot flow

9.6.2 Serial NAND Flash Boot over FlexSPI

The boot ROM supports a number of Serial NAND Flash devices from different vendors. The Embedded Error Correction and Control (ECC) module in SPI NAND devices are used to detect and correct the errors.

9.6.2.1 Serial NAND eFUSE Configuration

The boot ROM determines the configuration of external Serial NAND flash by parameters, either provided by eFUSE, or sampled on GPIO pins during boot. See below table for parameters details.

Table 9-11. Fuse definition for Serial NAND over FlexSPI

Fuse	Config	Definitions	GPIO	Shipped Value	Settings
BOOT_CFG1[1:0]	OEM	Search Stride for FCB and DBBT Search strides in terms of page	Yes	0	0 - 64 1 - 128 2 - 256 3 - 32
BOOT_CFG1[3:2]	OEM	Hold Time before access to Serial NAND	Yes	0	0 - Hold time determined by Read Status command 1 - 500us 2 - 1ms 3 - 3ms
BOOT_CFG1[4]	OEM	Column address width	Yes	0	0 - 12 bits 1 - 13 bits
BOOT_CFG1[5]	OEM	Default safe communication frequency	Yes	0	0 – High Speed (50MHz) 1 – Low Speed (30MHz)
BOOT_CFG1[7:6]	OEM	Primary boot device selection	Yes	0	00 – Serial NOR 11 – Serial NAND
BOOT_CFG2[0]	OEM	Boot Search Count of FCB and DBBT	Yes	0	0 - 1 1 - 2
BOOT_CFG2[2:1]	OEM	CS de-asserted interval between two commands	Yes	0	0 – 100ns 1 – 200ns 2 – 400ns 3 – 50ns
0x6E0[3:0]	OEM	SPI NAND BOOT - Busy Bit offset	No	0	
0x6E0[7]	OEM	SPI NAND BOOT - Override Busy Offset	No	0	0 – Use default busy bit offset 0 1 – Override default busy bit offset using Busy bit offset
0x6E0[13:8]	OEM	SPI NAND Boot - Page Read Time (in terms of micro seconds)	No	0	

Table continues on the next page...

Table 9-11. Fuse definition for Serial NAND over FlexSPI (continued)

Fuse	Config	Definitions	GPIO	Shipped Value	Settings
0x6E0[23:16]	OEM	Page Read Command	No	0	Available only if it is not 0
0x6F0[31:24]	OEM	Cache Read command	No	0	Available only if it is not 0

NOTE

BOOT_CFGx sampled on GPIO pins depends on BT_FUSE_SEL setting.

BT_FUSE_SEL = 1 is programmed on every Teensy board.
BOOT_CFGx pins do not override eFuse values during boot.

9.6.2.2 FlexSPI NAND Flash Boot Flow and Boot Control Blocks (BCB)

There are two BCB data structures:

- FCB
- DBBT

As part of the NAND media initialization, the ROM driver uses safe NAND timings to search for a Firmware Configuration Block (FCB) that contains the optimum NAND timings, page address of Discovered Bad Block Table (DBBT) Search Area and start page address of primary and secondary firmware.

The built-in HW ECC in Serial NAND device is used during data read, the FCB data structure is protected using CRC checksum. Driver reads raw 2048 bytes of first sector and runs through CRC check that determines whether FCB data is valid or not.

If the FCB is found, the optimum NAND timings are loaded for further reads. If the ECC fails, or the fingerprints do not match, the Block Search state machine increments page number to Search Stride number of pages to read for the next BCB until SearchCount pages have been read.

If search fails to find a valid FCB, the NAND driver responds with an error and the boot ROM enters into serial download mode.

The FCB contains the page address of DBBT Search Area, and the page address for primary and secondary boot images. DBBT is searched in DBBT Search Area just like how FCB is searched. After the FCB is read, the DBBT is loaded, and the primary or secondary boot image is loaded using starting page address from FCB.

Figure 9-4 shows the state diagram of FCB search.

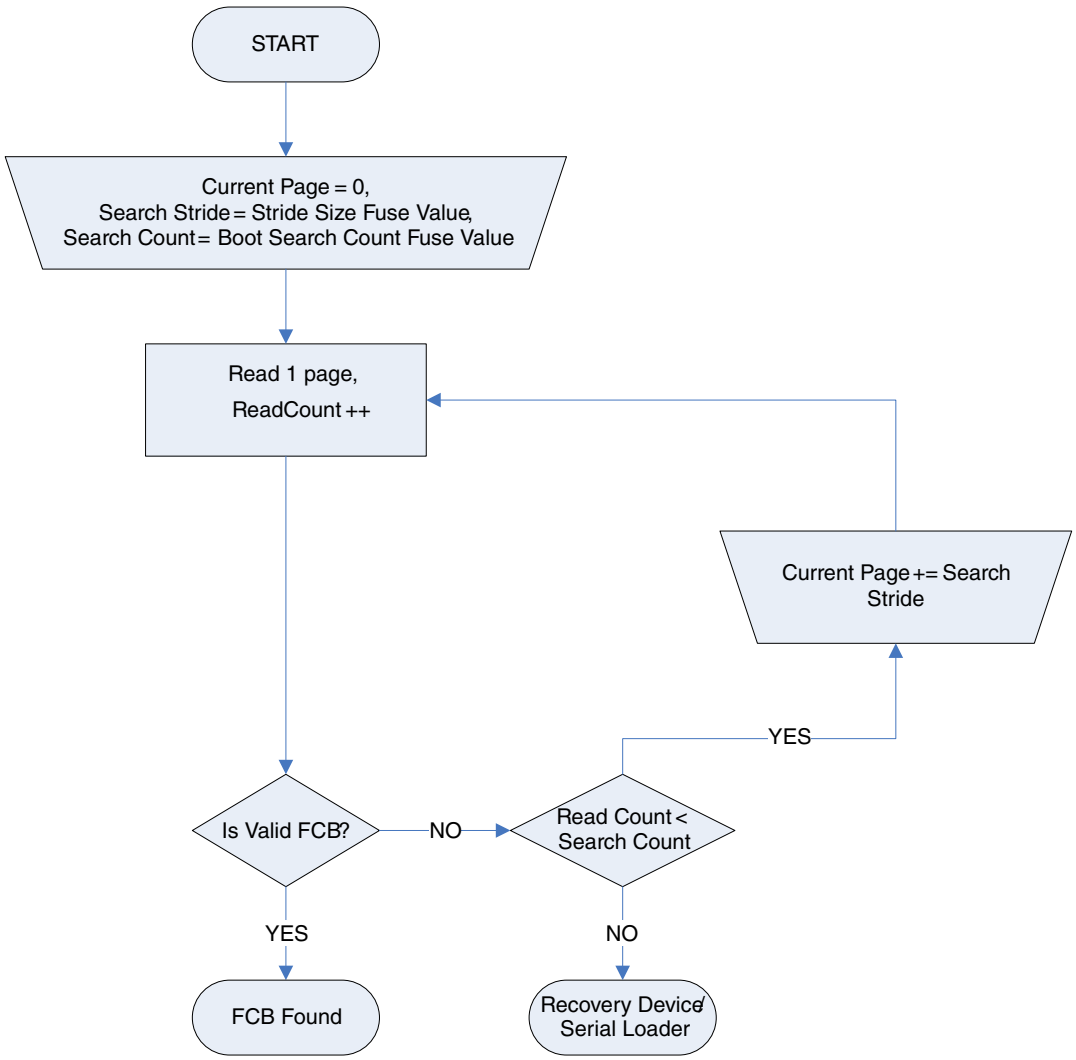


Figure 9-4. FCB Search Flow

After FCB is found, the boot ROM searches for the Discovered Bad Blocks Table (DBBT). If DBBTSearchStartPage is 0 in the FCB, then ROM assumes that there are no bad blocks in the NAND device boot area. See Figure 9-5 for the DBBT search flow.

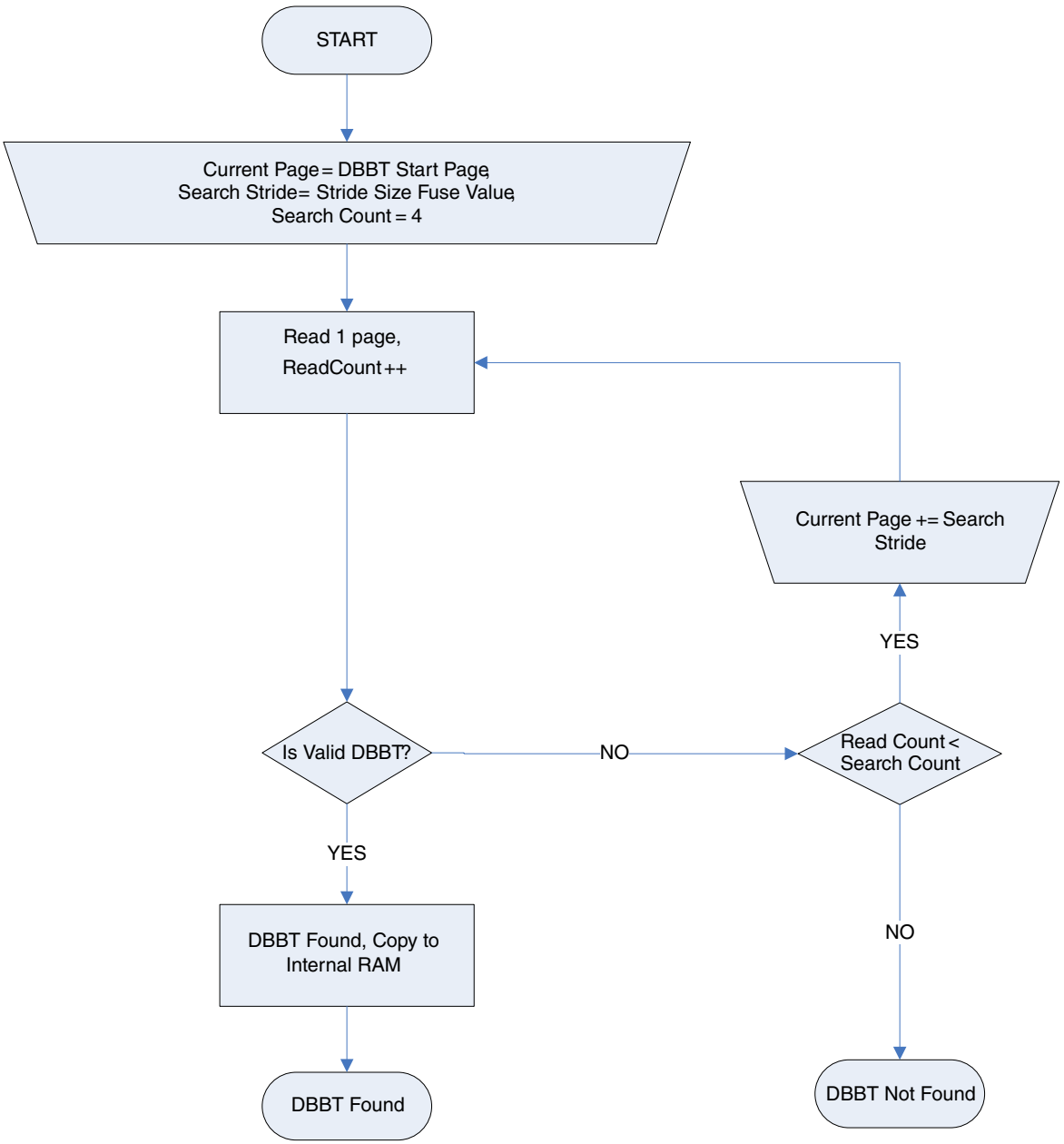


Figure 9-5. DBBT Search Flow

The BCB search and load function also monitors the ECC correction threshold and sets the PERSIST_BLOCK_REWRITE persistent bit if the threshold exceeds the maximum ECC correction ability.

If during primary image read there is a page with a number of errors higher than ECC can correct, the boot ROM will turn on PERSIST_SECONDARY_BOOT bit and perform SW reset (After SW reset, secondary image is used).

If during secondary image read there is a page with number of errors higher than ECC can correct, the boot ROM goes to serial downloader.

9.6.2.3 Firmware Configuration Block

The FCB is at the first page in the first good block. The FCB should be present at each search stride of the search area.

The search area contains copies of the FCB at each stride distance, in case the first Serial NAND block becomes corrupted, the ROM will find its copy in the next Serial NAND block. The search area should span over at least two Serial NAND blocks. The location information for DBBT search area and images are all specified in the FCB. The following table shows the Flash Control Block Structure.

Table 9-12. Flash Control Block Structure

Name	Offset	Size Bytes	Description
crcChecksum	0x000	4	Checksum
fingerprint	0x004	4	0x4E46_4342 ASCII: “NFCB”
version	0x008	4	0x0000_0001
DBBTSearchStartPage	0x00C	4	Start Page address for bad block table search area
searchStride	0x010	2	Search stride for DBBT and FCB search. Not used by ROM Max value is 8.
searchCount	0x012	2	Copies of DBBT and FCB. Not used by ROM, max value is 8.
firmwareCopies	0x014	4	Firmware copies Valid range 1-4.
Reserved	0x018	40	Reserved for future use Must be set to 0.
firmwareInfoTable	0x40	64	This table consists of (up to 8 entries): **Field** / **Size** / **Description** StartPage / 4 / Start page of this firmware pageCount / 4 / Pages in this firmware

Table continues on the next page...

Table 9-12. Flash Control Block Structure (continued)

Name	Offset	Size Bytes	Description
			NOTE: The StartPage must be the first page of a NAND block.
Reserved	0x080	128	Reserved Must be set to 0
spiNandConfigBlock	0x100	512	Serial NAND configuration block over FlexSPI
Reserved	0x300	256	Must be set to 0

NOTE

1. The “crcChecksum” is calculated with an MPEG2 variant of CRC-32. See Table 9-13 for more details.
2. The “crcChecksum” calculation starts from fingerprint to the end of FCB, 1020 bytes in total.
3. The “spiNandConfigBlock” is FlexSPI NAND configuration block which consists of common FlexSPI memory configuration block and Serial NAND specified configuration parameters.

Table 9-13. CRC-32 variant algorithm

Property	Description
Width	32 bits
Polynomial	0x04C11DB7
Init Value	0xFFFFFFFF
Reflect in	False
Reflect Out	False
XOR Out	0x00000000

9.6.2.4 Discovered Bad Blocks Table (DBBT)

Table 9-14. DBBT Structure

Name	Offset	Size in Bytes	Decription
crcChecksum	0x000	4	Checksum
Fingerprint	0x004	4	32-bit word with a value of 0x4442_4254, in ASCII “DBBT”

Table continues on the next page...

Table 9-14. DBBT Structure (continued)

Name	Offset	Size in Bytes	Decription
Version	0x008	4	32-bit version number, this version of DBBT is 0x00000001
-	0x00C	4	Reserved
badBlockNumber	0x010	4	Number of bad blocks
Reserved	0x014	12	Must be filled with 0x00s
Bad Block entries	0x020	1024 (256*4)	Each bad block entry is a 32-bit value specifying the number of a found bad block. The number of valid bad block entries is specified by the badBlockNumber field, where valid bad block entries are stored sequentially starting at the beginning of the bad block entries segment. Unused bad block entries (those beyond the badBlockNumber) should be filled with 0xFs.

NOTE

1. Maximum badBlockNumber is 256.
2. The "crcChecksum" is calculated with the same algorithm as the one in FCB, from Fingerprint to the end of DBBT, 1052 bytes in total.

9.6.2.5 Bad block handling in ROM

During the firmware boot, at the block boundary, the Bad Block table is searched for a match to the next block.

If no match is found, the next block can be loaded. If a match is found, the block must be skipped and the next block checked.

9.6.3 Serial NOR and NAND Configuration based on FlexSPI Interface

The ROM SW supports Serial NOR and Serial NAND based on FlexSPI module, using a 448-bytes common FlexSPI configuration block and several specified parameters for Serial NOR and Serial NAND respectively. See the following sections for more details.

9.6.3.1 FlexSPI Configuration Block

FlexSPI Configuration block consists of parameters regarding specific Flash devices including read command sequence, quad mode enablement sequence (optional), etc.

Table 9-15. FlexSPI Configuration block

Name	Offset	Size(bytes)	Description
Tag	0x000	4	0x42464346, ascii: 'FCFB'
Version	0x004	4	0x56010000 / 0x56010100 [07:00] bugfix = 0 [15:08] minor [23:16] major = 1 [31:24] ascii 'V'
-	0x008	4	Reserved
readSampleClkSrc	0x00C	1	0 – internal loopback 1 – loopback from DQS pad 3 – Flash provided DQS
csHoldTime	0x00D	1	Serial Flash CS Hold Time Recommend default value is 0x03
csSetupTime	0x00E	1	Serial Flash CS setup time Recommended default value is 0x03
columnAdressWidth	0x00F	1	3 – For HyperFlash 12/13 – For Serial NAND, see the NAND FLASH datasheet to find the correct value 0 – Other devices
deviceModeCfgEnable	0x010	1	Device Mode Configuration Enable feature 0 – Disabled 1 – Enabled
-	0x011	1	Reserved
waitTimeCfgCommands	0x012	2	Wait time for all configuration commands, unit 100us. Available for device that support v1.1.0 FlexSPI configuration block. If it is greater than 0, ROM will wait waitTimeCfgCommands * 100us for all device memory configuration commands instead of using read status to wait until these commands complete.
deviceModeSeq	0x014	4	Sequence parameter for device mode configuration Bit[7:0] - number of LUT sequences for Device mode configuration command Bit[15:8] - starting LUT index of Device mode configuration command Bit[31:16] - must be 0
deviceModeArg	0x018	4	Device Mode argument, effective only when deviceModeCfgEnable = 1

Table continues on the next page...

Table 9-15. FlexSPI Configuration block (continued)

Name	Offset	Size(bytes)	Description
configCmdEnable	0x01C	1	Config Command Enable feature 0 – Disabled 1 – Enabled
-	0x01D	3	Reserved
configCmdSeqs	0x020	12	Sequences for Config Command, allow 3 separate configuration command sequences.
-	0x02C	4	Reserved
cfgCmdArgs	0x030	12	Arguments for each separate configuration command sequence.
-	0x03C	4	Reserved
controllerMiscOption	0x040	4	Bit0 – differential clock enable Bit1 – CK2 enable, must set to 0 in this silicon Bit2 – ParallelModeEnable, must set to 0 for this silicon Bit3 – wordAddressableEnable Bit4 – Safe Configuration Frequency enable set to 1 for the devices that support DDR Read instructions Bit5 – Pad Setting Override Enable Bit6 – DDR Mode Enable, set to 1 for device supports DDR read command
deviceType	0x044	1	1 – Serial NOR 2 – Serial NAND
sflashPadType	0x045	1	1 – Single pad 2 – Dual pads 4 – Quad pads 8 – Octal pads
serialClkFreq	0x046	1	Chip specific value, for this silicon 1 – 30 MHz 2 – 50 MHz 3 – 60 MHz 4 – 75 MHz 5 – 80 MHz 6 – 100 MHz 7 – 120 MHz 8 – 133 MHz 9 - 166 MHz Other value: 30 MHz
lutCustomSeqEnable	0x047	1	0 – Use pre-defined LUT sequence index and number 1 - Use LUT sequence parameters provided in this block
-	0x048	8	Reserved

Table continues on the next page...

Table 9-15. FlexSPI Configuration block (continued)

Name	Offset	Size(bytes)	Description
sflashA1Size	0x050	4	For SPI NOR, need to fill with actual size For SPI NAND, need to fill with actual size * 2
sflashA2Size	0x054	4	The same as above
sflashB1Size	0x058	4	The same as above
sflashB2Size	0x05C	4	The same as above
csPadSettingOverride	0x060	4	Set to 0 if it is not supported
sclkPadSettingOverride	0x064	4	Set to 0 if it is not supported
dataPadSettingOverride	0x068	4	Set to 0 if it is not supported
dqsPadSettingOverride	0x06C	4	Set to 0 if it is not supported
timeoutInMs	0x070	4	Maximum wait time during read busy status 0 – Disabled timeout checking feature Other value – Timeout if the wait time exceeds this value
commandInterval	0x074	4	Unit: ns Currently, it is used for SPI NAND only at high frequency
dataValidTime	0x078	4	Time from clock edge to data valid edge, unit ns. This field is used when the FlexSPI Root clock is less than 100 MHz and the read sample clock source is device provided DQS signal without CK2 support. [31:16] data valid time for DLLB in terms of 0.1 ns [15:0] data valid time for DLLA in terms of 0.1 ns
busyOffset	0x07C	2	busy bit offset, valid range :0-31
busyBitPolarity	0x07E	2	0 – busy bit is 1 if device is busy 1 – busy bit is 0 if device is busy
lookupTable	0x080	256	Lookup table
lutCustomSeq	0x180	48	Customized LUT sequence, see below table for details.
	0x1B0	16	Reserved for future use

Note:

1. To customize the LUT sequence for some specific device, users need to enable “lutCustomSeqEnable” and fill in corresponding “lutCustomSeq” field specified by command index below.
2. For Serial (SPI) NOR, the pre-defined LUT index is as follows:

Table 9-16. LUT sequence definition for Serial NOR

Command Index	Name	Index in lookup table	Description
0	Read	0	Read command Sequence
1	ReadStatus	1	Read Status command

Table continues on the next page...

Table 9-16. LUT sequence definition for Serial NOR (continued)

Command Index	Name	Index in lookup table	Description
2	WriteEnable	3	Write Enable command sequence
3	EraseSector	5	Erase Sector Command
4	PageProgram	9	Page Program Command
5	ChipErase	11	Full Chip Erase
6	Dummy	15	Dummy Command as needed
7-12	Reserved	2,4,6,7,8,10,12,13,14	All reserved indexes can be freely used for other purpose
13	NOR_CMD_LUT_SEQ_IDX_READ_SFDP	13	Read SFDP sequence in lookupTable id stored in config block
14	NOR_CMD_LUT_SEQ_IDX_RESTORE_NOCMD	14	Restore 0-4-4/0-8-8 mode sequence id in lookupTable stored in config block

3. For Serial (SPI) NAND, the pre-defined LUT index is as follows:

Table 9-17. LUT sequence definition for Serial NAND

Command Index	Name	Index in lookup table	Description
0	ReadFromCache	0	Read from cache
1	ReadStatus	1	Read Status
2	WriteEnable	3	Write Enable
3	BlockErase	5	Erase block
4	ProgramLoad	9	Program Load
5	ReadPage	11	Read page to cache
6	ReadEccStatus	13	Read ECC Status
7	ProgramExecute	14	Program Execute
8	ReadFromCacheOdd	4	Read from Cache while page in odd plane
9	ProgramLoadOdd	10	Program Load for pages within odd blocks
	Reserved	2,6,7,8,12,15	All reserved indexes can be freely used for other purposes

NOTE

1. All the pre-defined LUT indexes are only applicable to boot stage. User application can use the whole 16 LUT entries freely based on their requirement.
2. The FlexSPI NOR ROM APIs occupy LUT index 0-5. User application should **NOT** use these LUT indexes if the ROM API is called in their application frequently.

9.6.3.2 Serial NOR configuration block (512 bytes)

Table 9-18. Serial NOR configuration block

Name	Offset	Size (Bytes)	Description
memCfg	0	448	The common memory configuration block, see FlexSPI configuration block for more details
pageSize	0x1C0	4	Page size in terms of bytes, not used by ROM
sectorSize	0x1C4	4	Sector size in terms of bytes, not used by ROM
ipCmdSerialClkFreq	0x1C8	1	Chip specific value, not used by ROM 0 – No change, keep current serial clock unchanged 1 – 30 MHz 2 – 50 MHz 3 – 60 MHz 4 – 75 MHz 5 – 80 MHz 6 – 100 MHz 7 – 120 MHz 8 – 133 MHz 9 – 166 MHz
Reserved	0x1CC	55	Reserved for future use

9.6.3.3 Serial NAND configuration block (512 bytes)

Table 9-19. Serial NAND configuration block

Name	Offset	Size (Bytes)	Description
memCfg	0	448	The common memory configuration block, see FlexSPI configuration block for more details
pageDataSize	0x1C0	4	Page size in terms of bytes, usually, it is 2048 or 4096
pageTotalSize	0x1C4	4	It equals to 2 ^ width of column adddress
pagesPerBlock	0x1C8	4	Pages in one block
bypassReadStatus	0x1CC	1	0 – Read Status Register 1 – Bypass Read status register
bypassEccRead	0x1CD	1	0 – Perform ECC read 1 – Bypass ECC read
hasMultiPlanes	0x1CE	1	0 – Only 1 plane 1 – Has two planes
skippOddBlocks	0x1CF	1	0 – Read Odd blocks 1 – Skip Odd blocks
eccCheckCustomEnable	0x1D0	1	0 – Use the common ECC check command and ECC related masks 1 - Use ECC check related masks provided in this configuration block
ipCmdSerialClkFreq	0x1D1	1	Chip specific value, not used by ROM 0 – No change, keep current serial clock unchanged 1 – 30 MHz 2 – 50 MHz 3 – 60 MHz 4 – 75 MHz 5 – 80 MHz 6 – 100 MHz 7 – 120 MHz 8 – 133 MHz 9 – 166 MHz
readPageTimeUs	0x1D2	2	Wait time during page read, this field will take effect on if the bypassReadStatus is set to 1.

Table continues on the next page...

Table 9-19. Serial NAND configuration block (continued)

Name	Offset	Size (Bytes)	Description
			NOTE: Only applicable to ROM.
eccStatusMask	0x1D4	4	ECC Status Mask
eccFailureMask	0x1D8	4	ECC Check Failure mask
blocksPerDevice	0x1DC	4	Blocks in a Serial NAND
Reserved	0x1ED	32	Reserved for future use

Below is an example of Serial NAND configuration block for Winbond W25N01GVZEIG:

```
const flexspi_nand_config_t kSerialNandCfgBlk =
{
    .memConfig =
    {
        .tag = FLEXSPI_CFG_BLK_TAG,
        .version = FLEXSPI_CFG_BLK_VERSION,
        .readSampleClkSrc = kFlexSPIReadSampleClk_LoopbackInternally,
        .dataHoldTime = 3,
        .dataSetupTime = 3,
        .columnAddressWidth = 12,
        .deviceModeCfgEnable = 1,
        .deviceModeSeq = { 1, 2 },
        .deviceType = kFlexSpiDeviceType_SerialNAND,
        .sflashPadType = kSerialFlash_4Pads,
        .serialClkFreq = kFlexSpiSerialClk_50MHz,
        .lutCustomSeqEnable = 0,
        .sflashA1Size = 128 * 1024 * 1024U * 2, // Flash size = 2 * actual data size
(exclude spare space)
        .lookupTable =
        {
            // Read cache 4 I/0
            [4 * NOR_CMD_LUT_SEQ_IDX_READ] = FLEXSPI_LUT_SEQ(CMD_SDR, FLEXSPI_1PAD, 0xEB,
CADDR_SDR, FLEXSPI_4PAD, 0x10),
            [4 * NOR_CMD_LUT_SEQ_IDX_READ + 1] = FLEXSPI_LUT_SEQ(DUMMY_SDR, FLEXSPI_4PAD,
0x04, READ_SDR, FLEXSPI_4PAD, 0x80),
            // Clear Status1 flag
            [4 * 2] = FLEXSPI_LUT_SEQ(CMD_SDR, FLEXSPI_1PAD, 0x1F, CMD_SDR, FLEXSPI_1PAD,
0xA0),
            [4 * 2 + 1] = FLEXSPI_LUT_SEQ(CMD_SDR, FLEXSPI_1PAD, 0x00, STOP, FLEXSPI_1PAD,
0x00),
            // Read Page
            [4 * NAND_CMD_LUT_SEQ_IDX_READPAGE] = FLEXSPI_LUT_SEQ(CMD_SDR, FLEXSPI_1PAD,
0x13, RADDR_SDR, FLEXSPI_1PAD, 0x18),
            // Read Status
            [4 * NAND_CMD_LUT_SEQ_IDX_READSTATUS] = FLEXSPI_LUT_SEQ(CMD_SDR, FLEXSPI_1PAD,
0x0F, CMD_SDR, FLEXSPI_1PAD, 0xC0),
            [4 * NAND_CMD_LUT_SEQ_IDX_READSTATUS + 1] = FLEXSPI_LUT_SEQ(READ_SDR,
FLEXSPI_1PAD, 0x01, STOP, FLEXSPI_1PAD, 0),

            // Write Enable
            [4 * NAND_CMD_LUT_SEQ_IDX_WRITEENABLE] = FLEXSPI_LUT_SEQ(CMD_SDR, FLEXSPI_1PAD,
0x06, STOP, FLEXSPI_1PAD, 0),

            // Page Program Load 4x
            [4 * NAND_CMD_LUT_SEQ_IDX_PROGRAMLOAD] = FLEXSPI_LUT_SEQ(CMD_SDR, FLEXSPI_1PAD,
0x32, CADDR_SDR, FLEXSPI_1PAD, 0x10),
            [4 * NAND_CMD_LUT_SEQ_IDX_PROGRAMLOAD + 1] = FLEXSPI_LUT_SEQ(WRITE_SDR,
FLEXSPI_4PAD, 0x40, STOP, FLEXSPI_1PAD, 0),
            // Page Program Execute
            [4 * NAND_CMD_LUT_SEQ_IDX_PROGRAMEXECUTE] = FLEXSPI_LUT_SEQ(CMD_SDR,
```

```
FLEXSPI_1PAD, 0x10, RADDR_SDR, FLEXSPI_1PAD, 0x18),
            // Erase Sector
            [4 * NAND_CMD_LUT_SEQ_IDX_ERASEBLOCK] = FLEXSPI_LUT_SEQ(CMD_SDR, FLEXSPI_1PAD,
0xD8, RADDR_SDR, FLEXSPI_1PAD, 0x18),
            // Read ECC status
            [4 * NAND_CMD_LUT_SEQ_IDX_READECCSTAT] = FLEXSPI_LUT_SEQ(CMD_SDR, FLEXSPI_1PAD,
0x0F, CMD_SDR, FLEXSPI_1PAD, 0xC0),
            [4 * NAND_CMD_LUT_SEQ_IDX_READECCSTAT + 1] = FLEXSPI_LUT_SEQ(READ_SDR,
FLEXSPI_1PAD, 0x01, STOP, FLEXSPI_1PAD, 0),
        },
    },
    .pageDataSize = 2048,
    .pageTotalSize = 4096,
    .pagesPerBlock = 64,
};
```

9.6.4 Parallel NOR flash Boot over SEMC

The Smart External Memory Controller (SEMC) works in an asynchronous mode, and supports Muxed Address/Data scheme.

9.6.4.1 Parallel NOR eFuse Configuration

The boot ROM determines the configuration of external Parallel NOR flash by parameters, either provided by eFuse, or sampled on GPIO pins during boot. See below table for parameters details.

Table 9-20. Fuse definition for Parallel NOR over SEMC

Fuse	Config	Definitions	GPIO	Shipped Value	Settings
BOOT_CFG [7:4]	OEM	Primary boot device selection	Yes	0000	0001 – Parallel NOR
BOOT_CFG [2:0]	OEM	SEMC Clock frequency	Yes	000	000 – 33MHz 001 – 66MHz 010 – 108MHz 011 – 133MHz 1xx – 166MHz
BOOT_CFG [8]	OEM	AC timing parameter configuration	Yes	0	0 – Default configuration 1 – Configured by Fuse
BOOT_CFG [9]	OEM	Data port size	Yes	0	0 – 16 bits 1 – 8 bits
BOOT_CFG [10]	OEM	DQS pin pad enablement	Yes	0	0 – Disabled 1 – Enabled
0x6E0[2:0]	OEM	PCS pin selection	No	000	000 – CSX0 001 – CSX1 010 – CSX2

Table continues on the next page...

Table 9-20. Fuse definition for Parallel NOR over SEMC (continued)

Fuse	Config	Definitions	GPIO	Shipped Value	Settings
					011 – CSX3 1xx – A8
0x6E0[5:3]	OEM	Address port size	No	000	0xx – <= 24 bits 100 – 25 bits 101 – 26 bits 110 – 27 bits 111 – 28 bits
0x6E0[6]	OEM	ADV pin polarity	No	0	0 – Low active 1 – High active
0x6E0[7]	OEM	RDY pin polarity	No	0	0 – High active 1 – Low active
0x6E0[9:8]	OEM	AC timing parameter - CES	No	00	Value = (CES*5 + 1) cycle
0x6E0[11:10]	OEM	AC timing parameter - CEH	No	00	Value = (CEH*5 + 1) cycle
0x6E0[13:12]	OEM	AC timing parameter - CEITV	No	00	Value = (CEITV*5 + 1) cycle
0x6E0[15:14]	OEM	AC timing parameter - TA	No	00	Value = (TA*5 + 1) cycle
0x6E0[19:16]	OEM	AC timing parameter - AS	No	0000	Value = (AS + 1) cycle
0x6E0[23:20]	OEM	AC timing parameter - AH	No	0000	Value = (AH + 1) cycle
0x6E0[27:24]	OEM	AC timing parameter - REL	No	0000	Value = (REL + 1) cycle
0x6E0[31:28]	OEM	AC timing parameter - REH	No	0000	Value = (REH + 1) cycle

9.6.4.2 SEMC Parallel NOR Flash Boot Operation

The Boot ROM attempts to boot from Parallel NOR flash if the BOOT_CFG [7:4] fuses are programmed to 0b’0001 as shown in the Parallel NOR eFuse Configuration table. ROM supports booting from both XIP and non-XIP images from Parallel NOR Flash. For XIP boot, the image has to be built for SEMC address space and for non XIP the image can be built to execute from Internal RAM.

9.6.5 Parallel NAND flash Boot over SEMC

The boot ROM supports a number of Parallel SLC NAND flash devices from different vendors. Both the Error Correction and Control (ECC) module in NAND device and software ECC algorithm (SECDED) in boot ROM can be used to detect the errors, based on the fuse settings.

9.6.5.1 Parallel NAND eFuse Configuration

The boot ROM determines the configuration of external Parallel NAND flash by parameters, either provided by eFuse, or sampled on GPIO pins, during boot. See below table for parameters details:

Table 9-21. Fuse definition for Parallel NAND over SEMC

Fuse	Config	Definitions	GPIO	Shipped Value	Settings
BOOT_CFG[0]	OEM	Boot Search count of FCB and DBBT	Yes	0	0 - 1 1 – 2
BOOT_CFG[4:1]	OEM	Search stride for FCB and DBBT in terms of pages	Yes	0000	0000 – 64 Others – 2 ^ BOOT_SEARCH_STRIDE
BOOT_CFG[7:5]	OEM	Primary boot device selection	Yes	000	001 – Parallel NAND
BOOT_CFG[8]	OEM	NAND ONFI compliant	Yes	0	0 – ONFI 1.0 1 – Non-ONFI
BOOT_CFG [9]	OEM	ECC selection	Yes	0	0 – Software ECC (SECDED) 1 – Device ECC **NOTE:** For “ECC selection” option, it can only be set as Device ECC When NAND device has built-in ECC module and the ECC module is enabled by default.
BOOT_CFG [10]	OEM	DQS pin pad enablement	Yes	0	0 – Disabled 1 – Enabled
0x6E0[2:0]	OEM	PCS pin selection	No	000	000 – CSX0 001 – CSX1 010 – CSX2 011 – CSX3 1xx – A8
0x6E0[4]	OEM	EDO mode	No	0	0 – Disabled

Table continues on the next page...

Table 9-21. Fuse definition for Parallel NAND over SEMC (continued)

Fuse	Config	Definitions	GPIO	Shipped Value	Settings
					1 – Enabled
0x6E0[5]	OEM	RDY pin polarity	No	0	0 – High active 1 – Low active
0x6E0[6]	OEM	Ready check type	No	0	0 – R/B# pin 1 – Status Register
0x6E0[10:8]	OEM	Row Column address mode	No	000	Applicable only for Non-ONFI device 00x – 5 bytes (CA2+RA3) 010 – 4 bytes (CA2+RA2) 011 – 3 bytes (CA2+RA1) 10x – 4 bytes (CA1+RA3) 110 – 3 bytes (CA1+RA2) 111 – 2 bytes (CA1+RA1)
0x6E0[13:11]	OEM	Column address width	No	000	Applicable only for Non-ONFI device 000 – 12 bits 001 – 09 bits 010 – 10 bits 011 – 11 bits 100 – 13 bits 101 – 14 bits 110 – 15 bits 111 – 16 bits
0x6E0[14]	OEM	Status command type	No	0	Applicable only for Non-ONFI device 0 – Common (0x70) 1 – Enhanced (0x78)
0x6E0[18:16]	OEM	Pages in block	No	000	Applicable only for Non-ONFI device 000 – 128 pages 001 – 8 pages 010 – 16 pages 011 – 32 pages 100 – 64 pages 101 – 256 pages 110 – 512 pages 111 – 1024 pages
0x6E0[24]	OEM	Device ECC initial status	No	0	Applicable only for ONFI 1.0 device

Table 9-21. Fuse definition for Parallel NAND over SEMC

Fuse	Config	Definitions	GPIO	Shipped Value	Settings
					0 – Enabled 1 – Disabled

9.6.5.2 Parallel NAND Flash Boot Control Blocks (BCB)

There are two BCB data structures:

- Firmware Configuration Block (FCB)
- Discovered Bad Blocks Table (DBBT)

As part of the Parallel NAND media initialization, the ROM driver uses proper Parallel NAND parameters specified by FUSE to search for an FCB that contains the complete Parallel NAND parameters, page address of DBBT Search Area and Image info, including image copies, start page address and image size in terms of pages for each image.

FCB data structure is protected using Embedded ECC module in Parallel NAND devices or software ECC in ROM. The ROM driver reads 2048 bytes of first sector and checks the ECC check status to determine whether FCB data is valid or not.

If the FCB is found, the complete NAND parameters (Parallel NAND configuration block) are loaded for further reads, if the ECC fails, or the fingerprint does not match, or the CRC checksum does not match, the Block Search state machine increments page number to Search Stride number of pages to read for the next FCB until Search Count pages have been read.

If search fails to find a valid FCB, the Parallel NAND driver responds with an error and the boot ROM enters into Recovery boot mode (Secondary boot, if it is enabled, or Serial download mode).

The FCB contains the page address of DBBT Search Area, and the info for images. DBBT is searched in DBBT Search area just like how FCB is searched. After the FCB is read, the DBBT is loaded, then the boot image is loaded using starting page address from FCB.

9.6.5.3 Firmware Configuration Block (FCB)

The FCB is at the first page in the first good block. The FCB should be present at each search stride of the search area.

The search area contains copies of the FCB at each stride distance, in case the first Serial NAND block becomes corrupted, the ROM will find its copy in the next Parallel NAND block. The search area should span over at least two Parallel NAND blocks. The location information for DBBT search area and images are all specified in the FCB. Below Table shows the FCB Structure.

Name	offset	Size (Bytes)	Description
bcbHeader	0x000	12	See Table 9-22 for details
DBBTSerachAreaStartPage	0x00c	4	Start Page address for bad block table search area
searchStride	0x010	2	Search stride for DBBT and FCB search. Not used by ROM Max value is 8.
searchCount	0x012	2	Copies of DBBT and FCB. Not used by ROM, max value is 8.
firmwareCopies	0x014	4	Firmware copies Valid range 1-4.
-	0x018	40	Reserved
firmwareTable	0x040	64	For details see Table 9-23
-	0x080	128	Reserved
nandConfig	0x100	256	Parallel NAND configuration block over SEMC
-	0x200	512	Reserved

Table 9-22. Header Description

Field	Size	Description
crcChecksum	4	Checksum
fingerprint	4	0x4E46_4342 ASCII: “NFCB”
version	4	0x0000_0001

Table 9-23. Table Descriptions

Field	Size	Description
startPage	4	Start page of this firmware
pagesInFirmware	4	Pages in this firmware

NOTE

- The “crcChecksum” is calculated with an MPEG2 variant of CRC-32.

- The “crcChecksum” calculation starts from fingerprint to the end of FCB, 1020 bytes in total.
- The “nandConfig” is SEMC NAND configuration block which consists of common SEMC memory configuration block and Parallel NAND specified configuration parameters. See Parallel NAND eFuse Configuration for more details.

9.6.5.4 Discovered Bad Blocks Table (DBBT)

Table 9-24. DBBT Structure

Name	offset	Size (Bytes)	Description
bcbHeader	0x000	12	See Table 9-25 for details
-	0x00C	4	Reserved
badBlockNumber	0x010	4	Number of bad blocks
-	0x014	12	Reserved
badBlockTable	0x020	1024 (256*4)	Each bad block entry is a 32-bit value specifying the number of a found bad block. The number of valid bad block entries is specified by the badBlockNumber field, where valid bad block entries are stored sequentially starting at the beginning of the bad block entries segment. Unused bad block entries (those beyond the badBlockNumber) should be filled with 0xFs.

Table 9-25. Header Description

Field	Size	Description
crcChecksum	4	Checksum
fingerprint	4	0x4442_4254 ASCII: “DBBT”
version	4	0x0000_0001

NOTE

- Maximum bad block number is 256.
- The "crcChecksum" is calculated with the same algorithm as the one in FCB, from Fingerprint to the end of DBBT, 1052 bytes in total.

9.6.5.5 Bad block handling in ROM

During firmware boot, at the block boundary, the Bad Block table is searched for a match to the next block.

If no match is found, the next block can be loaded. If a match is found, the block must be skipped and the next block is checked.

9.6.6 Parallel NOR and NAND configuration based on SEMC interface

The ROM SW supports Parallel NOR and Parallel NAND based on SEMC module, using an 80-bytes common SEMC configuration block and several specified parameters for Parallel NOR and Parallel NAND respectively. See the following sections for more details.

9.6.6.1 SEMC Configuration Block

SEMC Configuration block consists of all parameters related to specific Flash devices.

Table 9-26. SEMC control block structure

Name	offset	Size (Bytes)	Description
tag	0x000	4	0x434D4553, ascii:"SEMC"
version	0x004	4	0x00010000 [07:00] bugfix = 0 [15:08] minor = 0 [31:16] major = 1
deviceMemType	0x008	1	0 – NOR Flash 1 – NAND Flash
accessCommandType	0x009	1	0 – IPG bus command 1 – AXI32 command
-	0x00A	2	Reserved
asyncClkFreq	0x00C	1	0 – 33MHz 1 – 40MHz 2 – 50MHz 3 – 66MHz 4 – 108MHz 5 – 133MHz 6 – 166MHz
busTimeoutCycles	0x00D	1	0 – 255 * 1024 cycles

Table continues on the next page...

Table 9-26. SEMC control block structure (continued)

Name	offset	Size (Bytes)	Description
			n – n * 1024 cycles
commandExecutionTimeoutCycles	0x00E	1	0 – 256 * 1024 cycles n – n * 1024 cycles
readStrobeMode	0x00F	1	0 – Dummy read strobe loopbacked internally 1 – Dummy read strobe loopbacked from DQS pad
norMemConfig	0x010	64	See detail in Table SEMC NOR control block structure
nandMemConfig	0x050	64	See detail in Table SEMC NAND control block structure

Table 9-27. SEMC NOR control block structure

Name	offset	Size (Bytes)	Description
comMemBaseAddress	0x00	4	SoC level Base Address for NAND AXI and IPG command
comMemSizeInByte	0x04	4	SoC level Memory size for NAND AXI and IPG command
-	0x08	8	Reserved
addressMode	0x10	1	0 – Address/Data MUX mode 1 – Advanced Address/Data MUX mode 2 – Address/Data non-MUX mode
addressPortWidth	0x11	1	Address Port bit number
dataPortWidth	0x12	1	Data Port bit number
columnAddressWidth	0x13	1	Column Address bit width
burstLengthInBytes	0x14	1	Burst Length
	0x15	3	Reserved
cePortOutputSelection	0x18	1	0 – CSX0 1 – CSX1 2 – CSX2 3 – CSX3 4 – A8
rdyPortPolarity	0x19	1	0 – Low active 1 – High active
advPortPolarity	0x1A	1	0 – Low active 1 – High active
-	0x1B	13	Reserved
ceSetupTime	0x28	1	value[3:0] + 1 cycles

Table continues on the next page...

Table 9-27. SEMC NOR control block structure (continued)

Name	offset	Size (Bytes)	Description
ceMinHoldTime	0x29	1	value[3:0] + 1 cycles
ceMinIntervalTime	0x2A	1	value[3:0] + 1 cycles
addressSetupTime	0x2B	1	value[3:0] + 1 cycles
addressHoldTime	0x2C	1	value[3:0] + 1 cycles
asyncWeLowTime	0x2D	1	value[3:0] + 1 cycles
asyncWeHighTime	0x2E	1	value[3:0] + 1 cycles
asyncOeLowTime	0x2F	1	value[3:0] + 1 cycles
asyncOeHighTime	0x30	1	value[3:0] + 1 cycles
asyncTurnaroundTime	0x31	1	value[3:0] + 1 cycles
asyncAddressToDataHoldTime	0x32	1	value[3:0] + 1 cycles
syncDataSetupTime	0x33	1	value[3:0] + 1 cycles
syncDataHoldTime	0x34	1	value[3:0] + 1 cycles
syncLatencyCount	0x35	1	value[3:0] + 1 cycles
syncReadCycleTime	0x36	1	value[3:0] + 1 cycles
-	0x37	9	Reserved

Table 9-28. SEMC NAND control block structure

Name	offset	Size (Bytes)	Description
axiMemBaseAddress	0x00	4	SoC level Base Address for NAND AXI command
axiMemSizeInByte	0x04	4	SoC level Memory size for NAND AXI command
ipgMemBaseAddress	0x08	4	SoC level Base Address for NAND IPG command
ipgMemSizeInByte	0x0c	4	SoC level Memory size for NAND IPG command
edoMode	0x10	1	0 - EDO mode disabled 1 - EDO mode enabled
ioPortWidth	0x11	1	IO Port bit number
arrayAddressOption	0x12	1	0 – 5 bytes (CA2+RA3) 1 – 4 bytes (CA1+RA3) 2 – 4 bytes (CA2+RA2) 3 – 3 bytes (CA1+RA2) 4 – 3 bytes (CA2+RA1) 7 – 2 bytes (CA1+RA1)
columnAddressWidth	0x13	1	Column address bit number
burstLengthInBytes	0x14	1	Burst Length
-	0x15	11	Reserved

Table continues on the next page...

Table 9-28. SEMC NAND control block structure (continued)

Name	offset	Size (Bytes)	Description
cePortOutputSelection	0x20	1	0 – CSX0 1 – CSX1 2 – CSX2 3 – CSX3 4 – A8
rdyPortPolarity	0x21	1	0 – Low active 1 – High active
-	0x22	14	Reserved
ceSetupTime	0x30	1	value[3:0] + 1 cycles
ceMinHoldTime	0x31	1	value[3:0] + 1 cycles
ceMinIntervalTime	0x32	1	value[3:0] + 1 cycles
weLowTime	0x33	1	value[3:0] + 1 cycles
weHighTime	0x34	1	value[3:0] + 1 cycles
reLowTime	0x35	1	value[3:0] + 1 cycles
reHighTime	0x36	1	value[3:0] + 1 cycles
weHighToReLowTime	0x37	1	value[5:0] + 1 cycles
reHighToWeLowTime	0x38	1	value[5:0] + 1 cycles
aleToDataStartTime	0x39	1	value[5:0] + 1 cycles
readyToReLowTime	0x3a	1	value[5:0] + 1 cycles
weHighToBusyTime	0x3b	1	value[5:0] + 1 cycles
asyncTurnaroundTime	0x3c	1	value[3:0] + 1 cycles
-	0x3d	3	Reserved

9.6.6.2 Parallel NOR Configuration Block (80 bytes)

Table 9-29. Parallel NOR control block structure

Name	offset	Size (Bytes)	Description
memConfig	0x000	80	See SEMC control block structure for more details

9.6.6.3 Parallel NAND Configuration Block (256 bytes)

Table 9-30. Parallel NAND control block structure

Name	offset	Size (Bytes)	Description
memConfig	0x000	80	See SEMC control block structure for more details

Table continues on the next page...

Table 9-30. Parallel NAND control block structure (continued)

Name	offset	Size (Bytes)	Description
vendorType	0x050	1	0 – Micron 1 – Spansion 2 – Samsung 3 – Winbond 4 – Hynix 5 – Toshiba 6 – Macronix
cellTechnology	0x051	1	0 – SLC 1 – MLC
onfiVersion	0x052	1	0 – Non-ONFI 1 – ONFI 1.0 2 – ONFI 2.0 3 – ONFI 3.0 4 – ONFI 4.0
acTimingTableIndex	0x053	1	0 – User Defined 1 – ONFI 1.0 Mode0 10MHz 2 – ONFI 1.0 Mode1 20MHz 3 – ONFI 1.0 Mode2 28MHz 4 – ONFI 1.0 Mode3 33MHz 5 – ONFI 1.0 Mode4 40MHz 6 – ONFI 1.0 Mode5 50MHz 7 – Auto Detection
enableEccCheck	0x054	1	0 – Enabled 1 – Disabled
eccCheckType	0x055	1	0 – Software ECC 1 – Device ECC
deviceEccStatus	0x056	1	0 – Enabled 1 – Disabled
swEccAlgorithm	0x057	1	0 – SEC Hamming Code
swEccBlockBytes	0x058	4	Software ECC block bytes (256, 512)
readyCheckOption	0x05c	1	0 – Via Status Register 1 – Via R/B# signal
statusCommandType	0x05d	1	0 – Common (0x70) 1 – Enhanced (0x78)
readyCheckTimeoutInMs	0x05e	2	Ready Check timeout
readyCheckIntervalInUs	0x060	2	Ready Check interval
-	0x062	62	Reserved

Table continues on the next page...

Table 9-30. Parallel NAND control block structure (continued)

Name	offset	Size (Bytes)	Description
-	0x062	30	Reserved
bytesInPageDataArea	0x0a0	4	Page Main data size
bytesInPageSpareArea	0x0a4	4	Page Spare data size
pagesInBlock	0x0a8	4	Page number in one block
blocksInPlane	0x0ac	4	Block number in one plane
planesInDevice	0x0b0	4	Plane number in Device
-	0x0b4	76	Reserved

9.6.7 Expansion device

The ROM supports booting from the MMC/eMMC and SD/eSD compliant devices.

9.6.7.1 Expansion device eFUSE configuration

The SD/MMC/eSD/eMMC/SDXC boot can be performed using the USDHC ports. The port can be configured based on either the setting of the BOOT_CFG1[1] (Port Select) fuse or its corresponding GPIO overrides.

All USDHC ports support the fast boot. See this table for details:

Table 9-31. USDHC eFuse Descriptions

Fuse	Config	Definition	GPIO	Shipped value	Settings
0x450[0]	OEM	Fast boot support	Yes	0	MMC 0 - Normal boot 1 - Fast boot
0x450[1]	OEM	USDHC port selection	Yes	0	0 - USDHC-1 1 - USDHC-2
0x450[2]	OEM	SD loopback clock source sel (for SDR50 and SDR104 only)	Yes	0	0 - through the SD pad 1 - direct
0x450[3]	OEM	SD power cycle enable	Yes	0	0 - No power cycle 1 - Power cycle enabled via the SD_RST pad
0x450[5:4]	OEM	SD/MMC speed mode, and eMMC acknowledge enabled selection	Yes	00	MMC 0x - Normal speed mode 1x - High-speed mode

Table continues on the next page...

Table 9-31. USDHC eFuse Descriptions (continued)

					x0 - eMMC fast boot acknowledge disable x1 - eMMC fast boot acknowledge enable SD 00 - Normal/SDR12 01 - High/SDR25 10 - SDR50 11 - SDR104
0x450[7:6]	OEM	Boot device selection	Yes	00	01 - SD/eSD/SDXC boot from the USDHC interface 10 - MMC/eMMC boot from the USDHC interface
0x450[8]	OEM	USDHC1 voltage selection	Yes	0	0 - 3.3 V 1 - 1.8 V
0x450[10:9]	OEM	SD MMC bus width selection	Yes	00	SD x0 - 1-bit x1 - 4-bit MMC 00 - 4-bit 01 - 8-bit 10 - 4-bit DDR (MMC 4.4) 11 - 8-bit DDR (MMC 4.4)
0x470[14]	OEM	eMMC4.4 pre-idle enabled selection	Yes	0	0 - Issue pre-idle command 1 - Do not issue
0x470[6]	OEM	USDHC1 reset polarity selection	Yes	0	0 - Reset active low 1 - Reset active high
0x460[29]	OEM	Power stable cycle selection	Yes	0	0 - 5 ms 1 - 2.5 ms
0x460[31:30]	OEM	Power cycle selection	Yes	00	00 - 20 ms 01 - 10 ms 10 - 5 ms 11 - 2.5 ms
0x460[24]	OEM	SD/MMC DLL enable selection	Yes	0	0 - Disable DLL for SD/eMMC 1 - Enable DLL for SD/Emm
0x470[7]	OEM	DLL override selection	Yes	0	0 - No override 1 - DLL override mode for SD/eMMC (Override by MMC_DLL_DLY, 0x470[19:16])
0X6D0[31:30]	OEM	SD calibration step	Yes	00	SD 00 - 1 delay cell

Table continues on the next page...

Table 9-31. USDHC eFuse Descriptions (continued)

					01 - 2 delay cells 10 - 4 delay cells 11 - 4 delay cells
0x470[0]	OEM	SD/MMC pad settings override selection	Yes	0	0 - No override 1 - Override (override by PAD_SETTINGS, 0x6d0[5:0])
0x470[3]	OEM	Disable SDMMC Manufacture mode	Yes	0	0 - Enable 1 - Disable
0x470[5]	OEM	USDHC2 voltage selection	Yes	0	0 - 3.3 V 1 - 1.8 V
0x470[8]	OEM	USDHC_IOMUX_SRE_ENABLE	Yes	0	0 - Disable 1 - Enable
0x470[9]	OEM	USDHC_IOMUX_SION_BIT_ENABLE	Yes	0	0 - Disable 1 - Enable
0x470[11]	OEM	ENABLE_EMMC_22K_PULLUP	Yes	0	MMC 0 - 47K pullup 1 - 22K pullup
0x470[12]	OEM	USDHC_PAD_PULL_DOWN	Yes	0	SD 0 - no action 1 - pull down
0x470[13]	OEM	Override HYS bit for SD/MMC pads	Yes	0	0 - No override 1 - Override HYS bit with 1
0x470[15]	OEM	USDHC2 reset polarity selection	Yes	0	0 - Reset active-low 1 - Reset active-high
0X470[30:24]	OEM	MMC_DLL_DLY	Yes	0000000	Override number
0X6D0[5:0]	OEM	PAD_SETTINGS	Yes	000000	Override number

The boot code supports these standards:

- MMCv4.4 or less
- eMMCv4.4 or less
- SDv2.0 or less
- eSDv2.10 rev-0.9, with or without FAST_BOOT
- SDXCv3.0

The MMC/SD/eSD/SDXC/eMMC can be connected to any of the USDHC blocks and can be booted by copying 4 KB of data from the MMC/SD/eSD/eMMC device to the internal RAM. After checking the Image Vector Table header value (0xD1) from program image, the ROM code performs a DCD check. After a successful DCD extraction, the ROM code extracts from the Boot Data Structure the destination pointer and length of image to be copied to the RAM device from where the code execution occurs.

The maximum image size to load into the SD/MMC boot is 32 MB. This is due to a limited number of uSDHC ADMA Buffer Descriptors allocated by the ROM.

NOTE

The initial 4 KB of the program image must contain the IVT, DCD, and the Boot Data structures.

Table 9-32. SD/MMC frequencies

	SD	MMC	MMC (DDR mode)
Identification (KHz)	347.22		
Normal-speed mode (MHz)	25	20	25
High-speed mode (MHz)	50	40	50
UHSI SDR50 (MHz)	100		
UHSI SDR104 (MHz)	200		

NOTE

The boot ROM code reads the application image length and the application destination pointer from the image.

9.6.7.2 MMC and eMMC boot

This table provides the MMC and eMMC boot details.

Table 9-33. MMC and eMMC boot details

Normal boot mode	During the initialization (normal boot mode), the MMC frequency is set to 347.22 KHz. When the MMC card enters the identification portion of the initialization, the voltage validation is performed, and the ROM boot code checks the high-voltage settings and the card capacity. The ROM boot code supports both the high-capacity and low-capacity MMC/eMMC cards. After the initialization phase is complete, the ROM boot code switches to a higher frequency (20 MHz in the normal boot mode or 40 MHz in the high-speed mode). The eMMC is also interfaced via the USDHC and follows the same flow as the MMC. The boot partition can be selected for an MMC4.x card after the card initialization is complete. The ROM code reads the BOOT_PARTITION_ENABLE field in the Ext_CSD[179] to get the boot partition to be set. If there is no boot partition mentioned in the BOOT_PARTITION_ENABLE field or the user partition was mentioned, the ROM boots from the user partition.
eMMC4.3 or eMMC4.4 device supporting special boot mode	If using an eMMC4.3 or eMMC4.4 device that supports the special boot mode, it can be initiated by pulling the CMD line low. If the BOOT ACK is enabled, the eMMC4.3/eMMC4.4 device sends the BOOT ACK via the DATA lines and the

Table continues on the next page...

Table 9-33. MMC and eMMC boot details (continued)

	ROM can read the BOOT ACK [S010E] to identify the eMMC4.3/eMMC4.4 device. The eMMC4.3/eMMC4.4 device with the "boot mode" feature can only be supported via the ESDHCV3-3 and with or without the BOOT ACK. If the BOOT ACK is enabled, the ROM waits 50 ms to get the BOOT ACK and if the BOOT ACK is received by the ROM. If BOOT ACK is disabled ROM waits 1 second for data. If the BOOT ACK or data was received, the eMMC4.3/eMMC4.4 is booted in the "boot mode", otherwise the eMMC4.3/eMMC4.4 boots as a normal MMC card from the selected boot partition. This boot mode can be selected by the BOOT_CFG1[4] (fast boot) fuse. The BOOT ACK is selected by the BOOT_CFG2[1].
eMMC4.4 device	If using the eMMC4.4 device, the Double Data Rate (DDR) mode can be used. This mode can be selected by the BOOT_CFG2[7:5] (bus width) fuse.

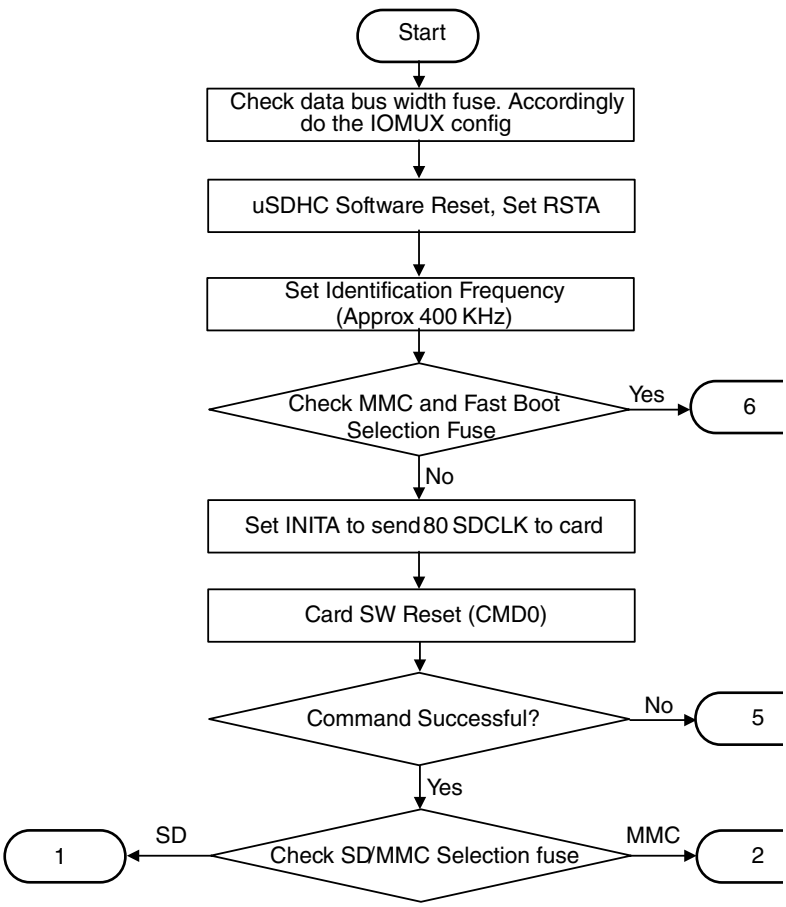


Figure 9-6. Expansion device boot flow (1 of 6)

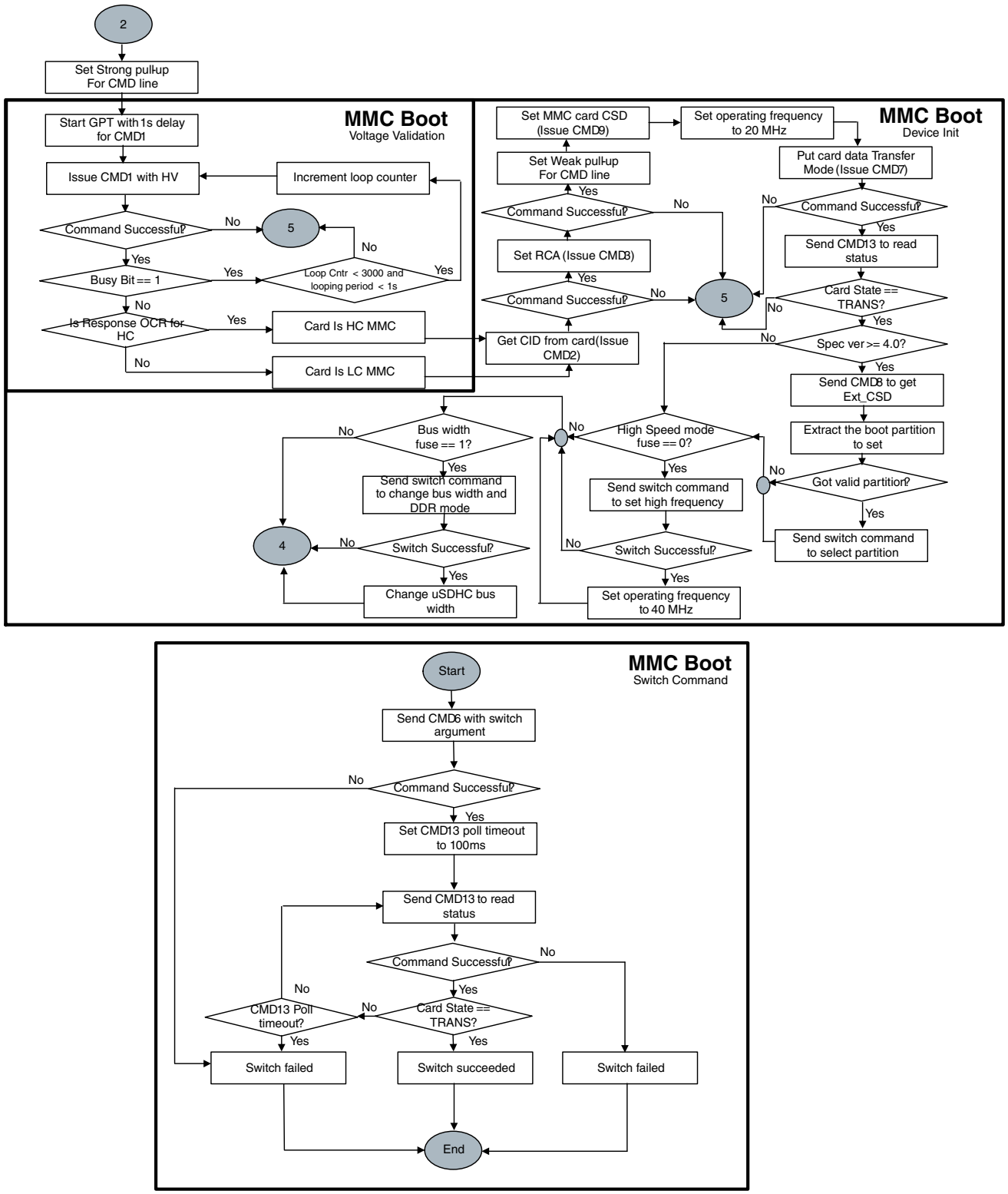


Figure 9-7. Expansion device (MMC) boot flow (2 of 6)

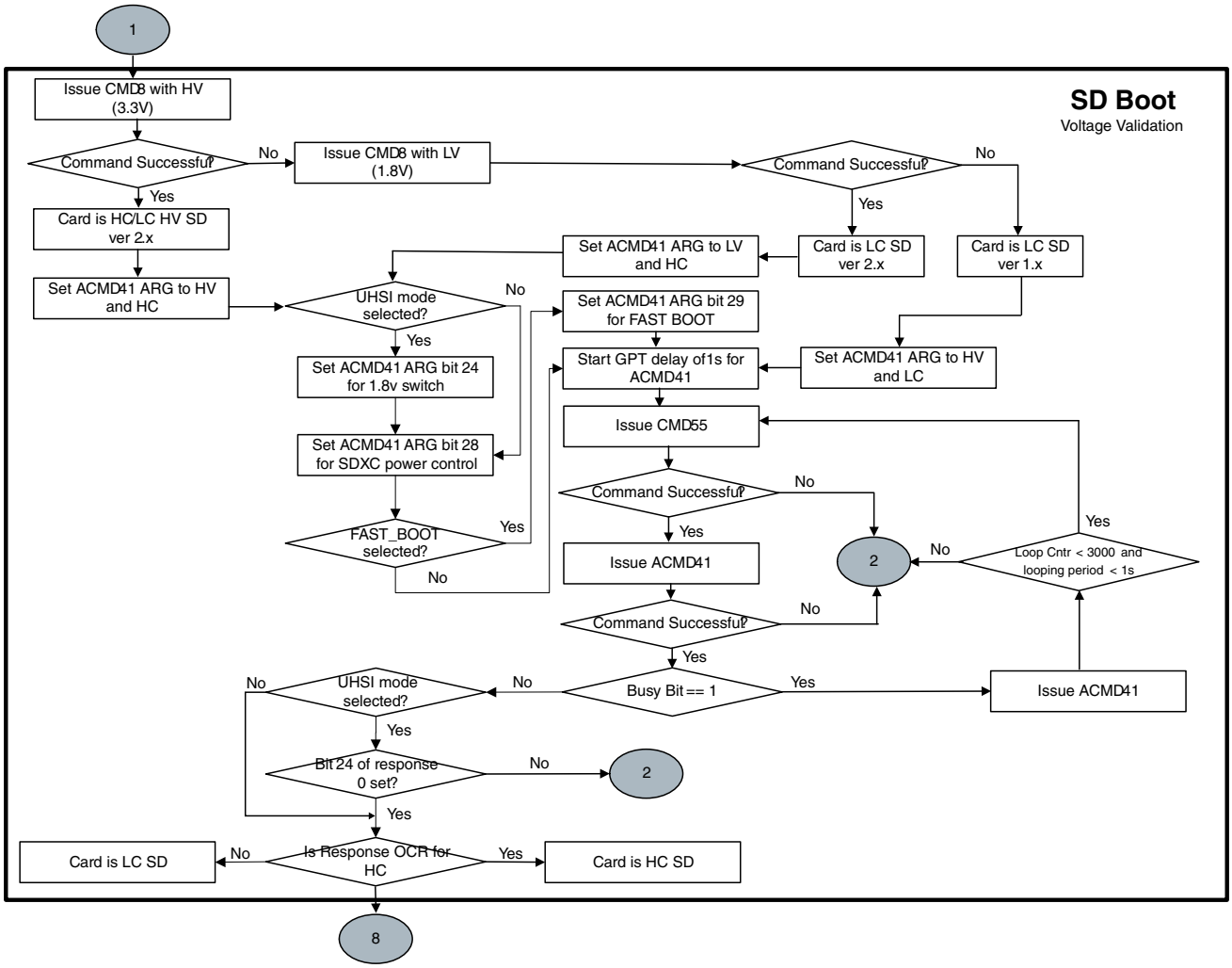


Figure 9-8. Expansion device (SD/eSD/SDXC) boot flow (3 of 6) part 1

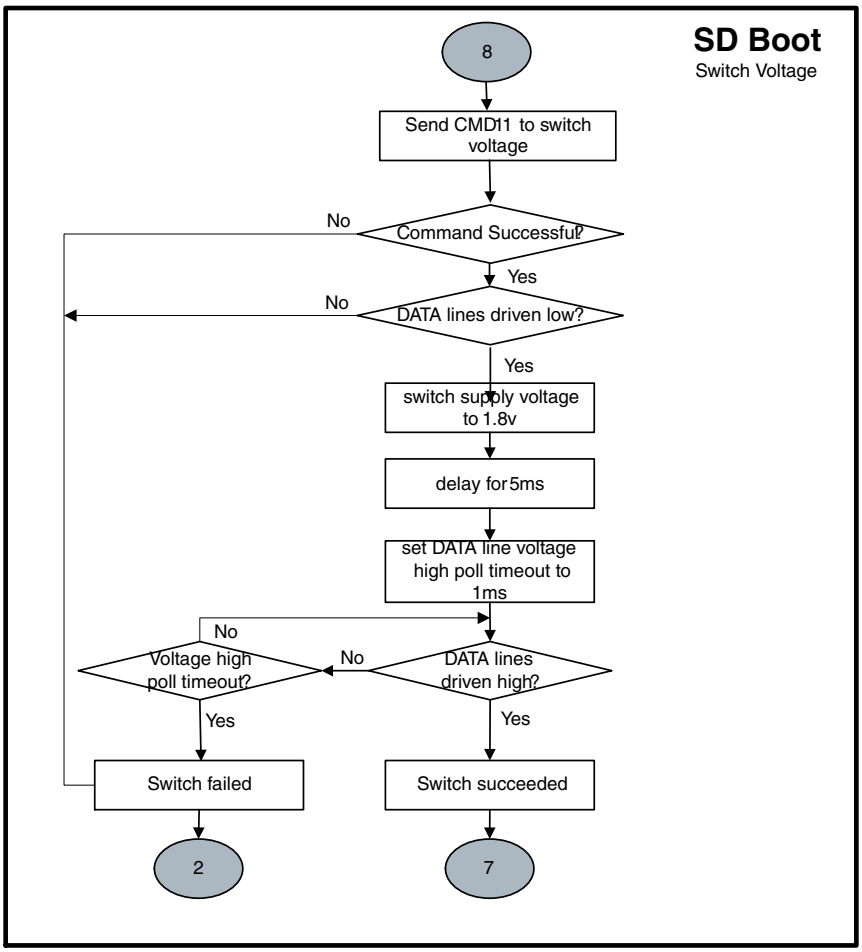


Figure 9-9. Expansion device (SD/eSD/SDXC) boot flow (3 of 6) part 2

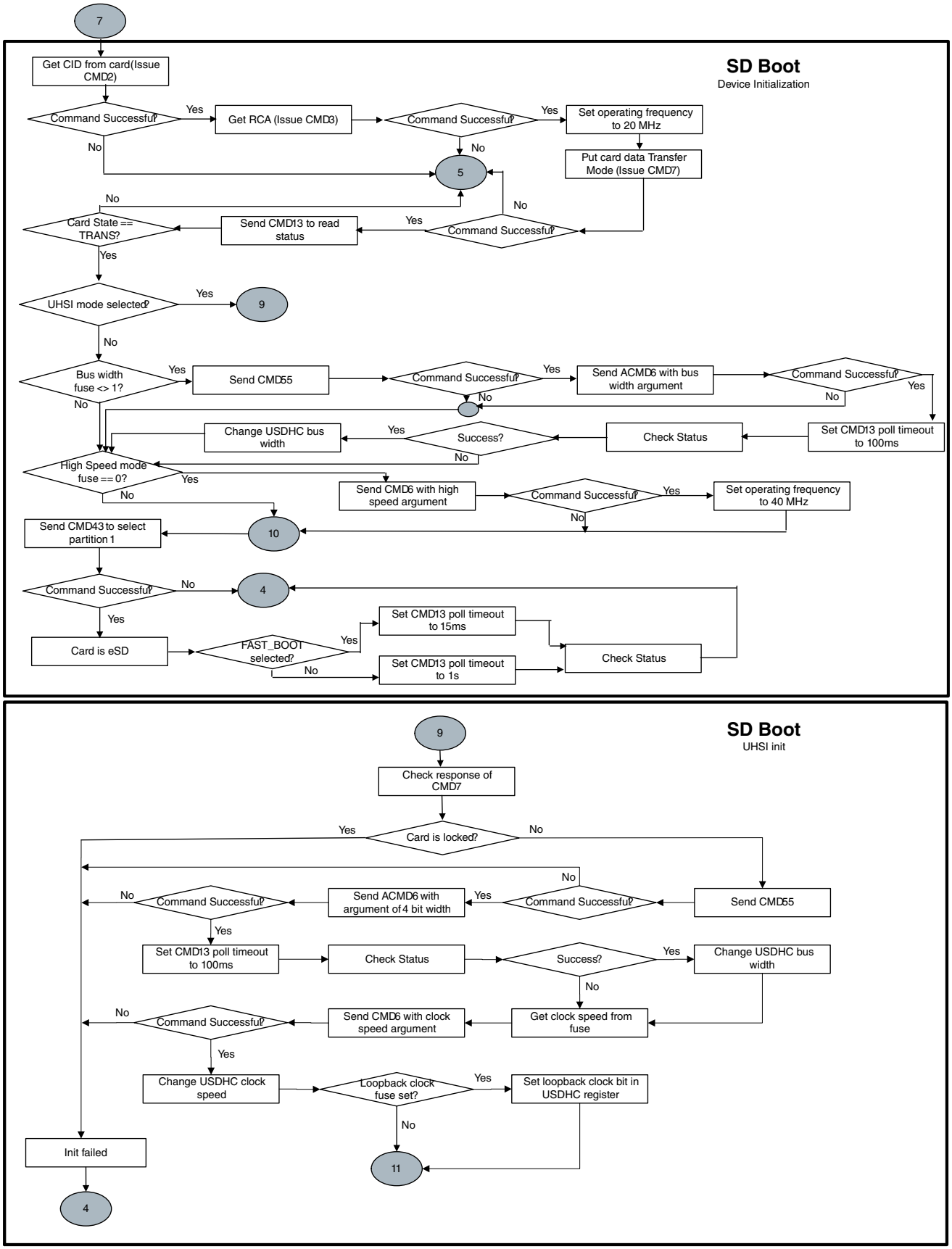


Figure 9-10. Expansion device (MMCSD/eSD/SDXC) boot flow (4 of 6)

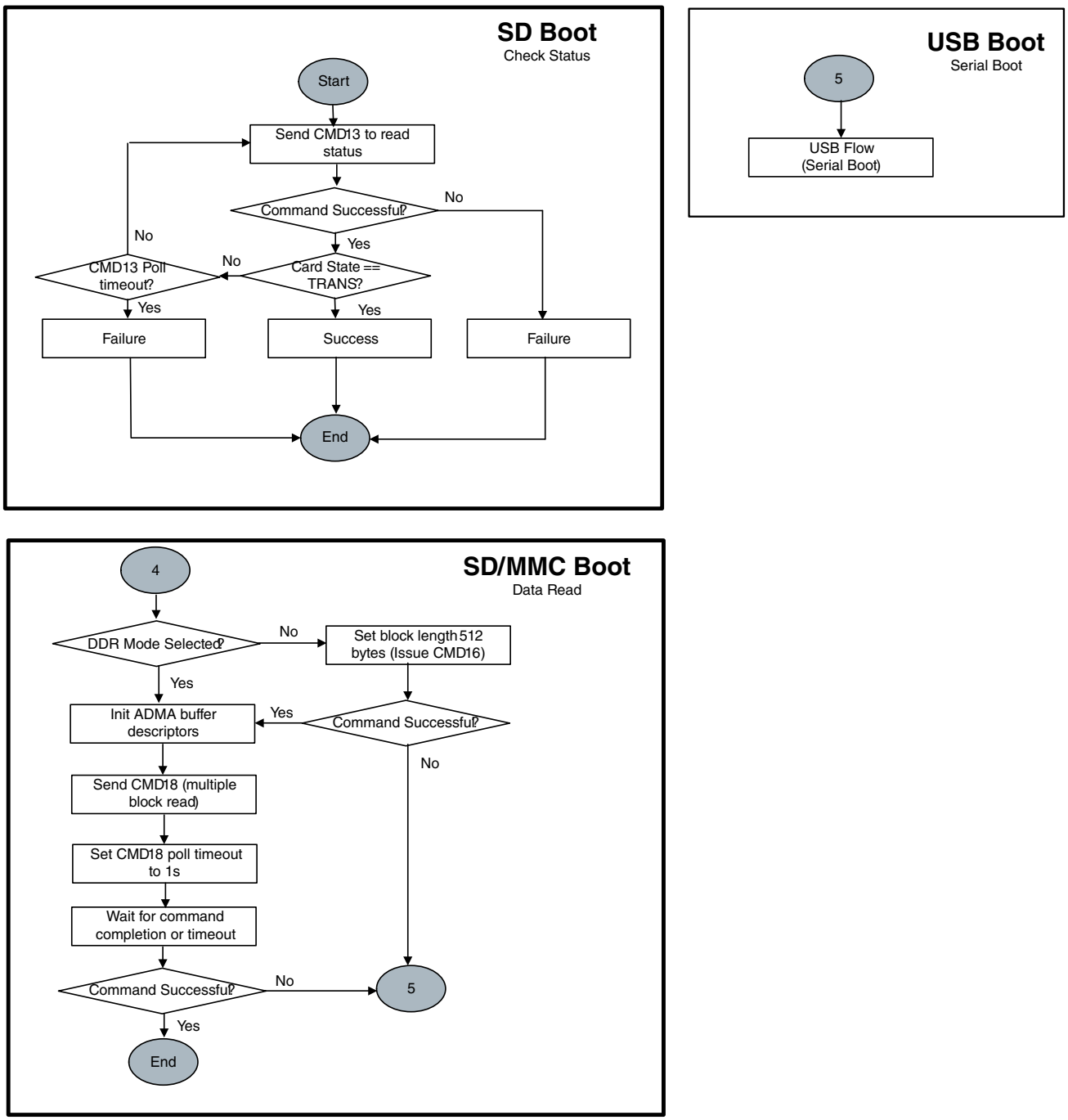
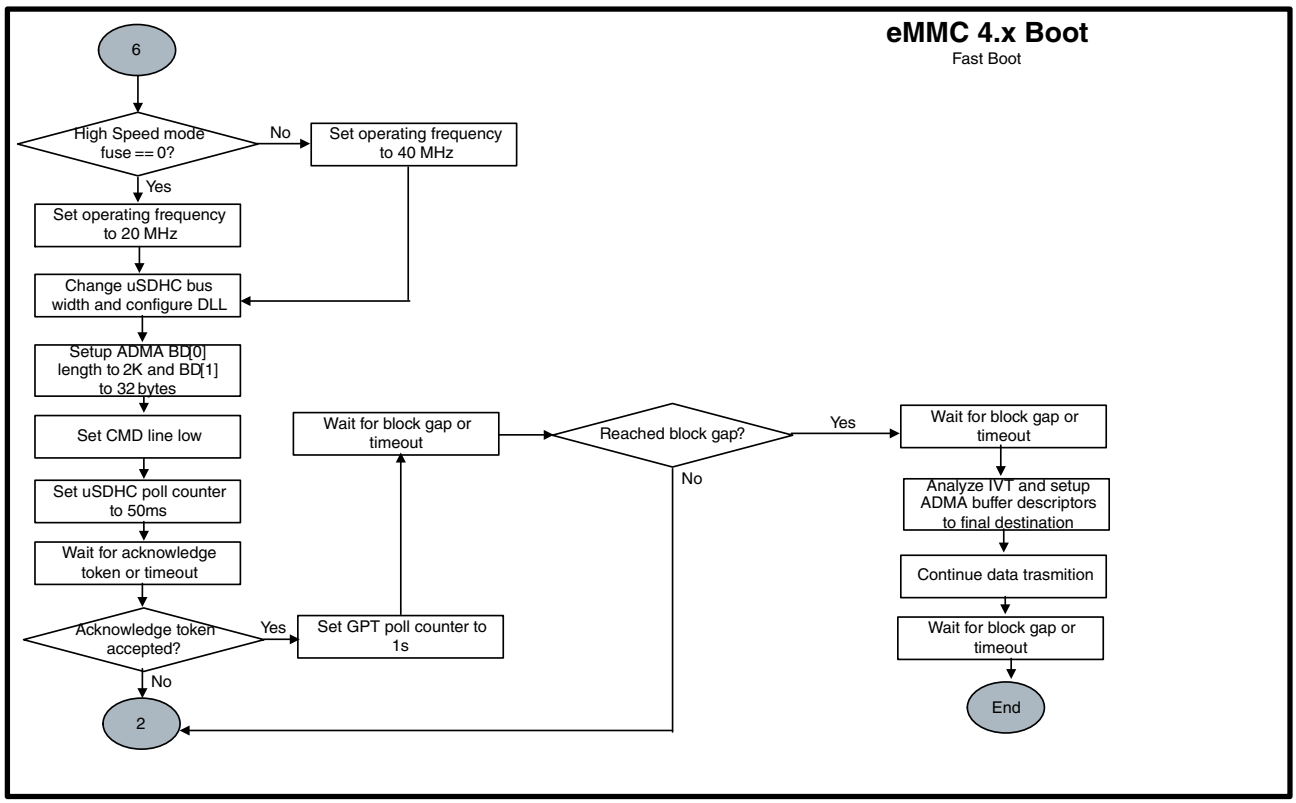


Figure 9-11. Expansion device (SD/eSD) boot flow (5 of 6)



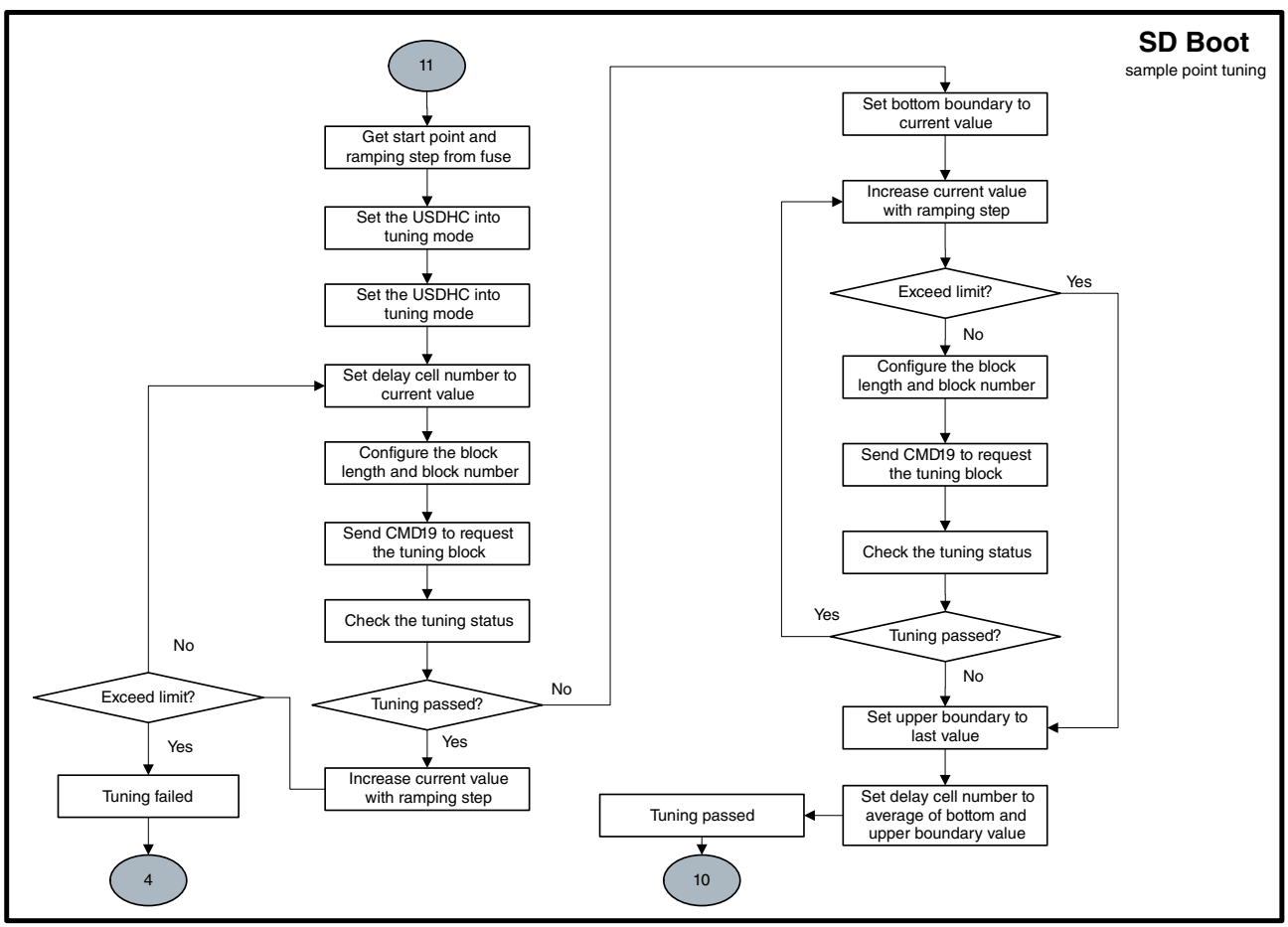


Figure 9-12. Expansion device boot flow (6 of 6)

9.6.7.3 SD, eSD, and SDXC

After the normal boot mode initialization begins, the SD/eSD/SDXC frequency is set to 347.22 kHz. During the identification phase, the SD/eSD/SDXC card voltage validation is performed. During the voltage validation, the boot code first checks with the high-voltage settings; if that fails, it checks with the low-voltage settings.

The capacity of the card is also checked. The boot code supports the high-capacity and low-capacity SD/eSD/SDXC cards after the voltage validation card initialization is done.

During the card initialization, the ROM boot code attempts to set the boot partition for all SD, eSD, and SDXC devices. If this fails, the boot code assumes that the card is a normal SD or SDXC card. If it does not fail, the boot code assumes it is an eSD card. After the initialization phase is over, the boot code switches to a higher frequency (25 MHz in the normal-speed mode or 50 MHz in the high-speed mode). The ROM also supports the FAST_BOOT mode booting from the eSD card. This mode can be selected by the BOOT_CFG1[0] (Fast Boot).

For the UHSI cards, the clock speed fuses can be set to SDR50 or SDR104 on USDHC1, USDHC2 ports. This enables the voltage switch process to set the signaling voltage to 1.8 V during the voltage validation. The bus width is fixed at a 4-bit width and a sampling point tuning process is needed to calibrate the number of the delay cells. If the SD Loopback Clock eFuse is set, the feedback clock comes directly from the loopback SD clock, instead of the card clock (by default). The SD clock speed can be selected by the BOOT_CFG1[5:4], and the SD Loopback Clock is selected by the BOOT_CFG1[2].

The UHSI calibration start value (MMC_DLL_DLY[6:0]) and the step value SD_CALIBRATION_STEP[1:0] can be set to optimize the sample point tuning process.

If the SD Power Cycle Enable eFuse is 1, the ROM sets the SD_RST pad low, waits for 5 ms, and then sets the SD_RST pad high. If the SD_RST pad is connected to the SD power supply enable logic on board, it enables the power cycle of the SD card. This may be crucial in case the SD logic is in the 1.8 V states and must be reset to the 3.3 V states.

The SDR50 and SDR104 boots are not supported on the USDHC1 and USDHC2 ports because there are no reset signals for those ports when connected in the IOMUX.

9.6.7.4 Redundant boot support for expansion device

The ROM supports the redundant boot for an expansion device. The primary or secondary image is selected, depending on the PERSIST_SECONDARY_BOOT setting. (see Table 9-8).

If the PERSIST_SECONDARY_BOOT is 0, the boot ROM uses address 0x0 for the primary image.

If the PERSIST_SECONDARY_BOOT is 1, the boot ROM reads the secondary image table from address 0x200 on the boot media and uses the address specified in the table.

Table 9-34. Secondary image table format

Reserved (chipNum)
Reserved (driveType)
tag
firstSectorNumber
Reserved (sectorCount)

Where:

- The tag is used as an indication of the valid secondary image table. It must be 0x00112233.
- The firstSectorNumber is the first 512-byte sector number of the secondary image.

For the secondary image support, the primary image must reserve the space for the secondary image table. See this figure for the typical structures layout on an expansion device.

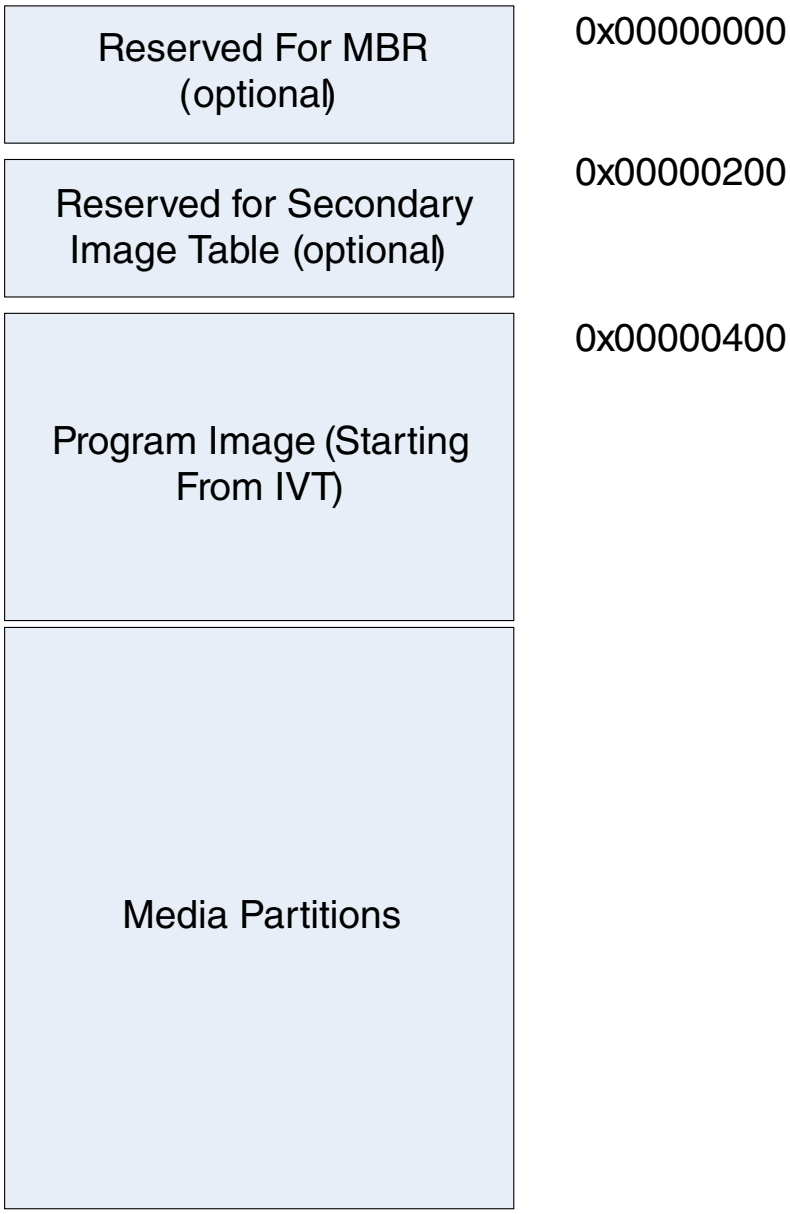


Figure 9-13. Expansion device structures layout

For the Closed mode, if there are failures during primary image authentication, the boot ROM turns on the PERSIST_SECONDARY_BOOT bit (see Table 9-8) and performs the software reset. (After the software reset, the secondary image is used.)

9.6.8 Serial NOR/EEPROM through LPSPI

The chip supports booting from serial memory devices, such as EEPROM and serial flash, using the LPSPI.

These ports are available for serial boot: LPSPI interfaces.

9.6.8.1 Serial NOR/EEPROM eFUSE configuration

The boot ROM code determines the type of device using the following parameters, provided by the eFUSE settings during boot.

Table 9-35. Serial NOR/EEPROM boot eFUSE descriptions

Fuse	Config	Definition	GPIO	Shipped value	Settings
EEPROM_RECOVERY_EN(0x6D0[24])	OEM	EEPROM recovery enable	No	0	0 - Disabled 1 - Enabled
LPSPI_PORT_SEL(0x6D0[26:25])	OEM	Port select	No	00	00 - LPSPI1 01 - LPSPI2 10 - LPSPI3 (if applicable in the device) 11 - LPSPI4 (if applicable in the device)
LPSPI_ADDR(0x6D0[27])	OEM	SPI addressing (SPI only)	No	0	0 - 3B (24-bit) 1 - 2B (16-bit)
LPSPI_SPEED (0x6D0[29:28])	OEM	LPSPI Speed select	No	00	00 - 20 MHz 01 - 10 MHz 10 - 5 MHz 11 - 2 MHz

The LPSPI*n* block can be used as a boot device using the LPSPI interface for the serial ROM boot. The SPI interface is configured to operate at speed specified by LPSPI_SPED_SEL fuse field.

The boot ROM copies 4 KB of data from the serial ROM device to the internal RAM. After checking the Image Vector Table header value (0xD1) from the program image, the ROM code performs a DCD check. After a successful DCD extraction, the ROM code extracts the destination pointer and length of image from the Boot Data Structure to be copied to the RAM device from where the code execution occurs.

NOTE

The Initial 4 KB of program image must contain the IVT, DCD, and the Boot Data Structures.

9.7 Program image

This section describes the data structures that are required to be included in the user's program image. The program image consists of:

- Image vector table—a list of pointers located at a fixed address that the ROM examines to determine where the other components of the program image are located.
- Boot data—a table that indicates the program image location, program image size in bytes, and the plugin flag.
- Device configuration data—IC configuration data.
- User code and data.

9.7.1 Image Vector Table and Boot Data

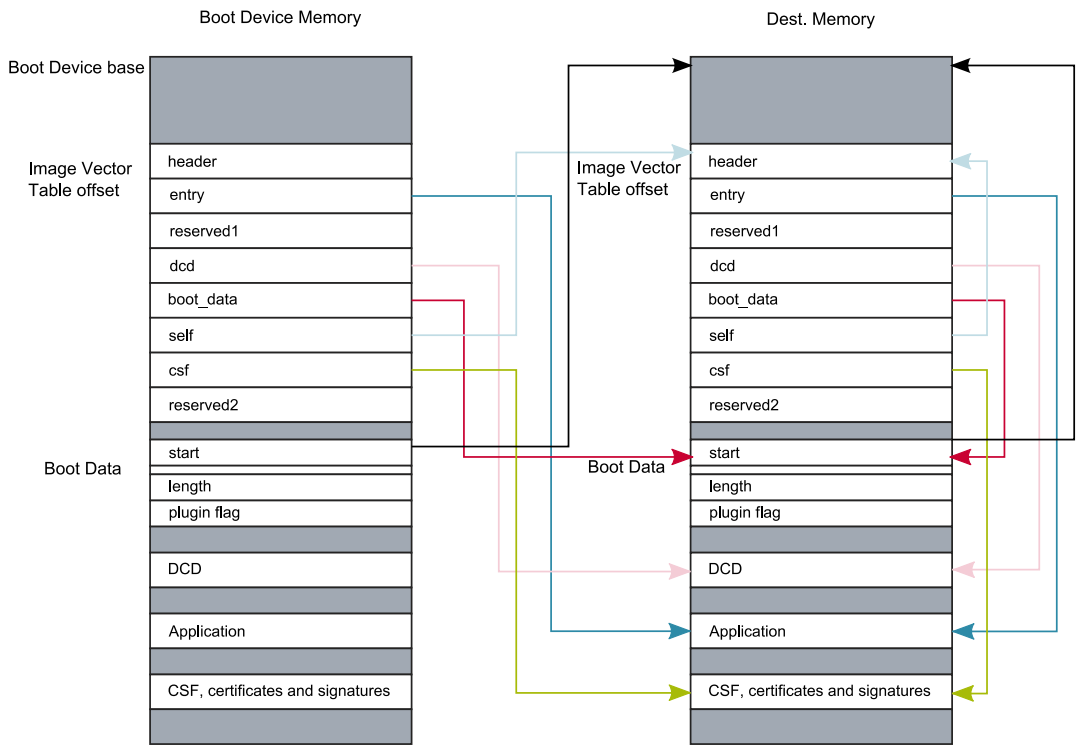
The Image Vector Table (IVT) is the data structure that the ROM reads from the boot device supplying the program image containing the required data components to perform a successful boot.

The IVT includes the program image entry point, a pointer to Device Configuration Data (DCD) and other pointers used by the ROM during the boot process. The ROM locates the IVT at a fixed address that is determined by the boot device connected to the Chip. The IVT offset from the base address for each boot device type is defined in the table below. The location of the IVT is the only fixed requirement by the ROM. The remainder or the image memory map is flexible and is determined by the contents of the IVT.

Table 9-36. Image Vector Table Offset and Initial Load Region Size

Boot Device Type	Image Vector Table Offset
FlexSPI NOR/SEMC NOR	4 Kbyte = 0x1000 bytes
SD/MMC/eSD/eMMC/SDXC	1 Kbyte = 0x400 bytes
SPI NOR/EEPROM SEMC NAND FlexSPI NAND	1 Kbyte = 0x400 bytes

IVT is at 0x60001000 because Teensy boots from FlexSPI NOR



This figure is rather confusing. It's probably leftover copy-paste from NXP's application processors. For Teensy boot, the "Boot Device Memory" and "Dest. Memory" are the same. They are not different physical memory or different addresses as this figure implies.

Figure 9-14. Image Vector Table

9.7.1.1 Image vector table structure

The IVT has the following format where each entry is a 32-bit word:

Table 9-37. IVT format

header
entry: Absolute address of the first instruction to execute from the image LSB=1 for Thumb mode
reserved1: Reserved and should be zero
dcd: Absolute address of the image DCD. The DCD is optional so this field may be set to NULL if no DCD is required. See Device Configuration Data (DCD) for further details on the DCD.
boot data: Absolute address of the boot data
self: Absolute address of the IVT. Used internally by the ROM
csf: Absolute address of the Command Sequence File (CSF) used by the HAB library. See High-Assurance Boot (HAB) for details on the secure boot using HAB. This field must be set to NULL if a CSF is not provided in the image
reserved2: Reserved and should be zero

Figure 9-15 shows the IVT header format:

Figure 9-15. IVT header format

where:

```
Tag: A single byte field set to 0xD1
Length: a two byte field in big endian format containing the overall length of the IVT,
in bytes, including the header. (the length is fixed and must have a value of 32 bytes)
Version: A single byte field set to 0x40/0x41
```

9.7.1.2 Boot data structure

The boot data must follow the format defined in the table found here, each entry is a 32-bit word.

Table 9-38. Boot data format

start	Absolute address of the image
length	Size of the program image
plugin	Plugin flag (see Plugin image)

9.7.2 Device Configuration Data (DCD)

Upon reset, the chip uses the default register values for all peripherals in the system. However, these settings typically are not ideal for achieving the optimal system performance and there are even some peripherals that must be configured before they can be used.

The DCD is a configuration information contained in the program image (external to the ROM) that the ROM interprets to configure various peripherals on the chip.

For example, some components (such as SDRAM) require some sequence of register programming as a part of the configuration before it is ready to be used. The DCD feature can be used to program the SEMC register to the optimal settings.

The ROM determines the location of the DCD table based on the information located in the Image Vector Table (IVT). See Image Vector Table and Boot Data for more details. The DCD table shown below is a big-endian byte array of the allowable DCD commands. The maximum size of the DCD is limited to 1768 B.

Figure 9-16. DCD data format

The DCD header is 4 B with the following format:

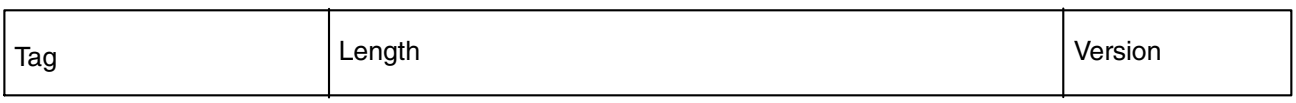


Figure 9-17. DCD header format

where:

```
Tag: A single-byte field set to 0xD2
Length: a two-byte field in the big-endian format containing the overall length of the DCD
(in bytes) including the header
Version: A single-byte field set to 0x41
```

9.7.2.1 Write data command

The write data command is used to write a list of given 1-, 2- or 4-byte values (or bitmasks) to a corresponding list of target addresses.

The Value/Mask fields are always 32-bits. If the parameter field specifies a smaller size, then the extra bytes must be zero.

The format of the write data command (in a big-endian byte array) is shown in this table:

Table 9-39. Write data command format

Tag	Length	Parameter
Address		
Value/Mask		
[Address]		
[Value/Mask]		

Table continues on the next page...

Table 9-39. Write data command format (continued)

...
[Address]
[Value/Mask]

where:

```
Tag: a single-byte field set to 0xCC
Length: a two-byte field in a big-endian format, containing the length of the Write Data
Command (in bytes) including the header
Address: the target address to which the data must be written
Value/Mask: the data value (or bitmask) to be written to the preceding address
```

The parameter field is a single byte divided into bitfields as follows:

Table 9-40. Write data command parameter field

7	6	5	4	3	2	1	0
flags					bytes		

where

```
bytes: the width of the target locations in bytes (either 1, 2, or 4)
flags: control flags for the command behavior
Data Mask = bit 3: if set, only specific bits may be overwritten at the target address
(otherwise all bits may be overwritten)
Data Set = bit 4: if set, the bits at the target address are overwritten with this flag
(otherwise it is ignored)
```

One or more target address and value/bitmask pairs can be specified. The same bytes' and flags' parameters apply to all locations in the command.

When successful, this command writes to each target address in accordance with the flags as follows:

Table 9-41. Interpretation of write data command flags

"Mask"	"Set"	Action	Interpretation
0	0	*address = val_msk	Write value
0	1	*address = val_msk	Write value
1	0	*address &= ~val_msk	Clear bitmask
1	1	*address |= val_msk	Set bitmask

NOTE

If any of the target addresses does not have the same alignment as the data width indicated in the parameter field, none of the values are written.

If any of the values are larger or any of the bitmasks are wider than permitted by the data width indicated in the parameter field, none of the values are written.

If any of the target addresses do not lie within the allowed region, none of the values are written. The list of allowable blocks and target addresses for the chip are provided below.

Table 9-42. Valid DCD address ranges

Address range	Start address	Last address
IOMUX Control SNVS GPR (IOMUXC_SNVS_GPR)registers	0x400A_4000	0x400A_7FFF
IOMUX Control SNVS (IOMUXC_SNVS)registers	0x400A_8000	0x400A_BFFF
IOMUX Control GPR (IOMUXC_GPR)registers	0x400A_C000	0x400A_FFFF
IOMUX Control (IOMUXC) registers	0x401F_8000	0x401F_BFFF
CCM Analog (ANATOP) registers	0x400D_8000	0x400D_BFFF
CCM registers	0x400F_C000	0x400F_FFFF
SEMC registers	0x402F_0000	0x402F_3FFF

9.7.2.2 Check data command

The check data command is used to test for a given 1-, 2-, or 4-byte bitmasks from a source address.

The check data command is a big-endian byte array with the format shown in this table:

Table 9-43. Check data command format

Tag	Length	Parameter
Address		
Mask		
[Count]		

where:

```
Tag: a single-byte field set to 0xCF
Length: a two-byte field in the big-endian format containing the length of the check data
command (in bytes) including the header
Address: the source address to test
Mask: the bit mask to test
Count: an optional poll count; If the count is not specified, this command polls
indefinitely
until the exit condition is met. If count = 0, this command behaves as for the NOP.
```

The parameter field is a single byte divided into bitfields, as follows:

Table 9-44. Check data command parameter field

7	6	5	4	3	2	1	0
flags					bytes		

where

```
bytes: the width of target locations in bytes (either 1, 2, or 4)
flags: control flags for the command behavior
Data Mask = bit 3: if set, only the specific bits may be overwritten at a target address
(otherwise all bits may be overwritten)
Data Set = bit 4: if set, the bits at the target address are overwritten with this flag
(otherwise it is ignored)
```

This command polls the source address until either the exit condition is satisfied, or the poll count is reached. The exit condition is determined by the flags as follows:

Table 9-45. Interpretation of check data command flags

"Mask"	"Set"	Action	Interpretation
0	0	(*address & mask) == 0	All bits clear
0	1	(*address & mask) == mask	All bits set
1	0	(*address & mask)!= mask	Any bit clear
1	1	(*address & mask)!= 0	Any bit set

NOTE

If the source address does not have the same alignment as the data width indicated in the parameter field, the value is not read.

If the bitmask is wider than permitted by the data width indicated in the parameter field, the value is not read.

9.7.2.3 NOP command

This command has no effect.

The format of the NOP command is a big-endian four-byte array, as shown in this table:

Table 9-46. NOP command format

Tag	Length	Undefined

where:

```
Tag: a single-byte field set to 0xC0
Length: a two-byte field in big endian containing the length of the NOP command in bytes
(fixed to a
```

```
value of 4)
Undefined: this byte is ignored and can be set to any value.
```

9.8 Plugin image

The ROM supports a limited number of boot devices. When using other devices as a boot source (for example, Ethernet or USB), the supported boot device must be used (typically serial flash) as a firmware to provide the missing boot drivers. Additionally, the plugin can be customized to support boot drivers, which is more flexible when performing the device initialization, such as condition judging, delay assertion, or to apply custom settings to the boot device and memory system.

In addition to the standard images, the chip also supports plugin images. The plugin images return the execution to the ROM whereas the standard image does not.

The boot ROM detects the image type using the plugin flag of the boot data structure (see Boot data structure). If the plugin flag is 1, then the ROM uses the image as a plugin function. The function must initialize the boot device and copy the program image to the final location. At the end, the plugin function must return with the program image parameters. (See High-level boot sequence for details about the boot flow).

The boot ROM authenticates the plugin image before running the plugin function and then authenticates the program image.

The plugin function must follow the API described below:

typedef BOOLEAN (*plugin_download_f)(void **start, size_t *bytes, UINT32 *ivt_offset);

ARGUMENTS PASSED:

- start - the image load address on exit.
- bytes - the image size on exit.
- ivt_offset - the offset (in bytes) of the IVT from the image start address on exit.

RETURN VALUE:

- 1 - success
- 0 - failure

9.9 Serial Downloader

The Serial Downloader provides a means to download a program image to the chip over the USB and UART serial connection.

In this mode, the ROM programs the WDOG1 for a time-out specified by the fuse WDOG Time-out Select (See the Fusemap chapter for details) if the WDOG_ENABLE eFuse is 1 and continuously polls for the USB and UART connection. If no activity is found on the USB OTG1 and UART and the watchdog timer expires, the Arm core is reset.

NOTE

After the downloaded image is loaded, it is responsible for managing the watchdog resets properly.

This figure shows the USB and UART boot flow:

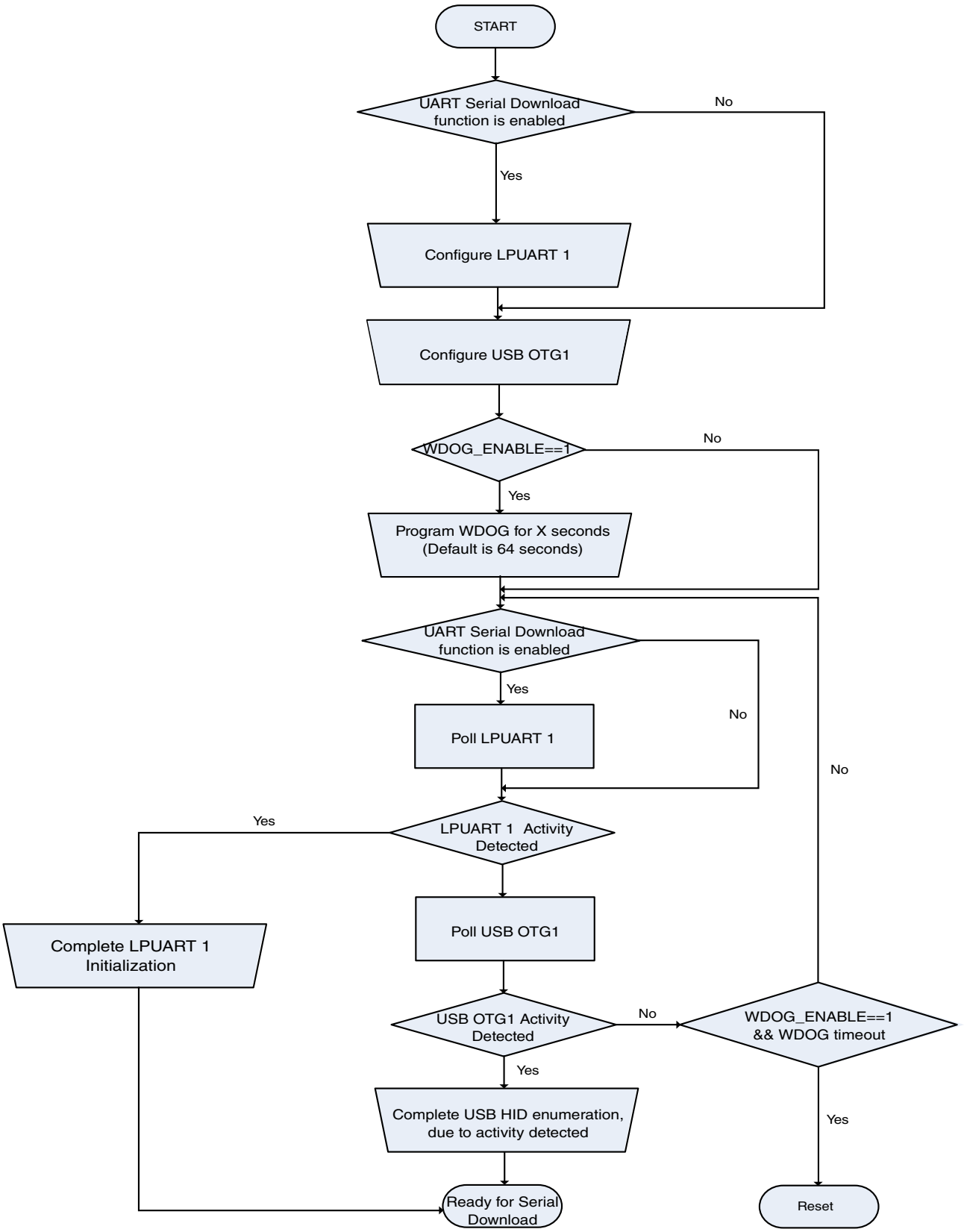


Figure 9-18. Serial Downloader boot flow

9.9.1 USB

The USB support is composed of the USB1 (USB OTG1 core controller, compliant with the USB 2.0 specification) and the USBPHY (HS USB transceiver).

The ROM supports the USB1 port for boot purposes. The other USB ports on the chip are not supported for boot purposes.

The USB Driver is implemented as a USB HID class. A collection of four HID reports are used to implement the SDP protocol for data transfers, as described in Table 9-47.

Table 9-47. USB HID reports

Report ID (first byte)	Transfer endpoint	Direction	Length	Description
1	control OUT	Host to device	17 B	SDP command from the host to the device.
2	control OUT	Host to device	Up to 1025 B	Data associated with the report 1 SDP command.
3	interrupt	Device to host	5 B	HAB security configuration. The device sends 0x12343412 in the closed mode and 0x56787856 in the open mode.
4	interrupt	Device to host	Up to 65 B	Data in response to the SDP command in report 1.

9.9.1.1 USB configuration details

The USB OTG function device driver supports a high speed (HS for UTMI) non-stream mode with a maximal packet size of 512 B and a low-level USB OTG function.

The VID/PID and strings for the USB device driver are listed in the following table.

Table 9-48. VID/PID and strings for USB device driver

Descriptor	Value
VID	0x1FC9 (NXP vendor ID)
PID[1]	0x0135
String Descriptor1 (manufacturer)	NXP Semiconductors
String Descriptor2 (product)	S Blank

Table continues on the next page...

Table 9-48. VID/PID and strings for USB device driver (continued)

Descriptor	Value
String Descriptor4	NXP Flash
String Descriptor5	NXP Flash

1. Allocation based on the BPN (Before Part Number)

9.9.1.2 IOMUX configuration for USB

The interface signals of the UTMI PHY are not configured in the IOMUX. The UTMI PHY interface uses the dedicated contacts on the IC. See the chip data sheet for details.

9.9.2 Serial Download Protocol (SDP)

The 16-byte SDP command from the host to device is sent using the HID report 1. This table describes the 16-byte SDP command data structure:

Table 9-49. 16-byte SDP command data structure

BYTE offset	Size	Name	Description
0	2	COMMAND TYPE	These commands are supported for the ROM: • 0x0101 READ_REGISTER • 0x0202 WRITE_REGISTER • 0x0404 WRITE_FILE • 0x0505 ERROR_STATUS • 0x0A0A DCD_WRITE • 0x0B0B JUMP_ADDRESS • 0x0D0D SET_BAUDRATE (only applicable to UART)
2	4	ADDRESS	Only relevant for these commands: READ_REGISTER, WRITE_REGISTER, WRITE_FILE, DCD_WRITE, and JUMP_ADDRESS. For the READ_REGISTER and WRITE_REGISTER commands, this field is the address to a register. For the WRITE_FILE and JUMP_ADDRESS commands, this field is an address to the internal or external memory address. **NOTE:** For SET_BAUDRATE command, this word is the baudrate value in big-endian format.
6	1	FORMAT	Format of access, 0x8 for an 8-bit access, 0x10 for a 16-bit access, and 0x20 for a 32-bit access. Only relevant for the READ_REGISTER and WRITE_REGISTER commands.

Table continues on the next page...

Table 9-49. 16-byte SDP command data structure (continued)

BYTE offset	Size	Name	Description
7	4	DATA COUNT	Size of the data to read or write in bytes. Only relevant for the WRITE_FILE, READ_REGISTER, WRITE_REGISTER, and DCD_WRITE commands. For the WRITE_FILE and DCD_WRITE commands, the DATA COUNT is in the byte units.
11	4	DATA	The value to write. Only relevant for the WRITE_REGISTER command.
15	1	RESERVED	Reserved

9.9.2.1 SDP commands

The SDP commands are described in the following sections.

9.9.2.1.1 READ_REGISTER

The transaction for the READ_REGISTER command consists of these reports: Report1 for the command, Report3 for the security configuration, and Report4 for the response or the register value.

The register to read is specified in the ADDRESS field of the SDP command. The first device sends Report3 with the security configuration followed by the Report4 with the bytes read at a given address. If the count is greater than 64, multiple reports with the report id 4 are sent until the entire data requested by the host is sent. The STATUS is either 0x12343412 for the closed parts and 0x56787856 for the open or field return parts.

Report1, Command, Host to Device:

1	Valid values for the READ_REGISTER COMMAND, ADDRESS, FORMAT, DATA_COUNT

ID 16-byte SDP command

Report3, Response, Device to Host:

3	Four bytes indicating the security configuration

ID 4 bytes status

Report4, Response, Device to Host: first response report

4	Register value

ID 4 bytes of data containing the register value. If the number of bytes requested is less than 4, the remaining bytes must be ignored by the host.

Multiple reports of the report id 4 are sent until the entire requested data is sent.

Report4, Response, Device to Host: last response report

4	Register value

ID 64 bytes of data containing the register value. If the number of bytes requested is less than 64, the remaining bytes must be ignored by the host.

9.9.2.1.2 WRITE_REGISTER

The transaction for the WRITE_REGISTER command consists of these reports: Report1 for the command, Report3 for the security configuration and Report4 for the write status.

The host sends Report1 with the WRITE_REGISTER command. The register to write is specified in the ADDRESS field of the SDP command of Report1, with the FORMAT field set to the data type (number of bits to write, either 8, 16, or 32) and the value to write in the DATA field of the SDP command. The device writes the DATA to the register address and returns the WRITE_COMPLETE code using Report4 and the security configuration using Report3 to complete the transaction.

Report1, Command, Host to Device:

1	Valid values for WRITE_REGISTER COMMAND, ADDRESS, FORMAT, DATA_COUNT and DATA

ID 16-byte SDP command

Report3, Response, Device to Host:

3	4 bytes indicating the security configuration

ID 4 bytes status

Report4, Response, Device to Host:

4	WRITE_COMPLETE (0x128A8A12) status

ID 64 bytes data with the first 4 bytes to indicate that the write is completed with code 0x128A8A12. On failure, the device reports the HAB error status.

9.9.2.1.3 WRITE_FILE

The transaction for the WRITE_FILE command consists of these reports: Report1 for the command phase, Report2 for the data phase, Report3 for the HAB mode, and Report4 to indicate that the data are received in full.

The size of each Report2 is limited to 1024 bytes (limitation of the USB HID protocol). Hence, multiple Report2 packets are sent by the host in the data phase until the entire data is transferred to the device. When the entire data (DATA_COUNT bytes) is received, the device sends Report3 with the HAB mode and Report4 with 0x88888888, indicating that the file download completed.

Report1, Host to Device:

1	Valid values for WRITE_FILE COMMAND, ADDRESS, DATA_COUNT

ID 16-byte SDP command

==========Optional Begin==========

Host sends the ERROR_STATUS command to query if the HAB rejected the address

========== Optional End==========

Report2, Host to Device:

2	File data

ID Max 1024 bytes data per report

Report2, Host to Device:

2	File data

ID Max 1024 bytes data per report

Report3, Device to Host:

3	4 bytes indicating security configuration

ID 4 bytes status

Report4, Response, Device to Host:

4	COMPLETE (0x88888888) status

ID 64 bytes data with the first four bytes to indicate that the file download completed with code 0x88888888. On failure, the device reports the HAB error status.

9.9.2.1.4 ERROR_STATUS

The transaction for the SDP command ERROR_STATUS consists of three reports.

Report1 is used by the host to send the command; the device sends global error status in four bytes of Report4 after returning the security configuration in Report3. When the device receives the ERROR_STATUS command, it returns the global error status that is updated for each command. This command is useful to find out whether the last command resulted in a device error or succeeded.

Report1, Command, Host to Device:

1	ERROR_STATUS COMMAND

ID 16-byte SDP Command

Report3, Response, Device to Host:

3	Four bytes indicating the security configuration

ID 4 bytes status

Report4, Response, Device to Host:

4	Four bytes Error status

ID first 4 bytes status in 64 bytes Report4

9.9.2.1.5 DCD_WRITE

The SDP command DCD_WRITE is used by the host to send multiple register writes in one shot. This command is provided to speed up the process of programming the register writes (such as to configure an external RAM device).

The command goes with Report1 from the host with COMMAND TYPE set to DCD_WRITE, ADDRESS which is used as a temporary location of the DCD data, and DATA_COUNT to the number of bytes sent in the data out phase. In the data phase, the

host sends the data for a number of registers using Report2. The device completes the transaction with Report3 indicating the security configuration and Report4 with the WRITE_COMPLETE code 0x12828212.

Report1, Command, Host to Device:

1	DCD_WRITE COMMAND, ADDRESS, DATA_COUNT

ID 16-byte SDP Command

Report2, Data, Host to Device:

2	DCD binary data

ID Max 1024 bytes per report

Report3, Response, Device to Host:

3	Four bytes indicating the security configuration

ID 4 bytes status

Report4, Response, Device to Host:

4	WRITE_COMPLETE (0x128A8A12) status

ID 64 bytes report with the first four bytes to indicate that the write completed with the code 0x128A8A12. On failure, the device reports the HAB error status.

See Device Configuration Data (DCD) for the DCD format description.

9.9.2.1.6 JUMP_ADDRESS

The SDP command JUMP_ADDRESS is the last command that the host can send to the device. After this command, the device jumps to the address specified in the ADDRESS field of the SDP command and starts to execute. It actually expects the IVT address

This command usually follows after the WRITE_FILE command. The command is sent by the host in the command-phase of the transaction using Report1. There is no data phase for this command, but the device sends the status Report3 to complete the transaction. If the authentication fails, it also sends Report4 with the HAB error status.

Report1, Command, Host to Device:

1	JUMP_ADDRESS COMMAND, ADDRESS

ID 16-byte SDP Command

Report3, Response, Device to Host:

3	Four bytes indicating the security configuration

ID 4 bytes status

This report is sent by the device only in case of an error jumping to the given address, or if the device reports error in Report4, Response, Device to Host:

4	Four bytes HAB error status

ID 4 bytes status, 64 bytes report length

9.9.2.1.7 SET_BAUDRATE

The SDP command SET_BAUDRATE is used by the host to configure the UART baudrate on the device side. The default baudrate is 115200.

The transaction for SET_BAUDRATE command consists of 2 stages.

Stage 1, Command, Host to Device:

Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6 to 15
0x0d	0x0d	baudrate [31:24]	baudrate [23:16]	baudrate [15:8]	baudrate [7:0]	All 0x00

Stage 2, Response, Device to Host:

Byte 0	Byte 1	Byte 2	Byte 3
0x09	0xd0	0x0d	0x90

After receiving the Response, the host needs to wait about 100 µs until the device is ready for accepting a new command using the specified baudrate.

9.10 Recovery devices

The chip supports recovery devices. If the primary boot device fails, the boot ROM tries to boot from the recovery device using one of the LPSPI ports.

To enable the recovery device, the recovery boot fuse must be set. Additionally, the serial EEPROM fuses must be set as described in Serial NOR/EEPROM through LPSPI.

9.11 SD/MMC manufacture mode

When the internal boot and recover boot (if enabled) failed, the boot goes to the SD/MMC manufacture mode before the serial download mode. In the manufacture mode, one bit bus width is used despite of the fuse setting.

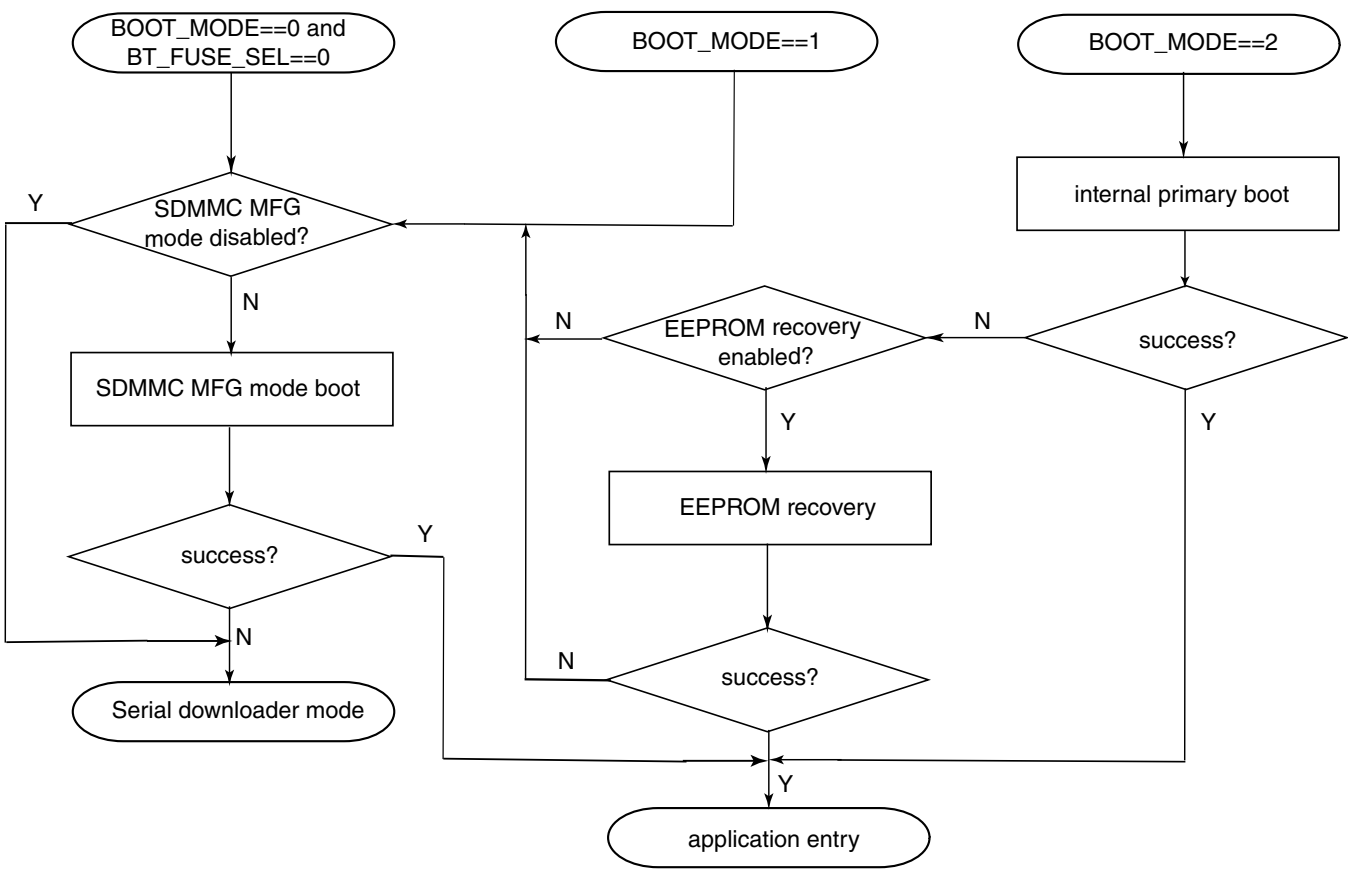


Figure 9-19. SD/MMC manufacture boot flow

9.12 High-Assurance Boot (HAB)

The High Assurance Boot (HAB) component of the ROM protects against the potential threat of attackers modifying the areas of code or data in the programmable memory to make it behave in an incorrect manner. The HAB also prevents the attempts to gain access to features which must not be available.

The integration of the HAB feature with the ROM code ensures that the chip does not enter an operational state if the existing hardware security blocks detected a condition that may be a security threat or if the areas of memory deemed to be important were modified. The HAB uses the RSA digital signatures to enforce these policies.

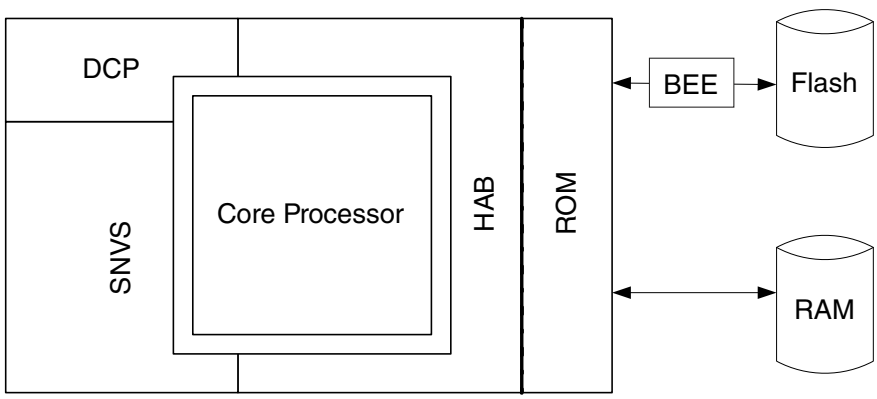


Figure 9-20. Secure boot components

The figure above illustrates the components used during a secure boot using HAB. The HAB interfaces with the SNVS to make sure that the system security state is as expected. The HAB also uses the hardware block to accelerate the SHA-256 message digest operations performed during the signature verifications. The HAB also includes a software implementation of SHA-256 for cases where a hardware accelerator cannot be used. The supported RSA key sizes are 1024, 2048, 3072, and 4096 bits. The main features supported by the HAB are:

- X.509 public key certificate support
- CMS signature format support

NOTE

NXP provides the reference Code Signing Tool (CST) for key generation and code signing for use with the HAB library. The

CST can be found by searching for "IMX_CST_TOOL" at http://www.nxp.com

9.12.1 HAB API vector table addresses

For devices that perform a secure boot, the HAB library may be called by the boot stages that execute after the ROM code.

The HAB API vector table for this device is at address 0x0020_0300 .

9.13 ROM APIs

9.13.1 Introduction

The ROM bootloader provides a set of ROM APIs to simplify the In-Application Programming (IAP) and WDOG operation.

The ROM bootloader supports the following APIs:

- runBootloader API
- FlexSPI NOR Flash Driver API
- RTWDOG Driver API
- WDOG Driver API

The ROM API root pointer is located at address 0x0020_001c . See the following figure for details of the ROM API layout.

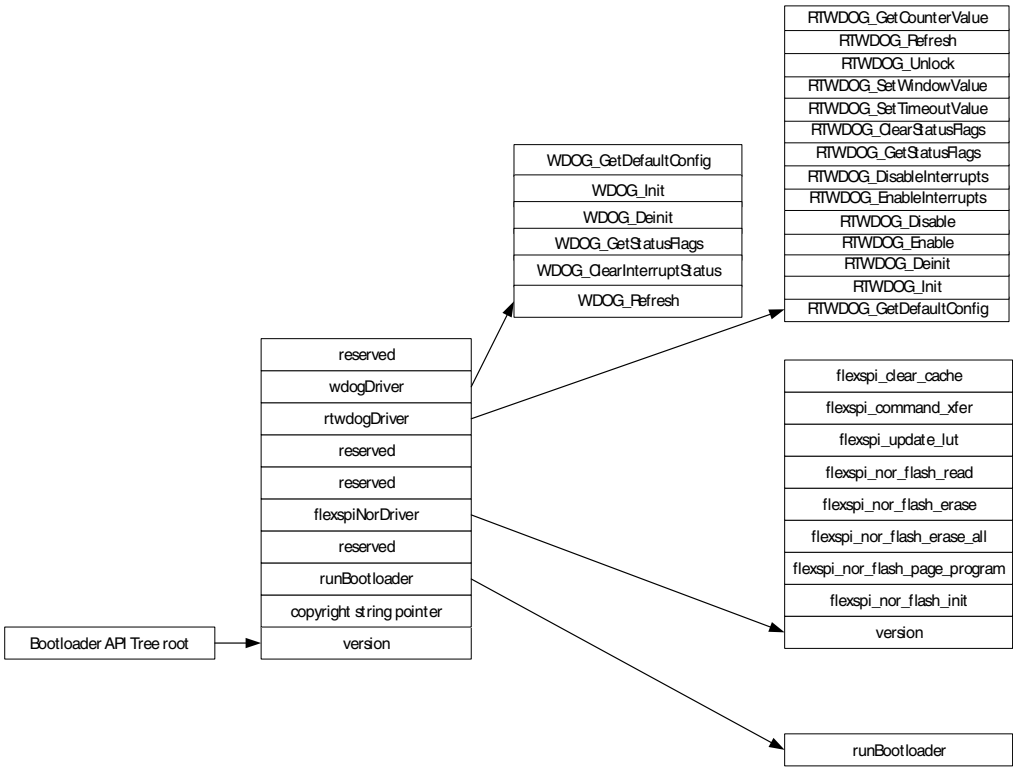


Figure 9-21. ROM API Structure

The ROM API structure definitions are as below:

1. RTWDOG driver API structure

```
typedef struct
{
    void (*RTWDOG_GetDefaultConfig)(rtwdog_config_t *config);
    void (*RTWDOG_Init)(RTWDOG_Type *base, const rtwdog_config_t *config);
    void (*RTWDOG_Deinit)(RTWDOG_Type *base);
    void (*RTWDOG_Enable)(RTWDOG_Type *base);
    void (*RTWDOG_Disable)(RTWDOG_Type *base);
    void (*RTWDOG_EnableInterrupts)(RTWDOG_Type *base, uint32_t mask);
    void (*RTWDOG_DisableInterrupts)(RTWDOG_Type *base, uint32_t mask);
    uint32_t (*RTWDOG_GetStatusFlags)(RTWDOG_Type *base);
    void (*RTWDOG_ClearStatusFlags)(RTWDOG_Type *base, uint32_t mask);
    void (*RTWDOG_SetTimeoutValue)(RTWDOG_Type *base, uint16_t timeoutCount);
    void (*RTWDOG_SetWindowValue)(RTWDOG_Type *base, uint16_t windowValue);
    void (*RTWDOG_Unlock)(RTWDOG_Type *base);
    void (*RTWDOG_Refresh)(RTWDOG_Type *base);
    uint16_t (*RTWDOG_GetCounterValue)(RTWDOG_Type *base);
} rtwdog_driver_interface_t;
```

2. WDOG driver API structure

```
typedef struct
{
    void (*WDOG_GetDefaultConfig)(wdog_config_t *config);
    void (*WDOG_Init)(WDOG_Type *base, const wdog_config_t *config);
    void (*WDOG_Deinit)(WDOG_Type *base);
    void (*WDOG_Enable)(WDOG_Type *base);
    void (*WDOG_Disable)(WDOG_Type *base);
    void (*WDOG_EnableInterrupts)(WDOG_Type *base, uint16_t mask);
    uint16_t (*WDOG_GetStatusFlags)(WDOG_Type *base);
```

```
    void (*WDOG_ClearInterruptStatus)(WDOG_Type *base, uint16_t mask);
    void (*WDOG_SetTimeoutValue)(WDOG_Type *base, uint16_t timeoutCount);
    void (*WDOG_SetInterrputTimeoutValue)(WDOG_Type *base, uint16_t timeoutCount);
    void (*WDOG_DisablePowerDownEnable)(WDOG_Type *base);
    void (*WDOG_Refresh)(WDOG_Type *base);
} wdog_driver_interface_t;
```

3. Bootloader API Entry Structure

```
typedef struct
{
    const uint32_t version;                                  //!< Bootloader version
number
    const char *copyright;                                   //!< Bootloader Copyright
    void (*runBootloader)(void *arg);                        //!< Function to start the
bootloader executing
    const uint32_t *reserved0;                               //!< Reserved
    const flexspi_nor_driver_interface_t *flexSpiNorDriver; //!< FlexSPI NOR Flash API
    const uint32_t *reserved1;                               //!< Reserved
    const rtwdog_driver_interface_t *rtwdogDriver;
    const wdog_driver_interface_t *wdogDriver;
    const uint32_t *reserved2;
} bootloader_api_entry_t;
#define g_bootloaderTree (*(bootloader_api_entry_t**)(0x0020001c))
```

4. FlexSPI NOR Driver API structure

```
typedef struct
{
    uint32_t version;
    status_t (*init)(uint32_t instance, flexspi_nor_config_t *config);
    status_t (*program)(uint32_t instance, flexspi_nor_config_t *config, uint32_t
dst_addr, const uint32_t *src);
    status_t (*erase_all)(uint32_t instance, flexspi_nor_config_t *config);
    status_t (*erase)(uint32_t instance, flexspi_nor_config_t *config, uint32_t start,
uint32_t lengthInBytes);
    status_t (*read)(
        uint32_t instance, flexspi_nor_config_t *config, uint32_t *dst, uint32_t addr,
uint32_t lengthInBytes);
    void (*clear_cache)(uint32_t instance);
    status_t (*xfer)(uint32_t instance, flexspi_xfer_t *xfer);
    status_t (*update_lut)(uint32_t instance, uint32_t seqIndex, const uint32_t
*lutBase, uint32_t seqNumber);
    status_t (*get_config)(uint32_t instance, flexspi_nor_config_t *config,
serial_nor_config_option_t *option);
} flexspi_nor_driver_interface_t;
```

9.13.2 FlexSPI NOR APIs

The ROM bootloader provides a set of Serial NOR FLASH API to simplify the external FLASH enablement on this RT chip. The version of the FlexSPI NOR API in this RT ROM bootloader is version 1.5.0 .

- See the API structure shown in section "FlexSPI NOR Driver API structure"
- See the possible status codes listed in section "Status codes for the FlexSPI NOR API"
- See the flexspi_nor_config_t defined in table "Serial NOR configuration block"

NOTE

1. The ROM API sets the FlexSPI clock root source to PFD480_PFD0 and sets the FLASH read frequency based on the field "serialClkFreq." The "serialClkFreq" and "ipcmdSerialClkFreq" must be set to 0 if the user application selects a different FlexSPI clock root source.
2. The ROM API dynamically changes the FLASH serial root clock for all the non-READ commands according to the "ipcmdSerialClkFreq" field if it is non-zero. This design helps avoid potential FLASH timing issues because some FLASH devices cannot support the program/erase operation at the same frequency as the read operation. This field can be set to 0 if the dynamic clock switch is unnecessary on the chosen FLASH device.
3. The user application must ensure that there is no FLASH access from any masters (for example, Cortex-M7, USB, PXP) during the FLASH erase/program if the FLASH device doesn't support Read-While-Write (RWW) feature. Otherwise, the behavior is unpredictable.
4. The user application should disable the global interrupt during the FLASH erase/program if the code is executing on the NOR FLASH, and the FLASH device doesn't support the RWW feature. Otherwise, the behavior is unpredictable.

FlexSPI NOR prototypes

1. flexspi_nor_flash_init
   Initialize the Serial NOR device via FLEXSPI

```
status_t flexspi_nor_flash_init(uint32_t instance, flexspi_nor_config_t *config)
{
    return g_bootloaderTree->flexSpiNorDriver->init(instance, config);
}
```

   See example code in section "FLASH API example on this RT device EVK board".

2. flexspi_nor_flash_page_program
   Program data to specified Flash address

```
status_t flexspi_nor_flash_page_program(uint32_t instance, flexspi_nor_config_t
*config, uint32_t dstAddr, const uint32_t *src)
{
    return g_bootloaderTree->flexSpiNorDriver->program(instance, config, dstAddr, src);
}
```

   See example code in section "FLASH API example on this RT device EVK board".

3. flexspi_nor_flash_erase_all
   Erase the whole Flash array via FLEXSPI

```
status_t flexspi_nor_flash_erase_all(uint32_t instance, flexspi_nor_config_t *config)
{
    return g_bootloaderTree->flexSpiNorDriver->erase_all(instance, config);
}
```

   Example code:

```
flexspi_nor_flash_erase_all(0, config);
```

4. flexspi_nor_get_config
   Get the Flash configuration block via the serial_nor_config_option_t block, see section "serial_nor_config_t definition ".

```
status_t flexspi_nor_get_config(uint32_t instance, flexspi_nor_config_t *config,
serial_nor_config_option_t *option)
{
    return g_bootloaderTree->flexSpiNorDriver->get_config(instance, config, option);
}
```

   See example code in section "FLASH API example on this RT device EVK board".

5. flexspi_nor_flash_erase
   Erase specified Flash region, the minimum erase unit is one sector.

```
status_t flexspi_nor_flash_erase(uint32_t instance, flexspi_nor_config_t *config,
uint32_t start, uint32_t length)
{
    return g_bootloaderTree->flexSpiNorDriver->erase(instance, config, start, length);
}
```

   See example code in section "FLASH API example on this RT device EVK board".

6. flexspi_nor_flash_read
   Read the FLASH via FLEXSPI using IP read command

```
status_t flexspi_nor_flash_read(uint32_t instance, flexspi_nor_config_t *config,
uint32_t *dst, uint32_t start, uint32_t bytes)
{
    return g_bootloaderTree->flexSpiNorDriver->read(instance, config, dst, start,
bytes);
}
```

   Example code:

```
uint32_t pageBuffer[256/sizeof(uint32_t)];
flexspi_nor_flash_read(0, config, pageBuffer, 0, sizeof(pageBuffer));
```

7. flexspi_update_lut
   Update the specified LUT entries

```
status_t flexspi_update_lut(uint32_t instance, uint32_t seqIndex, const uint32_t
*lutBase, uint32_t numberOfSeq)
{
```

```
    return g_bootloaderTree->flexSpiNorDriver->update_lut(instance, seqIndex, lutBase,
numberOfSeq);
}
```

Example code:

```
uint32_t chipEraseLUT[4] = {0x0460, 0, 0, 0};
flexspi_update_lut(0, 1, chipEraseLUT, 1);
```

8. flexspi_command_xfer
   Execute LUT sequence specified by xfer

```
status_t flexspi_command_xfer(uint32_t instance, flexspi_xfer_t *xfer)
{
    return g_bootloaderTree->flexSpiNorDriver->xfer(instance, xfer);
}
```

flexspi_xfer_t definition is as below:

```
//!@brief FlexSPI Transfer Context
typedef struct _FlexSpiXfer
{
    flexspi_operation_t operation; //!< FlexSPI operation
    uint32_t baseAddress;          //!< FlexSPI operation base address
    uint32_t seqId;                //!< Sequence Id
    uint32_t seqNum;               //!< Sequence Number
    bool isParallelModeEnable;     //!< Is a parallel transfer
    uint32_t *txBuffer;            //!< Tx buffer
    uint32_t txSize;               //!< Tx size in bytes
    uint32_t *rxBuffer;            //!< Rx buffer
    uint32_t rxSize;               //!< Rx size in bytes
} flexspi_xfer_t;
```

flexspi_operation_t definition is as below:

```
typedef enum _FlexSPIOperationType
{
    kFlexSpiOperation_Command, //!< FlexSPI operation: Only command, both TX and
    //! RX buffer are ignored.
    kFlexSpiOperation_Config, //!< FlexSPI operation: Configure device mode, the
    //! TX FIFO size is fixed in LUT.
    kFlexSpiOperation_Write, //!< FlexSPI operation: Write,  only TX buffer is
    //! effective
    kFlexSpiOperation_Read, //!< FlexSPI operation: Read, only Rx Buffer is
    //! effective.
    kFlexSpiOperation_End = kFlexSpiOperation_Read,
} flexspi_operation_t;
```

Example code, assuming the LUT index 1 is the Flash WriteEnable command.

```
flexspi_xfer_t flashXfer =
{
     kFlexSpiOperation_Command, 0, 1, 1, false, NULL, 0, NULL, 0
};
flexspi_command_xfer(0, &flashXfer);
```

9. flexspi_clear_cache
   Clear the AHB buffer in FLEXSPI module.

```
void flexspi_clear_cache(uint32_t instance)
{
    g_bootloaderTree->flexSpiNorDriver->clear_cache(instance);
}
```

Example code:

```
flexspi_clear_cache(0);
```

10. serial_nor_config_t definition
    serial_nor_config_t is defined as below.

```
typedef struct _serial_nor_config_option
{
    union
    {
        struct
        {
            uint32_t max_freq : 4;          //!< Maximum supported Frequency
            uint32_t misc_mode : 4;         //!< miscellaneous mode
            uint32_t quad_mode_setting : 4; //!< Quad mode setting
            uint32_t cmd_pads : 4;          //!< Command pads
            uint32_t query_pads : 4;        //!< SFDP read pads
            uint32_t device_type : 4;       //!< Device type
            uint32_t option_size : 4;       //!< Option size, in terms of uint32_t,
size = (option_size + 1) * 4
            uint32_t tag : 4;               //!< Tag, must be 0x0E
        } B;
        uint32_t U;
    } option0;

    union
    {
        struct
        {
            uint32_t dummy_cycles : 8;     //!< Dummy cycles before read
            uint32_t reserved0 : 8;  //!< Reserved for future use
            uint32_t pinmux_group : 4;     //!< The pinmux group selection
            uint32_t reserved1 : 8; //!< Reserved for future use
            uint32_t flash_connection : 4; //!< Flash connection option: 0 - Single
Flash connected to port A
        } B;
        uint32_t U;
    } option1;

} serial_nor_config_option_t;
```

Detailed infomation of serial_nor_config_option_t structure is shown in the following table.

Table 9-50. serial_nor_config_option_t definition

Offset	Field	Description
0	Option0	See option0 definition for more details
4	Option1	Optional, effective only if the Option Size field in Option0 is non-zero See option1 definition for more details.

Table 9-51. Option0 definition

Field	Bits	Description
tag	31:28	The tag of the config option, fixed to 0x0C
option_size	27:24	Size in bytes = (Option Size + 1) × 4

Table continues on the next page...

Table 9-51. Option0 definition (continued)

		It is 0 if only option0 is required
device_type	23:20	Device Detection Type 0 - Read SFDP for SDR commands 1 - Read SFDP for DDR Read commands 2 - HyperFLASH 1V8 3 - HyperFLASH 3V 4 - Macronix Octal DDR 6 - Micron Octal DDR 8 - Adesto EcoXiP DDR
query_pad	19:16	Data pads during Query command (read SFDP or read MID) 0 - 1 2 - 4 3 – 8
cmd_pad	15:12	Data pads during Flash access command 0 - 1 2 - 4 3 – 8
quad_mode_setting	11:8	Quad Mode Enable Setting 0 - Not configure 1 - Set bit 6 in Status Register 1 2 - Set bit 1 in Status Register 2 3 - Set bit 7 in Status Register 2 4 - Set bit 1 in Status Register 2 vis 0x31 command **NOTE:** This field will be effective only if device is compliant with JESD216 only (9 longword SDFP table).
misc_mode	7:4	Miscellaneous Mode 0 - Not enabled 1 - Enable 0-4-4 mode for High Random Read performance 3 - Data Order Swapped mode (for MXIC OctaFlash only) **NOTE:** Experimental feature, do not use in products, keep it as 0.
max_freq	3:0	Max Flash Operation speed 0 - Don't change FlexSPI clock setting Others - See System Boot chapter for more details.

Table 9-51. Option0 definition

		NOTE:	The field has a restriction that the FlexSPI clock source must be PLL480_PFD0, keep it as 0 and manually configure the FLEXSPI clock if another clock source is selected in user application.

Table 9-52. Option1 definition

Field	Bits	Description
flash_connection	31:28	Flash connection selection 0 - Only Port A
reserved	27:20	Reserved
pin_group	19:16	PinMux group 0 - Primary pin group 1 - Secondary pin group
reserved	15:8	Reserved for future use
dummy_cycles	7:0	Dummy cycles for read command 0 - Use detected dummy cycle Others - dummy cycles provided in flash data sheet

NOTE

a. These APIs only support 1 single FLASH device connected to PORTA and FLEXSPIA_SS0 .
b. Parallel mode is **NOT** supported.
c. The APIs always use 30 MHz clock for the programming option. Users need to change the "ipcmdSerialClkFreq" field in flexspi_nor_config_t field after flexspi_nor_get_config option if a higher programming speed is expected.
d. User application needs to set "max_freq" to 0 and manually configure the FLEXSPI clock prior to calling the flexspi_nor_get_config API, if the expected FLEXSPI clock source is not the default clock source configured by the ROM bootloader.

e. The pad drive strength is configured to "R0/6" (See IOMUXC chapter for more details). Users can change the "dataPadOverride" and "sclkPadOverride" setting in flexspi_nor_config_t structure after calling flexspi_nor_get_config if it is necessary.

11. Status codes for the FlexSPI NOR API
The following table lists all the error and status codes for the FlexSPI NOR API.

Table 9-53. Status and error codes for the FlexSPI NOR API

Status	Code	Description
kStatus_Success	0	Operation succeeded without error
kStatus_Fail	1	Operation failed with a generic error
kStatus_InvalidArgument	4	The requested argument is invalid
kStatus_Timeout	5	A timeout occurred
kStatus_FLEXSPI_SequenceExecutionTimeout	7000	The command timed out
kStatus_FLEXSPI_InvalidSequence	7001	Invalid LUT sequence
kStatus_FLEXSPI_DeviceTimeout	7002	The busy time exceeded provided timeout value
kStatus_FLEXSPINOR_ProgramFail	20100	Program Command failed
kStatus_FLEXSPINOR_EraseSectorFail	20101	Erase sector command failed
kStatus_FLEXSPINOR_EraseAllFail	20102	Erase All command failed
kStatus_FLEXSPINOR_WaitTimeout	20103	The wait time exceeded provided timeout value
kStatus_FlexSPINOR_NotSupported	20104	The operation is not supported
kStatus_FlexSPINOR_WriteAlignmentError	20105	Write address is unaligned to page size
kStatus_FlexSPINOR_CommandFailure	20106	Command failed
kStatus_FlexSPINOR_SFDP_NotFound	20107	SFDP table was not found, used for flexspi_nor_flash_get_config API
kStatus_FLEXSPINOR_Flash_NotFound	20109	Cannot detect a FLASH device
kStatus_FLEXSPINOR_DTRRead_DummyProbeFailed	20110	The dummy cycle for DDR/DTR read cannot be probed.

The following list provides typical market options for the serial NOR flash devices.

- QuadSPI NOR - Quad SDR Read: option0 = 0xc0000008 (133MHz)
- QuadSPI NOR - Quad DDR Read: option0 = 0xc0100003 (60MHz)
- HyperFLASH 1V8: option0 = 0xc0233009 (166MHz)
- HyperFLASH 3V0: option0 = 0xc0333006 (100MHz)
- MXIC OPI DDR (OPI DDR enabled by default): option=0xc0433008(133MHz)
- Micron Octal DDR: option0=0xc0600006 (100MHz)

- Micron OPI DDR: option0=0xc0603008 (133MHz), SPI->OPI DDR
- Micron OPI DDR (DDR read enabled by default): option0 = 0xc0633008 (133MHz)
- Adesto OPI DDR: option0=0xc0803008(133MHz)

The following is a typical use case of the FlexSPI NOR API.

```
flexspi_nor_config_t config;
serial_nor_config_option_t option;
status_t status;
uint32_t address = 0x40000; // 256KB
uint32_t sector_size = 0x1000; // 4KB
uint32_t page_buffer[256/ sizeof(uint32_t)];
uint32_t instance = 0;

option.option0.U = 0xC0000008; // QuadSPI NOR, Frequency: 133MHz

status = flexspi_nor_get_config(instance, &config, &option);
if (status != kStatus_Success)
{
    return status;
}

status = flexspi_nor_flash_init(instance, &config);
if (status != kStatus_Success)
{
    return status;
}

status = flexspi_nor_flash_erase(instance, &config, address , sector_size); // Erase 1
sector
if (status != kStatus_Success)
{
    return status;
}

// Fill data into the page_buffer;
for (uint32_t i=0; i<sizeof(page_buffer)/sizeof(page_buffer[0]); i++)
{
    page_buffer[i] = (i << 24) | (i << 16) | (i << 8) | i;
}
// Program data to destination
status = flexspi_nor_flash_page_program(instance, &config, address, page_buffer); // program
1 page
if (status != kStatus_Success)
{
    return status;
}

// Do cache maintenance here if the D-Cache is enabled
// Use memory mapped access to verify whether data are programmed into Flash correctly
uint32_t mem_address = 0x6000_0000 + address;
if (0 == memcmp((void*)mem_address, page_buffer, sizeof(page_buffer))
{
// Success
}
else
{     // Report error
}
```

9.13.3 Enter Bootloader API

The ROM bootloader provides an API for the user application to boot from a new updated application safely after In-Application Programming (IAP) or re-enter serial downloader mode for image update. See more details in the next section.

API prototype

```
void runBootloader(void* arg)
{
   g_bootloaderTree-> runBootloader(arg);
}
```

ARG definition

Field	Offset	Description
Tag	[31:24]	Fixed value: 0xEB (Enter Boot)
boot mode	[23:20]	0 - Determined by BMODE in SMBR2 or other Fuse combinations 1 - Serial downloader
Serial downloader media	[19:16]	0 - Auto detection 1 - USB 2 - UART
Reserved	[15:04]	
Boot image selection	[03: 00]	0 - Image 0 (Primary image) 1 - Image 1 (Redundant image, with primary image bypassed) 2 - Image 2 3 - Image 3 **NOTE:** It takes effect only if the boot mode is 0.

Typical use cases

1. Enter Serial downloader mode and select USB as the communication peripheral.

   ```
   uint32_t arg = 0xeb100000;
   runBootloader (&arg);
   ```

2. Select Image1 (Redundant image) as the boot image after reliable update in user application.

   ```
   uint32_t arg = 0xeb000001;
   runBootloader (&arg);
   ```

9.13.4 Watchdog API

The Watchdog API definition is identical to the SDK driver, see the SDK driver for more details.

Chapter 10
External Signals and Pin Multiplexing

10.1 Overview

The chip contains a limited number of pins, most of which have multiple signal options. These signal-to-pin and pin-to-signal options are selected by the input-output multiplexer called IOMUX. The IOMUX is also used to configure other pin characteristics, such as voltage level, drive strength, and hysteresis.

The muxing options table lists the external signals grouped by the module instance, the muxing options for each signal, and the registers used to route the signal to the chosen pad.

10.1.1 Muxing Options

NOTE

System STOP/WAIT status can be observed on pin. See CCM_WAIT and CCM_STOP in the table below.

Table 10-1. Muxing Options

Instance	Port	Pad	Mode
ADC1	ADC1_IN0	GPIO_AD_B1_11 pin 21	-
	ADC1_IN1	GPIO_AD_B0_12 pin 24	-
	ADC1_IN2	GPIO_AD_B0_13 pin 25	-
	ADC1_IN3	GPIO_AD_B0_14	-
	ADC1_IN4	GPIO_AD_B0_15	-
	ADC1_IN5	GPIO_AD_B1_00 pin 19	-
	ADC1_IN6	GPIO_AD_B1_01 pin 18	-
	ADC1_IN7	GPIO_AD_B1_02 pin 14	-
	ADC1_IN8	GPIO_AD_B1_03 pin 15	-
	ADC1_IN9	GPIO_AD_B1_04 pin 40 (T4.1)	-

Table continues on the next page...

Table 10-1. Muxing Options (continued)

	ADC1_IN10	GPIO_AD_B1_05	pin 41 (T4.1)
	ADC1_IN11	GPIO_AD_B1_06	pin 17
	ADC1_IN12	GPIO_AD_B1_07	pin 16
	ADC1_IN13	GPIO_AD_B1_08	pin 22
	ADC1_IN14	GPIO_AD_B1_09	pin 23
	ADC1_IN15	GPIO_AD_B1_10	pin 20
ADC2	ADC2_IN0	GPIO_AD_B1_11	pin 21
	ADC2_IN1	GPIO_AD_B1_12	pin 38 (T4.1)
	ADC2_IN2	GPIO_AD_B1_13	pin 39 (T4.1)
	ADC2_IN3	GPIO_AD_B1_14	pin 26
	ADC2_IN4	GPIO_AD_B1_15	pin 27
	ADC2_IN5	GPIO_AD_B1_00	pin 19
	ADC2_IN6	GPIO_AD_B1_01	pin 18
	ADC2_IN7	GPIO_AD_B1_02	pin 14
	ADC2_IN8	GPIO_AD_B1_03	pin 15
	ADC2_IN9	GPIO_AD_B1_04	pin 40 (T4.1)
	ADC2_IN10	GPIO_AD_B1_05	pin 41 (T4.1)
	ADC2_IN11	GPIO_AD_B1_06	pin 17
	ADC2_IN12	GPIO_AD_B1_07	pin 16
	ADC2_IN13	GPIO_AD_B1_08	pin 22
	ADC2_IN14	GPIO_AD_B1_09	pin 23
	ADC2_IN15	GPIO_AD_B1_10	pin 20
ACMP1	ACMP1_IN0	GPIO_AD_B1_01	pin 18
	ACMP1_IN1	GPIO_AD_B1_06	pin 17
	ACMP1_IN2	GPIO_AD_B0_13	pin 25
	ACMP1_IN3	GPIO_AD_B1_02	pin 14
	ACMP1_IN4	GPIO_AD_B0_00	
	ACMP1_IN5	GPIO_AD_B1_07	pin 16
	ACMP1_IN6	GPIO_AD_B1_11	pin 21
	ACMP1_OUT	GPIO_AD_B1_12	pin 38 (T4.1)
ACMP2	ACMP2_IN0	GPIO_AD_B1_01	pin 18
	ACMP2_IN1	GPIO_AD_B1_06	pin 17
	ACMP2_IN2	GPIO_AD_B0_14	
	ACMP2_IN3	GPIO_AD_B1_03	pin 15
	ACMP2_IN4	GPIO_AD_B0_01	
	ACMP2_IN5	GPIO_AD_B1_08	pin 22
	ACMP2_IN6	GPIO_AD_B1_12	pin 38 (T4.1)
	ACMP2_OUT	GPIO_AD_B1_13	pin 39 (T4.1)
ACMP3	ACMP3_IN0	GPIO_AD_B1_01	pin 18
	ACMP3_IN1	GPIO_AD_B1_06	pin 17

Table continues on the next page...

Table 10-1. Muxing Options (continued)

	ACMP3_IN2	GPIO_AD_B0_15	-
	ACMP3_IN3	GPIO_AD_B1_04	pin 40 (T4.1)
	ACMP3_IN4	GPIO_AD_B0_02	pin 1
	ACMP3_IN5	GPIO_AD_B1_09	pin 23
	ACMP3_IN6	GPIO_AD_B1_13	pin 39 (T4.1)
	ACMP3_OUT	GPIO_AD_B1_14	pin 26
ACMP4	ACMP4_IN0	GPIO_AD_B1_01	pin 18
	ACMP4_IN1	GPIO_AD_B1_06	pin 17
	ACMP4_IN2	GPIO_AD_B1_00	pin 19
	ACMP4_IN3	GPIO_AD_B1_05	pin 41 (T4.1)
	ACMP4_IN4	GPIO_AD_B0_03	pin 0
	ACMP4_IN5	GPIO_AD_B1_10	pin 20
	ACMP4_IN6	GPIO_AD_B1_14	pin 26
	ACMP4_OUT	GPIO_AD_B1_15	pin 27
ARM PLATFORM	ARM_EVENT0	GPIO_BO_14	ALT2
	ARM_EVENT1	GPIO_B0_15	ALT2
CCM	CCM_PMIC_STBY_REQ	PMIC_STBY_REQ	ALT0
	CCM_PMIC_READY	GPIO_AD_B0_12	pin 24 ALT1
		GPIO_EMC_32	pin 28 ALT3
		GPIO_AD_B1_08	pin 22 ALT3
		GPIO_AD_B1_01	pin 18 ALT4
		GPIO_SD_B1_03	ALT6
	CCM_CLKO1	GPIO_SD_B0_04	ALT6
	CCM_CLKO2	GPIO_SD_B0_05	ALT6
	CCM_WAIT	GPIO_SD_B1_02	ALT6
	CCM_STOP	GPIO_SD_B1_04	ALT6
	CCM_CLK1_P	CCM_CLK1_P	No Muxing
	CCM_CLK1_N	CCM_CLK1_N	No Muxing
CSI	CSI_DATA00	GPIO_B1_10	ALT2
	CSI_DATA01	GPIO_B1_11	ALT2
	CSI_DATA02	GPIO_AD_B1_15	pin 27 ALT4
		GPIO_AD_B0_11	ALT4
	CSI_DATA03	GPIO_AD_B1_14	pin 26 ALT4
		GPIO_AD_B0_10	ALT4
	CSI_DATA04	GPIO_AD_B1_13	pin 39 (T4.1) ALT4
		GPIO_AD_B0_09	ALT4
	CSI_DATA05	GPIO_AD_B1_12	pin 38 (T4.1) ALT4
		GPIO_AD_B0_08	ALT4
	CSI_DATA06	GPIO_AD_B1_11	pin 21 ALT4
		GPIO_AD_B0_07	ALT4

Table continues on the next page...

Table 10-1. Muxing Options (continued)

	CSI_DATA07	GPIO_AD_B1_10 pin 20	ALT4
		GPIO_AD_B0_06	ALT4
	CSI_DATA08	GPIO_AD_B1_09 pin 23	ALT4
		GPIO_AD_B0_05	ALT4
	CSI_DATA09	GPIO_AD_B1_08 pin 22	ALT4
		GPIO_AD_B0_04	ALT4
	CSI_DATA10	GPIO_B1_09	ALT2
	CSI_DATA11	GPIO_B1_08	ALT2
	CSI_DATA12	GPIO_B1_07	ALT2
	CSI_DATA13	GPIO_B1_06	ALT2
	CSI_DATA14	GPIO_B1_05	ALT2
	CSI_DATA15	GPIO_B1_04	ALT2
	CSI_DATA16	GPIO_EMC_37 pin 30	ALT4
	CSI_DATA17	GPIO_EMC_36 pin 31	ALT4
	CSI_DATA18	GPIO_EMC_35	ALT4
	CSI_DATA19	GPIO_EMC_34	ALT4
	CSI_DATA20	GPIO_EMC_33	ALT4
	CSI_DATA21	GPIO_EMC_32 pin 28	ALT4
	CSI_DATA22	GPIO_EMC_31 pin 29	ALT4
	CSI_DATA23	GPIO_EMC_30	ALT4
	CSI_PIXCLK	GPIO_B1_12 pin 35 (T4.1)	ALT2
		GPIO_AD_B1_04 pin 40 (T4.1)	ALT4
	CSI_VSYNC	GPIO_B1_13 pin 34 (T4.1)	ALT2
		GPIO_AD_B1_06 pin 17	ALT4
		GPIO_AD_B0_14	ALT4
	CSI_HSYNC	GPIO_B1_14	ALT2
		GPIO_AD_B1_07 pin 16	ALT4
		GPIO_AD_B0_15	ALT4
	CSI_MCLK	GPIO_B1_15	ALT2
		GPIO_AD_B1_05 pin 41 (T4.1)	ALT4
	CSI_FIELD	GPIO_EMC_38	ALT4
ENET	ENET_MDC	GPIO_B1_14	ALT0
		GPIO_AD_B1_04 pin 40 (T4.1)	ALT1
		GPIO_EMC_40	ALT4
	ENET_MDIO	GPIO_B1_15	ALT0
		GPIO_AD_B1_05 pin 41 (T4.1)	ALT1
		GPIO_EMC_41	ALT4
	ENET_TX_DATA0	GPIO_EMC_22	ALT3
		GPIO_B1_07	ALT3
	ENET_TX_DATA1	GPIO_EMC_21	ALT3

Table continues on the next page...

Table 10-1. Muxing Options (continued)

		GPIO_B1_08	ALT3
	ENET_TX_DATA2	GPIO_AD_B0_05	ALT2
	ENET_TX_DATA3	GPIO_AD_B0_04	ALT2
	ENET_TX_ER	GPIO_AD_B0_07	ALT2
	ENET_TX_EN	GPIO_EMC_24	ALT3
		GPIO_B1_09	ALT3
	ENET_TX_CLK	GPIO_B1_10	ALT3
		GPIO_EMC_25	ALT3
	ENET_RX_DATA0	GPIO_B1_04	ALT3
		GPIO_EMC_20	ALT3
	ENET_RX_DATA1	GPIO_B1_05	ALT3
		GPIO_EMC_19	ALT3
	ENET_RX_DATA2	GPIO_AD_B0_09	ALT2
	ENET_RX_DATA3	GPIO_AD_B0_08	ALT2
	ENET_RX_ER	GPIO_B1_11	ALT3
		GPIO_EMC_26	ALT3
	ENET_RX_EN	GPIO_B1_06	ALT3
		GPIO_EMC_23	ALT3
	ENET_RX_CLK	GPIO_AD_B0_06	ALT2
	ENET_CRS	GPIO_AD_B0_10	ALT2
	ENET_COL	GPIO_AD_B0_11	ALT2
	ENET_REF_CLK	GPIO_EMC_25	ALT4
		GPIO_B1_10	ALT6
	ENET_1588_EVENT0_OUT	GPIO_AD_B0_14	ALT3
		GPIO_AD_B0_10	ALT7
		GPIO_B1_13 pin 34 (T4.1)	ALT3
	ENET_1588_EVENT0_IN	GPIO_AD_B0_15	ALT3
		GPIO_AD_B0_11	ALT7
		GPIO_B1_12 pin 35 (T4.1)	ALT3
	ENET_1588_EVENT1_OUT	GPIO_AD_B0_12 pin 24	ALT6
	ENET_1588_EVENT1_IN	GPIO_AD_B0_13 pin 25	ALT6
	ENET_1588_EVENT2_OUT	GPIO_AD_B1_02 pin 14	ALT4
	ENET_1588_EVENT2_IN	GPIO_AD_B1_03 pin 15	ALT4
	ENET_1588_EVENT3_OUT	GPIO_AD_B0_07	ALT7
	ENET_1588_EVENT3_IN	GPIO_AD_B0_08	ALT7
ENET2	ENET2_MDC	GPIO_EMC_38	ALT8
		GPIO_B0_00 pin 10	ALT8
	ENET2_MDIO	GPIO_EMC_39	ALT8
		GPIO_B0_01 pin 12	ALT8
	ENET2_TDATA0	GPIO_EMC_30	ALT8

Table continues on the next page...

Table 10-1. Muxing Options (continued)

		GPIO_B0_12 pin 32	ALT8
		GPIO_B1_14	ALT8
	ENET2_TDATA1	GPIO_EMC_31 pin 29	ALT8
		GPIO_B0_13	ALT8
		GPIO_B1_15	ALT8
	ENET2_TDATA2	GPIO_B0_05 pin 41 (MM)	ALT8
	ENET2_TDATA3	GPIO_B0_04 pin 40 (MM)	ALT8
	ENET2_TX_CLK	GPIO_EMC_33	ALT8
		GPIO_B0_15	ALT8
		GPIO_SD_B0_01	ALT8
	ENET2_TX_EN	GPIO_EMC_32 pin 28	ALT8
		GPIO_B0_14	ALT8
		GPIO_SD_B0_00	ALT8
	ENET2_TX_ER	GPIO_B0_07 pin 43 (MM)	ALT8
	ENET2_RDATA0	GPIO_EMC_35	ALT8
		GPIO_B1_01 pin 7	ALT8
		GPIO_SD_B0_03	ALT8
	ENET2_RDATA1	GPIO_EMC_36 pin 31	ALT8
		GPIO_B1_02 pin 36 (T4.1)	ALT8
		GPIO_SD_B0_04	ALT8
	ENET2_RDATA2	GPIO_B0_09 pin 45 (MM)	ALT8
	ENET2_RDATA3	GPIO_B0_08 pin 44 (MM)	ALT8
	ENET2_RX_CLK	GPIO_B0_06 pin 42 (MM)	ALT8
	ENET2_RX_EN	GPIO_EMC_37 pin 30	ALT8
		GPIO_B1_03 pin 37 (T4.1)	ALT8
		GPIO_SD_B0_05	ALT8
	ENET2_RX_ER	GPIO_EMC_34	ALT8
		GPIO_B1_00 pin 8	ALT8
		GPIO_SD_B0_02	ALT8
	ENET2_REF_CLK2	GPIO_EMC_33	ALT9
		GPIO_B0_15	ALT9
		GPIO_SD_B0_01	ALT9
	ENET2_1588_EVENT0_IN	GPIO_AD_B1_01 pin 18	ALT8
		GPIO_B0_03 pin 13	ALT8
	ENET2_COL	GPIO_B0_11 pin 9	ALT8
	ENET2_CRS	GPIO_B0_10 pin 6	ALT8
	ENET2_1588_EVENT1_IN	GPIO_AD_B1_11 pin 21	ALT8
	ENET2_1588_EVENT2_IN	GPIO_AD_B1_13 pin 39 (T4.1)	ALT8
	ENET2_1588_EVENT3_IN	GPIO_AD_B1_15 pin 27	ALT8
	ENET2_1588_EVENT0_OUT	GPIO_AD_B1_00 pin 19	ALT8

Table continues on the next page...

Table 10-1. Muxing Options (continued)

		GPIO_B0_02 pin 11	ALT8
	ENET2_1588_EVENT1_OUT	GPIO_AD_B1_10 pin 20	ALT8
	ENET2_1588_EVENT2_OUT	GPIO_AD_B1_12 pin 38 (T4 LT)	ALT8
	ENET2_1588_EVENT3_OUT	GPIO_AD_B1_14 pin 26	ALT8
EWM	EWM_OUT_B	GPIO_AD_B0_01	ALT6
		GPIO_AD_B0_13 pin 25	ALT3
		GPIO_AD_B1_11 pin 21	ALT1
FLEXIO1	FLEXIO1_D00	GPIO_EMC_00	ALT4
	FLEXIO1_D01	GPIO_EMC_01	ALT4
	FLEXIO1_D02	GPIO_EMC_02	ALT4
	FLEXIO1_D03	GPIO_EMC_03	ALT4
	FLEXIO1_D04	GPIO_EMC_04 pin 2	ALT4
	FLEXIO1_D05	GPIO_EMC_05 pin 3	ALT4
	FLEXIO1_D06	GPIO_EMC_06 pin 4	ALT4
	FLEXIO1_D07	GPIO_EMC_07 pin 33	ALT4
	FLEXIO1_D08	GPIO_EMC_08 pin 5	ALT4
	FLEXIO1_D09	GPIO_EMC_09	ALT4
	FLEXIO1_D10	GPIO_EMC_10	ALT4
	FLEXIO1_D11	GPIO_EMC_11	ALT4
	FLEXIO1_D12	GPIO_EMC_26	ALT4
	FLEXIO1_D13	GPIO_EMC_27	ALT4
	FLEXIO1_D14	GPIO_EMC_28	ALT4
	FLEXIO1_D15	GPIO_EMC_29	ALT4
FLEXIO2	FLEXIO2_D00	GPIO_B0_00 pin 10	ALT4
	FLEXIO2_D01	GPIO_B0_01 pin 12	ALT4
	FLEXIO2_D02	GPIO_B0_02 pin 11	ALT4
	FLEXIO2_D03	GPIO_B0_03 pin 13	ALT4
	FLEXIO2_D04	GPIO_B0_04 pin 40 (MM)	ALT4
	FLEXIO2_D05	GPIO_B0_05 pin 41 (MM)	ALT4
	FLEXIO2_D06	GPIO_B0_06 pin 42 (MM)	ALT4
	FLEXIO2_D07	GPIO_B0_07 pin 43 (MM)	ALT4
	FLEXIO2_D08	GPIO_B0_08 pin 44 (MM)	ALT4
	FLEXIO2_D09	GPIO_B0_09 pin 45 (MM)	ALT4
	FLEXIO2_D10	GPIO_B0_10 pin 6	ALT4
	FLEXIO2_D11	GPIO_B0_11 pin 9	ALT4
	FLEXIO2_D12	GPIO_B0_12 pin 32	ALT4
	FLEXIO2_D13	GPIO_B0_13	ALT4
	FLEXIO2_D14	GPIO_B0_14	ALT4
	FLEXIO2_D15	GPIO_B0_15	ALT4
	FLEXIO2_D16	GPIO_B1_00 pin 8	ALT4

Table continues on the next page...

Table 10-1. Muxing Options (continued)

	FLEXIO2_D17	GPIO_B1_01 pin 7	ALT4
	FLEXIO2_D18	GPIO_B1_02 pin 36 (T4.1)	ALT4
	FLEXIO2_D19	GPIO_B1_03 pin 37 (T4.1)	ALT4
	FLEXIO2_D20	GPIO_B1_04	ALT4
	FLEXIO2_D21	GPIO_B1_05	ALT4
	FLEXIO2_D22	GPIO_B1_06	ALT4
	FLEXIO2_D23	GPIO_B1_07	ALT4
	FLEXIO2_D24	GPIO_B1_08	ALT4
	FLEXIO2_D25	GPIO_B1_09	ALT4
	FLEXIO2_D26	GPIO_B1_10	ALT4
	FLEXIO2_D27	GPIO_B1_11	ALT4
	FLEXIO2_D28	GPIO_B1_12 pin 35 (T4.1)	ALT4
	FLEXIO2_D29	GPIO_B1_13 pin 34 (T4.1)	ALT4
	FLEXIO2_D30	GPIO_B1_14	ALT4
	FLEXIO2_D31	GPIO_B1_15	ALT4
FLEXIO3	FLEXIO3_D00	GPIO_AD_B1_00 pin 19	ALT9
	FLEXIO3_D01	GPIO_AD_B1_01 pin 18	ALT9
	FLEXIO3_D02	GPIO_AD_B1_02 pin 14	ALT9
	FLEXIO3_D03	GPIO_AD_B1_03 pin 15	ALT9
	FLEXIO3_D04	GPIO_AD_B1_04 pin 40 (T4.1)	ALT9
	FLEXIO3_D05	GPIO_AD_B1_05 pin 41 (T4.1)	ALT9
	FLEXIO3_D06	GPIO_AD_B1_06 pin 17	ALT9
	FLEXIO3_D07	GPIO_AD_B1_07 pin 16	ALT9
	FLEXIO3_D08	GPIO_AD_B1_08 pin 22	ALT9
	FLEXIO3_D09	GPIO_AD_B1_09 pin 23	ALT9
	FLEXIO3_D10	GPIO_AD_B1_10 pin 20	ALT9
	FLEXIO3_D11	GPIO_AD_B1_11 pin 21	ALT9
	FLEXIO3_D12	GPIO_AD_B1_12 pin 38 (T4.1)	ALT9
	FLEXIO3_D13	GPIO_AD_B1_13 pin 39 (T4.1)	ALT9
	FLEXIO3_D14	GPIO_AD_B1_14 pin 26	ALT9
	FLEXIO3_D15	GPIO_AD_B1_15 pin 27	ALT9
	FLEXIO3_D16	GPIO_B1_00 pin 8	ALT9
	FLEXIO3_D17	GPIO_B1_01 pin 7	ALT9
	FLEXIO3_D18	GPIO_B1_02 pin 36 (T4.1)	ALT9
	FLEXIO3_D19	GPIO_B1_03 pin 37 (T4.1)	ALT9
	FLEXIO3_D20	GPIO_B1_04	ALT9
	FLEXIO3_D21	GPIO_B1_05	ALT9
	FLEXIO3_D22	GPIO_B1_06	ALT9
	FLEXIO3_D23	GPIO_B1_07	ALT9
	FLEXIO3_D24	GPIO_B1_08	ALT9

Table continues on the next page...

Table 10-1. Muxing Options (continued)

	FLEXIO3_D25	GPIO_B1_09	ALT9
	FLEXIO3_D26	GPIO_B1_10	ALT9
	FLEXIO3_D27	GPIO_B1_11	ALT9
	FLEXIO3_D28	GPIO_B1_12 pin 35 (T4.1)	ALT9
	FLEXIO3_D29	GPIO_B1_13 pin 34 (T4.1)	ALT9
	FLEXIO3_D30	GPIO_B1_14	ALT9
	FLEXIO3_D31	GPIO_B1_15	ALT9
FLEXPWM1	FLEXPWM1_PWM0_A	GPIO_EMC_23	ALT1
		GPIO_SD_B0_00	ALT1
	FLEXPWM1_PWM0_B	GPIO_SD_B0_01	ALT1
		GPIO_EMC_24	ALT1
	FLEXPWM1_PWM1_A	GPIO_EMC_25	ALT1
		GPIO_SD_B0_02	ALT1
	FLEXPWM1_PWM1_B	GPIO_EMC_26	ALT1
		GPIO_SD_B0_03	ALT1
	FLEXPWM1_PWM2_A	GPIO_SD_B0_04	ALT1
		GPIO_EMC_27	ALT1
	FLEXPWM1_PWM2_B	GPIO_SD_B0_05	ALT1
		GPIO_EMC_28	ALT1
	FLEXPWM1_PWM3_A	GPIO_EMC_38	ALT1
		GPIO_SD_B1_00	ALT2
		GPIO_AD_B0_10	ALT1
		GPIO_EMC_12	ALT4
		GPIO_B1_00 pin 8	ALT6
	FLEXPWM1_PWM3_B	GPIO_EMC_39	ALT1
		GPIO_B1_01 pin 7	ALT6
		GPIO_AD_B0_11	ALT1
		GPIO_SD_B1_01	ALT2
		GPIO_EMC_13	ALT4
	FLEXPWM1_PWM0_X	GPIO_AD_B0_02 pin 1	ALT4
	FLEXPWM1_PWM1_X	GPIO_AD_B0_03 pin 0	ALT4
	FLEXPWM1_PWM2_X	GPIO_AD_B0_12 pin 24	ALT4
	FLEXPWM1_PWM3_X	GPIO_AD_B0_13 pin 25	ALT4
FLEXPWM2	FLEXPWM2_PWM0_A	GPIO_B0_06 pin 42 (MM)	ALT2
		GPIO_EMC_06 pin 4	ALT1
	FLEXPWM2_PWM0_B	GPIO_B0_07 pin 43 (MM)	ALT2
		GPIO_EMC_07 pin 33	ALT1
	FLEXPWM2_PWM1_A	GPIO_EMC_08 pin 5	ALT1
		GPIO_B0_08 pin 44 (MM)	ALT2
	FLEXPWM2_PWM1_B	GPIO_EMC_09	ALT1

Table continues on the next page...

Table 10-1. Muxing Options (continued)

		GPIO_B0_09 pin 45 (MM)	ALT2
	FLEXPWM2_PWM2_A	GPIO_EMC_10	ALT1
		GPIO_B0_10 pin 6	ALT2
	FLEXPWM2_PWM2_B	GPIO_EMC_11	ALT1
		GPIO_B0_11 pin 9	ALT2
	FLEXPWM2_PWM3_A	GPIO_AD_B0_09	ALT1
		GPIO_SD_B1_02	ALT2
		GPIO_EMC_19	ALT1
		GPIO_B1_02 pin 36 (T4.1)	ALT6
		GPIO_AD_B0_00	ALT0
	FLEXPWM2_PWM3_B	GPIO_AD_B0_01	ALT0
		GPIO_SD_B1_03	ALT2
		GPIO_B1_03 pin 37 (T4.1)	ALT6
		GPIO_EMC_20	ALT1
FLEXPWM3	FLEXPWM3_PWM0_A	GPIO_EMC_29	ALT1
	FLEXPWM3_PWM0_B	GPIO_EMC_30	ALT1
	FLEXPWM3_PWM1_A	GPIO_EMC_31 pin 29	ALT1
	FLEXPWM3_PWM1_B	GPIO_EMC_32 pin 28	ALT1
	FLEXPWM3_PWM2_A	GPIO_EMC_33	ALT1
	FLEXPWM3_PWM2_B	GPIO_EMC_34	ALT1
	FLEXPWM3_PWM3_A	GPIO_EMC_21	ALT1
	FLEXPWM3_PWM3_B	GPIO_EMC_22	ALT1
FLEXPWM4	FLEXPWM4_PWM0_A	GPIO_AD_B1_08 pin 22	ALT1
		GPIO_EMC_00	ALT1
	FLEXPWM4_PWM0_B	GPIO_EMC_01	ALT1
	FLEXPWM4_PWM1_A	GPIO_AD_B1_09 pin 23	ALT1
		GPIO_EMC_02	ALT1
	FLEXPWM4_PWM1_B	GPIO_EMC_03	ALT1
	FLEXPWM4_PWM2_A	GPIO_B1_14	ALT1
		GPIO_EMC_04 pin 2	ALT1
	FLEXPWM4_PWM2_B	GPIO_EMC_05 pin 3	ALT1
	FLEXPWM4_PWM3_A	GPIO_B1_15	ALT1
		GPIO_EMC_17	ALT1
	FLEXPWM4_PWM3_B	GPIO_EMC_18	ALT1
FLEXCAN1	FLEXCAN1_TX	GPIO_AD_B1_08 pin 22	ALT2
		GPIO_B0_02 pin 11	ALT2
		GPIO_EMC_17	ALT3
		GPIO_SD_B1_02	ALT4
	FLEXCAN1_RX	GPIO_AD_B1_09 pin 23	ALT2
		GPIO_B0_03 pin 13	ALT2

Table continues on the next page...

Table 10-1. Muxing Options (continued)

		GPIO_EMC_18	ALT3
		GPIO_SD_B1_03	ALT4
FLEXCAN2	FLEXCAN2_TX	GPIO_AD_B0_02 pin 1	ALT0
		GPIO_EMC_09	ALT3
		GPIO_B1_08	ALT6
		GPIO_AD_B0_14	ALT6
	FLEXCAN2_RX	GPIO_AD_B0_03 pin 0	ALT0
		GPIO_EMC_10	ALT3
		GPIO_AD_B0_15	ALT6
		GPIO_B1_09	ALT6
FLEXCAN3	FLEXCAN3_RX	GPIO_AD_B0_11	ALT8
		GPIO_AD_B0_15	ALT8
		GPIO_EMC_37 pin 30	ALT9
	FLEXCAN3_TX	GPIO_AD_B0_10	ALT8
		GPIO_AD_B0_14	ALT8
		GPIO_EMC_36 pin 31	ALT9
FLEXSPI A	FLEXSPI_A_DATA0	GPIO_AD_B1_13 pin 39 (T4.1)	ALT0
		GPIO_SD_B1_08	ALT1
	FLEXSPI_A_DATA1	GPIO_AD_B1_12 pin 38 (T4.1)	ALT0
		GPIO_SD_B1_09	ALT1
	FLEXSPI_A_DATA2	GPIO_AD_B1_11 pin 21	ALT0
		GPIO_SD_B1_10	ALT1
	FLEXSPI_A_DATA3	GPIO_AD_B1_10 pin 20	ALT0
		GPIO_SD_B1_11	ALT1
	FLEXSPI_A_DQS	GPIO_AD_B1_09 pin 23	ALT0
		GPIO_SD_B1_05	ALT1
	FLEXSPI_A_SCLK	GPIO_AD_B1_14 pin 26	ALT0
		GPIO_SD_B1_07	ALT1
	FLEXSPI_A_SS0_B	GPIO_AD_B1_15 pin 27	ALT0
		GPIO_SD_B1_06	ALT1
	FLEXSPI_A_SS1_B	GPIO_AD_B1_08 pin 22	ALT0
		GPIO_SD_B0_00	ALT6
		GPIO_SD_B1_04	ALT4
FLEXSPI B	FLEXSPI_B_DATA0	GPIO_AD_B1_07 pin 16	ALT0
		GPIO_SD_B1_03	ALT1
	FLEXSPI_B_DATA1	GPIO_AD_B1_06 pin 17	ALT0
		GPIO_SD_B1_02	ALT1
	FLEXSPI_B_DATA2	GPIO_AD_B1_05 pin 41 (T4.1)	ALT0
		GPIO_SD_B1_01	ALT1
	FLEXSPI_B_DATA3	GPIO_AD_B1_04 pin 40 (T4.1)	ALT0

Table continues on the next page...

Table 10-1. Muxing Options (continued)

		GPIO_SD_B1_00	ALT1
	FLEXSPI_B_DQS	GPIO_SD_B0_05	ALT4
	FLEXSPI_B_SCLK	GPIO_SD_B1_04	ALT1
	FLEXSPI_B_SS0_B	GPIO_SD_B0_04	ALT4
		GPIO_SD_B1_05	ALT4
	FLEXSPI_B_SS1_B	GPIO_SD_B0_01	ALT6
FLEXSPI2 A	FLEXSPI2_A_DATA0	GPIO_EMC_26	ALT8
	FLEXSPI2_A_DATA1	GPIO_EMC_27	ALT8
	FLEXSPI2_A_DATA2	GPIO_EMC_28	ALT8
	FLEXSPI2_A_DATA3	GPIO_EMC_29	ALT8
	FLEXSPI2_A_DQS	GPIO_EMC_23	ALT8
	FLEXSPI2_A_SCLK	GPIO_EMC_25	ALT8
	FLEXSPI2_A_SS0_B	GPIO_EMC_24	ALT8
	FLEXSPI2_A_SS1_B	GPIO_EMC_22	ALT8
FLEXSPI2 B	FLEXSPI2_B_DATA0	GPIO_EMC_13	ALT8
	FLEXSPI2_B_DATA1	GPIO_EMC_14	ALT8
	FLEXSPI2_B_DATA2	GPIO_EMC_15	ALT8
	FLEXSPI2_B_DATA3	GPIO_EMC_16	ALT8
	FLEXSPI2_B_DQS	GPIO_EMC_11	ALT8
	FLEXSPI2_B_SCLK	GPIO_EMC_12	ALT8
	FLEXSPI2_B_SS0_B	GPIO_EMC_10	ALT8
	FLEXSPI2_B_SS1_B	GPIO_EMC_09	ALT8
GPIO1 GPIO6 is used to access GPIO1 pins as fast GPIO	GPIO1_IO00	GPIO_AD_B0_00	ALT5
	GPIO1_IO01	GPIO_AD_B0_01	ALT5
	GPIO1_IO02	GPIO_AD_B0_02 pin 1	ALT5
	GPIO1_IO03	GPIO_AD_B0_03 pin 0	ALT5
	GPIO1_IO04	GPIO_AD_B0_04	ALT5
	GPIO1_IO05	GPIO_AD_B0_05	ALT5
	GPIO1_IO06	GPIO_AD_B0_06	ALT5
	GPIO1_IO07	GPIO_AD_B0_07	ALT5
	GPIO1_IO08	GPIO_AD_B0_08	ALT5
	GPIO1_IO09	GPIO_AD_B0_09	ALT5
	GPIO1_IO10	GPIO_AD_B0_10	ALT5
	GPIO1_IO11	GPIO_AD_B0_11	ALT5
	GPIO1_IO12	GPIO_AD_B0_12 pin 24	ALT5
	GPIO1_IO13	GPIO_AD_B0_13 pin 25	ALT5
	GPIO1_IO14	GPIO_AD_B0_14	ALT5
	GPIO1_IO15	GPIO_AD_B0_15	ALT5
	GPIO1_IO16	GPIO_AD_B1_00 pin 19	ALT5
	GPIO1_IO17	GPIO_AD_B1_01 pin 18	ALT5

Table continues on the next page...

Table 10-1. Muxing Options (continued)

	GPIO1_IO18	GPIO_AD_B1_02 pin 14	ALT5
	GPIO1_IO19	GPIO_AD_B1_03 pin 15	ALT5
	GPIO1_IO20	GPIO_AD_B1_04 pin 40 (T4.1)	ALT5
	GPIO1_IO21	GPIO_AD_B1_05 pin 41 (T4.1)	ALT5
	GPIO1_IO22	GPIO_AD_B1_06 pin 17	ALT5
	GPIO1_IO23	GPIO_AD_B1_07 pin 16	ALT5
	GPIO1_IO24	GPIO_AD_B1_08 pin 22	ALT5
	GPIO1_IO25	GPIO_AD_B1_09 pin 23	ALT5
	GPIO1_IO26	GPIO_AD_B1_10 pin 20	ALT5
	GPIO1_IO27	GPIO_AD_B1_11 pin 21	ALT5
	GPIO1_IO28	GPIO_AD_B1_12 pin 38 (T4.1)	ALT5
	GPIO1_IO29	GPIO_AD_B1_13 pin 39 (T4.1)	ALT5
	GPIO1_IO30	GPIO_AD_B1_14 pin 26	ALT5
	GPIO1_IO31	GPIO_AD_B1_15 pin 27	ALT5
GPIO2 GPIO7 is used to access GPIO2 pins as fast GPIO	GPIO2_IO00	GPIO_B0_00 pin 10	ALT5
	GPIO2_IO01	GPIO_B0_01 pin 12	ALT5
	GPIO2_IO02	GPIO_B0_02 pin 11	ALT5
	GPIO2_IO03	GPIO_B0_03 pin 13	ALT5
	GPIO2_IO04	GPIO_B0_04 pin 40 (MM)	ALT5
	GPIO2_IO05	GPIO_B0_05 pin 41 (MM)	ALT5
	GPIO2_IO06	GPIO_B0_06 pin 42 (MM)	ALT5
	GPIO2_IO07	GPIO_B0_07 pin 43 (MM)	ALT5
	GPIO2_IO08	GPIO_B0_08 pin 44 (MM)	ALT5
	GPIO2_IO09	GPIO_B0_09 pin 45 (MM)	ALT5
	GPIO2_IO10	GPIO_B0_10 pin 6	ALT5
	GPIO2_IO11	GPIO_B0_11 pin 9	ALT5
	GPIO2_IO12	GPIO_B0_12 pin 32	ALT5
	GPIO2_IO13	GPIO_B0_13	ALT5
	GPIO2_IO14	GPIO_B0_14	ALT5
	GPIO2_IO15	GPIO_B0_15	ALT5
	GPIO2_IO16	GPIO_B1_00 pin 8	ALT5
	GPIO2_IO17	GPIO_B1_01 pin 7	ALT5
	GPIO2_IO18	GPIO_B1_02 pin 36 (T4.1)	ALT5
	GPIO2_IO19	GPIO_B1_03 pin 37 (T4.1)	ALT5
	GPIO2_IO20	GPIO_B1_04	ALT5
	GPIO2_IO21	GPIO_B1_05	ALT5
	GPIO2_IO22	GPIO_B1_06	ALT5
	GPIO2_IO23	GPIO_B1_07	ALT5
	GPIO2_IO24	GPIO_B1_08	ALT5
	GPIO2_IO25	GPIO_B1_09	ALT5

Table continues on the next page...

Table 10-1. Muxing Options (continued)

	GPIO2_IO26	GPIO_B1_10	ALT5
	GPIO2_IO27	GPIO_B1_11	ALT5
	GPIO2_IO28	GPIO_B1_12 pin 35 (T4.1)	ALT5
	GPIO2_IO29	GPIO_B1_13 pin 34 (T4.1)	ALT5
	GPIO2_IO30	GPIO_B1_14	ALT5
	GPIO2_IO31	GPIO_B1_15	ALT5
GPIO3 GPIO8 is used to access GPIO3 pins as fast GPIO	GPIO3_IO00	GPIO_SD_B1_00	ALT5
	GPIO3_IO01	GPIO_SD_B1_01	ALT5
	GPIO3_IO02	GPIO_SD_B1_02	ALT5
	GPIO3_IO03	GPIO_SD_B1_03	ALT5
	GPIO3_IO04	GPIO_SD_B1_04	ALT5
	GPIO3_IO05	GPIO_SD_B1_05	ALT5
	GPIO3_IO06	GPIO_SD_B1_06	ALT5
	GPIO3_IO07	GPIO_SD_B1_07	ALT5
	GPIO3_IO08	GPIO_SD_B1_08	ALT5
	GPIO3_IO09	GPIO_SD_B1_09	ALT5
	GPIO3_IO10	GPIO_SD_B1_10	ALT5
	GPIO3_IO11	GPIO_SD_B1_11	ALT5
	GPIO3_IO12	GPIO_SD_B0_00	ALT5
	GPIO3_IO13	GPIO_SD_B0_01	ALT5
	GPIO3_IO14	GPIO_SD_B0_02	ALT5
	GPIO3_IO15	GPIO_SD_B0_03	ALT5
	GPIO3_IO16	GPIO_SD_B0_04	ALT5
	GPIO3_IO17	GPIO_SD_B0_05	ALT5
	GPIO3_IO18	GPIO_EMC_32 pin 28	ALT5
	GPIO3_IO19	GPIO_EMC_33	ALT5
	GPIO3_IO20	GPIO_EMC_34	ALT5
	GPIO3_IO21	GPIO_EMC_35	ALT5
	GPIO3_IO22	GPIO_EMC_36 pin 31	ALT5
	GPIO3_IO23	GPIO_EMC_37 pin 30	ALT5
	GPIO3_IO24	GPIO_EMC_38	ALT5
	GPIO3_IO25	GPIO_EMC_39	ALT5
	GPIO3_IO26	GPIO_EMC_40	ALT5
	GPIO3_IO27	GPIO_EMC_41	ALT5
GPIO4 GPIO9 is used to access GPIO4 pins as fast GPIO	GPIO4_IO0	GPIO_EMC_00	ALT5
	GPIO4_IO1	GPIO_EMC_01	ALT5
	GPIO4_IO2	GPIO_EMC_02	ALT5
	GPIO4_IO3	GPIO_EMC_03	ALT5
	GPIO4_IO4	GPIO_EMC_04 pin 2	ALT5
	GPIO4_IO5	GPIO_EMC_05 pin 3	ALT5

Table continues on the next page...

Table 10-1. Muxing Options (continued)

	GPIO4_IO6	GPIO_EMC_06 pin 4	ALT5
	GPIO4_IO7	GPIO_EMC_07 pin 33	ALT5
	GPIO4_IO8	GPIO_EMC_08 pin 5	ALT5
	GPIO4_IO9	GPIO_EMC_09	ALT5
	GPIO4_IO10	GPIO_EMC_10	ALT5
	GPIO4_IO11	GPIO_EMC_11	ALT5
	GPIO4_IO12	GPIO_EMC_12	ALT5
	GPIO4_IO13	GPIO_EMC_13	ALT5
	GPIO4_IO14	GPIO_EMC_14	ALT5
	GPIO4_IO15	GPIO_EMC_15	ALT5
	GPIO4_IO16	GPIO_EMC_16	ALT5
	GPIO4_IO17	GPIO_EMC_17	ALT5
	GPIO4_IO18	GPIO_EMC_18	ALT5
	GPIO4_IO19	GPIO_EMC_19	ALT5
	GPIO4_IO20	GPIO_EMC_20	ALT5
	GPIO4_IO21	GPIO_EMC_21	ALT5
	GPIO4_IO22	GPIO_EMC_22	ALT5
	GPIO4_IO23	GPIO_EMC_23	ALT5
	GPIO4_IO24	GPIO_EMC_24	ALT5
	GPIO4_IO25	GPIO_EMC_25	ALT5
	GPIO4_IO26	GPIO_EMC_26	ALT5
	GPIO4_IO27	GPIO_EMC_27	ALT5
	GPIO4_IO28	GPIO_EMC_28	ALT5
	GPIO4_IO29	GPIO_EMC_29	ALT5
	GPIO4_IO30	GPIO_EMC_30	ALT5
	GPIO4_IO31	GPIO_EMC_31 pin 29	ALT5
GPIO5	GPIO5_IO00	WAKEUP	ALT5
	GPIO5_IO01	PMIC_ON_REQ	ALT5
	GPIO5_IO02	PMIC_STBY_REQ	ALT5
GPT1	GPT1_CLK	GPIO_AD_B0_13 pin 25	ALT1
		GPIO_B1_04	ALT8
	GPT1_CAPTURE1	GPIO_EMC_24	ALT4
		GPIO_B1_05	ALT8
	GPT1_CAPTURE2	GPIO_EMC_23	ALT4
		GPIO_B1_06	ALT8
	GPT1_COMPARE1	GPIO_EMC_35	ALT2
		GPIO_B1_07	ALT8
	GPT1_COMPARE2	GPIO_EMC_36 pin 31	ALT2
		GPIO_B1_08	ALT8
	GPT1_COMPARE3	GPIO_EMC_37 pin 30	ALT2

Table continues on the next page...

Table 10-1. Muxing Options (continued)

		GPIO_B1_09	ALT8
GPT2	GPT2_CLK	GPIO_AD_B0_09	ALT7
		GPIO_AD_B1_02 pin 14	ALT8
	GPT2_CAPTURE1	GPIO_EMC_41	ALT1
		GPIO_AD_B1_03 pin 15	ALT8
	GPT2_CAPTURE2	GPIO_EMC_40	ALT1
		GPIO_AD_B1_04 pin 40 (T4.1)	ALT8
	GPT2_COMPARE1	GPIO_AD_B0_06	ALT1
		GPIO_AD_B1_05 pin 41 (T4.1)	ALT8
	GPT2_COMPARE2	GPIO_AD_B0_07	ALT1
		GPIO_AD_B1_06 pin 17	ALT8
	GPT2_COMPARE3	GPIO_AD_B0_08	ALT1
		GPIO_AD_B1_07 pin 16	ALT8
JTAG	JTAG_TMS	GPIO_AD_B0_06	ALT0
	JTAG_TCK	GPIO_AD_B0_07	ALT0
	JTAG_MOD	GPIO_AD_B0_08	ALT0
	JTAG_TDI	GPIO_AD_B0_09	ALT0
	JTAG_TDO	GPIO_AD_B0_10	ALT0
	JTAG_TRSTB	GPIO_AD_B0_11	ALT0
	JTAG_DE_B	GPIO_EMC_01	ALT7
SWD	SWD_DIO	GPIO_AD_B0_06	ALT0
	SWD_CLK	GPIO_AD_B0_07	ALT0
TPIU	ARM_TRACE0	GPIO_B0_04 pin 40 (MM)	ALT3
	ARM_TRACE1	GPIO_B0_05 pin 41 (MM)	ALT3
	ARM_TRACE2	GPIO_B0_06 pin 42 (MM)	ALT3
	ARM_TRACE3	GPIO_B0_07 pin 43 (MM)	ALT3
	ARM_TRACE_CLK	GPIO_B0_12 pin 32	ALT2
	ARM_TRACE_SWO	GPIO_B0_13	ALT2
KPP	KPP_ROW0	GPIO_AD_B1_14 pin 26	ALT7
	KPP_COL0	GPIO_AD_B1_15 pin 27	ALT7
	KPP_ROW1	GPIO_AD_B1_12 pin 38 (T4.1)	ALT7
	KPP_COL1	GPIO_AD_B1_13 pin 39 (T4.1)	ALT7
	KPP_ROW2	GPIO_AD_B1_10 pin 20	ALT7
	KPP_COL2	GPIO_AD_B1_11 pin 21	ALT7
	KPP_ROW3	GPIO_AD_B1_08 pin 22	ALT7
	KPP_COL3	GPIO_AD_B1_09 pin 23	ALT7
	KPP_ROW4	GPIO_AD_B1_06 pin 17	ALT7
	KPP_COL4	GPIO_AD_B1_07 pin 16	ALT7
	KPP_ROW5	GPIO_AD_B1_04 pin 40 (T4.1)	ALT7
	KPP_COL5	GPIO_AD_B1_05 pin 41 (T4.1)	ALT7

Table continues on the next page...

Table 10-1. Muxing Options (continued)

	KPP_ROW6	GPIO_AD_B1_02 pin 14	ALT7
	KPP_COL6	GPIO_AD_B1_03 pin 15	ALT7
	KPP_ROW7	GPIO_AD_B1_00 pin 19	ALT7
	KPP_COL7	GPIO_AD_B1_01 pin 18	ALT7
LCD	LCD_CLK	GPIO_B0_00 pin 10	ALT0
	LCD_ENABLE	GPIO_B0_01 pin 12	ALT0
	LCD_HSYNC	GPIO_B0_02 pin 11	ALT0
	LCD_VSYNC	GPIO_B0_03 pin 13	ALT0
	LCD_DATA00	GPIO_B0_04 pin 40 (MM)	ALT0
	LCD_DATA01	GPIO_B0_05 pin 41 (MM)	ALT0
	LCD_DATA02	GPIO_B0_06 pin 42 (MM)	ALT0
	LCD_DATA03	GPIO_B0_07 pin 43 (MM)	ALT0
	LCD_DATA04	GPIO_B0_08 pin 44 (MM)	ALT0
	LCD_DATA05	GPIO_B0_09 pin 45 (MM)	ALT0
	LCD_DATA06	GPIO_B0_10 pin 6	ALT0
	LCD_DATA07	GPIO_B0_11 pin 9	ALT0
	LCD_DATA08	GPIO_B0_12 pin 32	ALT0
	LCD_DATA09	GPIO_B0_13	ALT0
	LCD_DATA10	GPIO_B0_14	ALT0
	LCD_DATA11	GPIO_B0_15	ALT0
	LCD_DATA12	GPIO_B1_00 pin 8	ALT0
	LCD_DATA13	GPIO_B1_01 pin 7	ALT0
	LCD_DATA14	GPIO_B1_02 pin 36 (T4.1)	ALT0
	LCD_DATA15	GPIO_B1_03 pin 37 (T4.1)	ALT0
	LCD_DATA16	GPIO_B1_04	ALT0
	LCD_DATA17	GPIO_B1_05	ALT0
	LCD_DATA18	GPIO_B1_06	ALT0
	LCD_DATA19	GPIO_B1_07	ALT0
	LCD_DATA20	GPIO_B1_08	ALT0
	LCD_DATA21	GPIO_B1_09	ALT0
	LCD_DATA22	GPIO_B1_10	ALT0
	LCD_DATA23	GPIO_B1_11	ALT0
LPI2C1 Wire	LPI2C1_SCL	GPIO_AD_B1_00 pin 19	ALT3
		GPIO_SD_B1_04	ALT2
	LPI2C1_SDA	GPIO_SD_B1_05	ALT2
		GPIO_AD_B1_01 pin 18	ALT3
	LPI2C1_SCLS	GPIO_AD_B0_00	ALT4
	LPI2C1_SDAS	GPIO_AD_B0_01	ALT4
	LPI2C1_HREQ	GPIO_AD_B0_02 pin 1	ALT6
LPI2C2 Wire3 on MM	LPI2C2_SCL	GPIO_B0_04 pin 40 (MM)	ALT2

Table continues on the next page...

Table 10-1. Muxing Options (continued)

Wire3 on MM		GPIO_SD_B1_11	ALT3
	LPI2C2_SDA	GPIO_B0_05 pin 41 (MM)	ALT2
		GPIO_SD_B1_10	ALT3
LPI2C3 Wire1 (Wire2 on MM)	LPI2C3_SCL	GPIO_AD_B1_07 pin 16	ALT1
		GPIO_EMC_22	ALT2
		GPIO_SD_B0_00	ALT2
	LPI2C3_SDA	GPIO_AD_B1_06 pin 17	ALT1
		GPIO_EMC_21	ALT2
		GPIO_SD_B0_01	ALT2
LPI2C4 Wire2 (Wire1 on MM)	LPI2C4_SCL	GPIO_AD_B0_12 pin 24	ALT0
		GPIO_EMC_12	ALT2
	LPI2C4_SDA	GPIO_EMC_11	ALT2
		GPIO_AD_B0_13 pin 25	ALT0
LPSPI1 SPI2	LPSPI1_PCS0	GPIO_EMC_30	ALT3
		GPIO_SD_B0_01	ALT4
	LPSPI1_PCS1	GPIO_EMC_31 pin 29	ALT3
	LPSPI1_PCS2	GPIO_EMC_40	ALT2
	LPSPI1_PCS3	GPIO_EMC_41	ALT2
	LPSPI1_SCK	GPIO_SD_B0_00	ALT4
		GPIO_EMC_27	ALT3
	LPSPI1_SOUT	GPIO_SD_B0_02	ALT4
		GPIO_EMC_28	ALT3
	LPSPI1_SIN	GPIO_EMC_29	ALT3
		GPIO_SD_B0_03	ALT4
LPSPI2	LPSPI2_PCS0	GPIO_SD_B1_06	ALT4
		GPIO_EMC_01	ALT2
	LPSPI2_PCS1	GPIO_EMC_14	ALT4
	LPSPI2_PCS2	GPIO_SD_B1_10	ALT4
	LPSPI2_PCS3	GPIO_SD_B1_11	ALT4
	LPSPI2_SCK	GPIO_SD_B1_07	ALT4
		GPIO_EMC_00	ALT2
	LPSPI2_SOUT	GPIO_EMC_02	ALT2
		GPIO_SD_B1_08	ALT4
	LPSPI2_SIN	GPIO_SD_B1_09	ALT4
		GPIO_EMC_03	ALT2
LPSPI3 SPI1	LPSPI3_PCS0	GPIO_AD_B1_12 pin 38 (T4.1)	ALT2
		GPIO_AD_B0_03 pin 0	ALT7
	LPSPI3_PCS1	GPIO_AD_B0_04	ALT7
	LPSPI3_PCS2	GPIO_AD_B0_05	ALT7
	LPSPI3_PCS3	GPIO_AD_B0_06	ALT7

Table continues on the next page...

Table 10-1. Muxing Options (continued)

	LPSPI3_SCK	GPIO_AD_B1_15 pin 27	ALT2
		GPIO_AD_B0_00	ALT7
	LPSPI3_SOUT	GPIO_AD_B0_01	ALT7
		GPIO_AD_B1_14 pin 26	ALT2
	LPSPI3_SIN	GPIO_AD_B1_13 pin 39 (T4.1)	ALT2
		GPIO_AD_B0_02 pin 1	ALT7
LPSPI4 SPI	LPSPI4_PCS0	GPIO_B1_04	ALT1
		GPIO_B0_00 pin 10	ALT3
	LPSPI4_PCS1	GPIO_B1_03 pin 37 (T4.1)	ALT2
	LPSPI4_PCS2	GPIO_B1_02 pin 36 (T4.1)	ALT2
	LPSPI4_PCS3	GPIO_B1_11	ALT6
	LPSPI4_SCK	GPIO_B1_07	ALT1
		GPIO_B0_03 pin 13	ALT3
	LPSPI4_SOUT	GPIO_B1_06	ALT1
		GPIO_B0_02 pin 11	ALT3
	LPSPI4_SIN	GPIO_B1_05	ALT1
		GPIO_B0_01 pin 12	ALT3
LPUART1 Serial6	LPUART1_TXD	GPIO_AD_B0_12 pin 24	ALT2
	LPUART1_RXD	GPIO_AD_B0_13 pin 25	ALT2
	LPUART1_CTS_B	GPIO_AD_B0_14	ALT2
	LPUART1_RTS_B	GPIO_AD_B0_15	ALT2
LPUART2 Serial3	LPUART2_TXD	GPIO_SD_B1_11	ALT2
		GPIO_AD_B1_02 pin 14	ALT2
	LPUART2_RXD	GPIO_AD_B1_03 pin 15	ALT2
		GPIO_SD_B1_10	ALT2
	LPUART2_CTS_B	GPIO_AD_B1_00 pin 19	ALT2
	LPUART2_RTS_B	GPIO_AD_B1_01 pin 18	ALT2
LPUART3 Serial2 (or Serial4 on MicroMod)	LPUART3_TXD	GPIO_AD_B1_06 pin 17	ALT2
		GPIO_B0_08 pin 44 (MM)	ALT3
		GPIO_EMC_13	ALT2
	LPUART3_RXD	GPIO_EMC_14	ALT2
		GPIO_B0_09 pin 45 (MM)	ALT3
		GPIO_AD_B1_07 pin 16	ALT2
	LPUART3_CTS_B	GPIO_EMC_15	ALT2
		GPIO_AD_B1_04 pin 40 (T4.1)	ALT2
	LPUART3_RTS_B	GPIO_AD_B1_05 pin 41 (T4.1)	ALT2
		GPIO_EMC_16	ALT2
LPUART4 Serial4 (or Serial2 on MicroMod)	LPUART4_TXD	GPIO_SD_B1_00	ALT4
		GPIO_B1_00 pin 8	ALT2
		GPIO_EMC_19	ALT2

Table continues on the next page...

Table 10-1. Muxing Options (continued)

	LPUART4_RXD	GPIO_SD_B1_01	ALT4
		GPIO_EMC_20	ALT2
		GPIO_B1_01 pin 7	ALT2
	LPUART4_CTS_B	GPIO_EMC_17	ALT2
	LPUART4_RTS_B	GPIO_EMC_18	ALT2
LPUART5 Serial8	LPUART5_TXD	GPIO_EMC_23	ALT2
		GPIO_B1_12 pin 35 (T4.1)	ALT1
	LPUART5_RXD	GPIO_B1_13 pin 34 (T4.1)	ALT1
		GPIO_EMC_24	ALT2
	LPUART5_CTS_B	GPIO_EMC_28	ALT2
	LPUART5_RTS_B	GPIO_EMC_27	ALT2
LPUART6 Serial1	LPUART6_TXD	GPIO_EMC_25	ALT2
		GPIO_AD_B0_02 pin 1	ALT2
	LPUART6_RXD	GPIO_AD_B0_03 pin 0	ALT2
		GPIO_EMC_26	ALT2
	LPUART6_CTS_B	GPIO_EMC_30	ALT2
	LPUART6_RTS_B	GPIO_EMC_29	ALT2
LPUART7 Serial7	LPUART7_TXD	GPIO_SD_B1_08	ALT2
		GPIO_EMC_31 pin 29	ALT2
	LPUART7_RXD	GPIO_EMC_32 pin 28	ALT2
		GPIO_SD_B1_09	ALT2
	LPUART7_CTS_B	GPIO_SD_B1_06	ALT2
	LPUART7_RTS_B	GPIO_SD_B1_07	ALT2
LPUART8 Serial5	LPUART8_TXD	GPIO_EMC_38	ALT2
		GPIO_AD_B1_10 pin 20	ALT2
		GPIO_SD_B0_04	ALT2
	LPUART8_RXD	GPIO_SD_B0_05	ALT2
		GPIO_AD_B1_11 pin 21	ALT2
		GPIO_EMC_39	ALT2
	LPUART8_CTS_B	GPIO_SD_B0_02	ALT2
	LPUART8_RTS_B	GPIO_SD_B0_03	ALT2
MQS	MQS_RIGHT	GPIO_AD_B0_04	ALT1
		GPIO_B0_00 pin 10	ALT2
		GPIO_EMC_13	ALT3
	MQS_LEFT	GPIO_EMC_14	ALT3
		GPIO_B0_01 pin 12	ALT2
		GPIO_AD_B0_05	ALT1
PIT	PIT_TRIGGER0	GPIO_AD_B0_04	ALT6
TMR1 (QUAD TIMER)	TMR1_TIMER0	GPIO_B0_00 pin 10	ALT1
	TMR1_TIMER1	GPIO_B0_01 pin 12	ALT1

Table continues on the next page...

Table 10-1. Muxing Options (continued)

	TMR1_TIMER2	GPIO_B0_02 pin 11	ALT1
	TMR1_TIMER3	GPIO_B1_08	ALT1
TMR2 (QUAD TIMER)	TMR2_TIMER0	GPIO_EMC_19	ALT4
		GPIO_B0_03 pin 13	ALT1
	TMR2_TIMER1	GPIO_EMC_20	ALT4
		GPIO_B0_04 pin 40 (MM)	ALT1
	TMR2_TIMER2	GPIO_EMC_21	ALT4
		GPIO_B0_05 pin 41 (MM)	ALT1
	TMR2_TIMER3	GPIO_EMC_22	ALT4
		GPIO_B1_09	ALT1
TMR3 (QUAD TIMER)	TMR3_TIMER0	GPIO_AD_B1_00 pin 19	ALT1
		GPIO_EMC_15	ALT4
		GPIO_B0_06 pin 42 (MM)	ALT1
	TMR3_TIMER1	GPIO_AD_B1_01 pin 18	ALT1
		GPIO_EMC_16	ALT4
		GPIO_B0_07 pin 43 (MM)	ALT1
	TMR3_TIMER2	GPIO_EMC_17	ALT4
		GPIO_AD_B1_02 pin 14	ALT1
		GPIO_B0_08 pin 44 (MM)	ALT1
	TMR3_TIMER3	GPIO_B1_10	ALT1
		GPIO_AD_B1_03 pin 15	ALT1
		GPIO_EMC_18	ALT4
TMR4 (QUAD TIMER)	TMR4_TIMER0	GPIO_B0_09 pin 45 (MM)	ALT1
	TMR4_TIMER1	GPIO_B0_10 pin 6	ALT1
	TMR4_TIMER2	GPIO_B0_11 pin 9	ALT1
	TMR4_TIMER3	GPIO_B1_11	ALT1
SAI1	SAI1_TX_DATA0	GPIO_AD_B1_13 pin 39 (T4.1)	ALT3
		GPIO_B1_01 pin 7	ALT3
		GPIO_SD_B1_07	ALT3
	SAI1_TX_DATA1 and SAI1_RX_DATA3	GPIO_B0_12 pin 32	ALT3
		GPIO_SD_B1_02	ALT3
	SAI1_TX_DATA2 and SAI1_RX_DATA2	GPIO_B0_11 pin 9	ALT3
		GPIO_SD_B1_01	ALT3
	SAI1_TX_DATA3 and SAI1_RX_DATA1	GPIO_B0_10 pin 6	ALT3
		GPIO_SD_B1_00	ALT3
	SAI1_TX_BCLK	GPIO_SD_B1_08	ALT3
		GPIO_B1_02 pin 36 (T4.1)	ALT3
		GPIO_AD_B1_14 pin 26	ALT3
	SAI1_TX_SYNC	GPIO_AD_B1_15 pin 27	ALT3
		GPIO_B1_03 pin 37 (T4.1)	ALT3

Table continues on the next page...

Table 10-1. Muxing Options (continued)

		GPIO_SD_B1_09	ALT3
	SAI1_RX_DATA0	GPIO_B1_00 pin 8	ALT3
		GPIO_AD_B1_12 pin 38 (T4.1)	ALT3
		GPIO_SD_B1_06	ALT3
	SAI1_RX_BCLK	GPIO_AD_B1_11 pin 21	ALT3
		GPIO_B0_15	ALT3
		GPIO_SD_B1_05	ALT3
	SAI1_RX_SYNC	GPIO_AD_B1_10 pin 20	ALT3
		GPIO_SD_B1_04	ALT3
		GPIO_B0_14	ALT3
	SAI1_MCLK	GPIO_B0_13	ALT3
		GPIO_SD_B1_03	ALT3
		GPIO_AD_B1_09 pin 23	ALT3
SAI2	SAI2_TX_DATA	GPIO_EMC_04 pin 2	ALT2
		GPIO_AD_B0_09	ALT3
	SAI2_TX_BCLK	GPIO_AD_B0_05	ALT3
		GPIO_EMC_06 pin 4	ALT2
	SAI2_TX_SYNC	GPIO_AD_B0_04	ALT3
		GPIO_EMC_05 pin 3	ALT2
	SAI2_RX_DATA	GPIO_AD_B0_08	ALT3
		GPIO_EMC_08 pin 5	ALT2
	SAI2_RX_BCLK	GPIO_EMC_10	ALT2
		GPIO_AD_B0_06	ALT3
	SAI2_RX_SYNC	GPIO_EMC_09	ALT2
		GPIO_AD_B0_07	ALT3
	SAI2_MCLK	GPIO_EMC_07 pin 33	ALT2
		GPIO_AD_B0_10	ALT3
SAI3	SAI3_TX_DATA	GPIO_EMC_36 pin 31	ALT3
		GPIO_SD_B1_01	ALT8
	SAI3_TX_BCLK	GPIO_EMC_38	ALT3
		GPIO_SD_B1_03	ALT8
	SAI3_TX_SYNC	GPIO_EMC_39	ALT3
		GPIO_SD_B1_02	ALT8
	SAI3_RX_DATA	GPIO_EMC_33	ALT3
		GPIO_SD_B1_00	ALT8
	SAI3_RX_BCLK	GPIO_EMC_35	ALT3
		GPIO_SD_B1_06	ALT8
	SAI3_RX_SYNC	GPIO_EMC_34	ALT3
		GPIO_SD_B1_05	ALT8
	SAI3_MCLK	GPIO_EMC_37 pin 30	ALT3

Table continues on the next page...

Table 10-1. Muxing Options (continued)

		GPIO_SD_B1_04	ALT8
SEMC	SEMC_DATA00	GPIO_EMC_00	ALT0
	SEMC_DATA01	GPIO_EMC_01	ALT0
	SEMC_DATA02	GPIO_EMC_02	ALT0
	SEMC_DATA03	GPIO_EMC_03	ALT0
	SEMC_DATA04	GPIO_EMC_04 pin 2	ALT0
	SEMC_DATA05	GPIO_EMC_05 pin 3	ALT0
	SEMC_DATA06	GPIO_EMC_06 pin 4	ALT0
	SEMC_DATA07	GPIO_EMC_07 pin 33	ALT0
	SEMC_DATA08	GPIO_EMC_30	ALT0
	SEMC_DATA09	GPIO_EMC_31 pin 29	ALT0
	SEMC_DATA10	GPIO_EMC_32 pin 28	ALT0
	SEMC_DATA11	GPIO_EMC_33	ALT0
	SEMC_DATA12	GPIO_EMC_34	ALT0
	SEMC_DATA13	GPIO_EMC_35	ALT0
	SEMC_DATA14	GPIO_EMC_36 pin 31	ALT0
	SEMC_DATA15	GPIO_EMC_37 pin 30	ALT0
	SEMC_ADDR00	GPIO_EMC_09	ALT0
	SEMC_ADDR01	GPIO_EMC_10	ALT0
	SEMC_ADDR02	GPIO_EMC_11	ALT0
	SEMC_ADDR03	GPIO_EMC_12	ALT0
	SEMC_ADDR04	GPIO_EMC_13	ALT0
	SEMC_ADDR05	GPIO_EMC_14	ALT0
	SEMC_ADDR06	GPIO_EMC_15	ALT0
	SEMC_ADDR07	GPIO_EMC_16	ALT0
	SEMC_ADDR08	GPIO_EMC_17	ALT0
	SEMC_ADDR09	GPIO_EMC_18	ALT0
	SEMC_ADDR10	GPIO_EMC_23	ALT0
	SEMC_ADDR11	GPIO_EMC_19	ALT0
	SEMC_ADDR12	GPIO_EMC_20	ALT0
	SEMC_DM0	GPIO_EMC_08 pin 5	ALT0
	SEMC_DM1	GPIO_EMC_38	ALT0
	SEMC_BA0	GPIO_EMC_21	ALT0
	SEMC_BA1	GPIO_EMC_22	ALT0
	SEMC_CAS	GPIO_EMC_24	ALT0
	SEMC_RAS	GPIO_EMC_25	ALT0
	SEMC_CLK	GPIO_EMC_26	ALT0
	SEMC_CKE	GPIO_EMC_27	ALT0
	SEMC_WE	GPIO_EMC_28	ALT0
	SEMC_CS0	GPIO_EMC_29	ALT0

Table continues on the next page...

Table 10-1. Muxing Options (continued)

	SEMC_DQS	GPIO_EMC_39	ALT0
	SEMC_RDY	GPIO_EMC_40	ALT0
	SEMC_CSX0	GPIO_EMC_41	ALT0
	SEMC_CSX1	GPIO_SD_B1_07	ALT0
	SEMC_CSX1	GPIO_B0_00 pin 10	ALT6
	SEMC_CSX2	GPIO_SD_B1_08	ALT6
	SEMC_CSX2	GPIO_B0_01 pin 12	ALT6
	SEMC_CSX3	GPIO_B0_02 pin 11	ALT6
SNVS	SNVS_PMIC_ON_REQ	PMIC_ON_REQ	ALT0
	SNVS_VIO_5_CTL	GPIO_EMC_18	ALT6
	SNVS_VIO_5_B	GPIO_EMC_19	ALT6
SPDIF	SPDIF_OUT	GPIO_EMC_15	ALT3
		GPIO_AD_B1_05 pin 41 (T4.1)	ALT3
		GPIO_AD_B1_02 pin 14	ALT3
	SPDIF_IN	GPIO_EMC_16	ALT3
		GPIO_AD_B1_03 pin 15	ALT3
	SPDIF_SR_CLK	GPIO_AD_B1_04 pin 40 (T4.1)	ALT3
	SPDIF_LOCK	GPIO_AD_B1_06 pin 17	ALT3
	SPDIF_EXT_CLK	GPIO_AD_B1_07 pin 16	ALT3
SRC	SRC_POR_B	POR_B	ALT0 (No Muxing)
	SRC_RESET_B	ONOFF	ALT0 (No Muxing)
	SRC_BOOT_MODE0	GPIO_AD_B0_04	ALT0
	SRC_BOOT_MODE1	GPIO_AD_B0_05	ALT0
	SRC_BT_CFG00	GPIO_B0_04	ALT6 SRC_BT_CFG pins are not used with Teensy because BT_FUSE_SEL is programmed
	SRC_BT_CFG01	GPIO_B0_05	ALT6
	SRC_BT_CFG02	GPIO_B0_06	ALT6
	SRC_BT_CFG03	GPIO_B0_07	ALT6
	SRC_BT_CFG04	GPIO_B0_08	ALT6
	SRC_BT_CFG05	GPIO_B0_09	ALT6
	SRC_BT_CFG06	GPIO_B0_10	ALT6
	SRC_BT_CFG07	GPIO_B0_11	ALT6
	SRC_BT_CFG08	GPIO_B0_12	ALT6
	SRC_BT_CFG09	GPIO_B0_13	ALT6
	SRC_BT_CFG10	GPIO_B0_14	ALT6
	SRC_BT_CFG11	GPIO_B0_15	ALT6
USB	USB_OTG1_PWR	GPIO_AD_B0_02 pin 1	ALT3
		GPIO_AD_B1_01 pin 18	ALT0
	USB_OTG1_OC	GPIO_AD_B0_03 pin 0	ALT3
		GPIO_AD_B1_03 pin 15	ALT0
	USB_OTG1_ID	GPIO_AD_B0_01	ALT3

Table continues on the next page...

Table 10-1. Muxing Options (continued)

		GPIO_AD_B1_02 pin 14	ALT0
	USB_OTG2_PWR	GPIO_EMC_41	ALT3
		GPIO_AD_B0_15	ALT0
	USB_OTG2_OC	GPIO_EMC_40	ALT3
		GPIO_AD_B0_14	ALT0
	USB_OTG2_ID	GPIO_AD_B0_00	ALT3
		GPIO_AD_B1_00 pin 19	ALT0
USB PHY 1	USB_PHY1_TSTI_TX_LS_MODE	GPIO_EMC_00	ALT6
	USB_PHY1_TSTI_TX_HS_MODE	GPIO_EMC_01	ALT6
	USB_PHY1_TSTI_TX_DN	GPIO_EMC_02	ALT6
	USB_PHY1_TSTO_RX_SQUELCH	GPIO_EMC_03	ALT6
	USB_PHY1_TSTO_RX_DISCON_DET	GPIO_EMC_04 pin 2	ALT6
	USB_PHY1_TSTO_RX_HS_RXD	GPIO_EMC_05 pin 3	ALT6
	USB_PHY1_TSTO_RX_FS_RXD	GPIO_EMC_07 pin 33	ALT6
	USB_PHY1_TSTI_TX_DP	GPIO_EMC_08 pin 5	ALT6
	USB_PHY1_TSTI_TX_EN	GPIO_EMC_09	ALT6
	USB_PHY1_TSTI_TX_HIZ	GPIO_EMC_10	ALT6
	USB_PHY1_TSTO_PLL_CLK20DIV	GPIO_EMC_12	ALT6
USB PHY 2	USB_PHY2_TSTO_RX_FS_RXD	GPIO_EMC_06 pin 4	ALT6
	USB_PHY2_TSTO_RX_HS_RXD	GPIO_EMC_11	ALT6
	USB_PHY2_TSTO_PLL_CLK20DIV	GPIO_EMC_13	ALT6
	USB_PHY2_TSTO_RX_SQUELCH	GPIO_EMC_14	ALT6
	USB_PHY2_TSTO_RX_DISCON_DET	GPIO_EMC_15	ALT6
USDHC1	USDHC1_CMD	GPIO_SD_B0_00	ALT0
	USDHC1_CLK	GPIO_SD_B0_01	ALT0
	USDHC1_DATA0	GPIO_SD_B0_02	ALT0
	USDHC1_DATA1	GPIO_SD_B0_03	ALT0
	USDHC1_DATA2	GPIO_SD_B0_04	ALT0
	USDHC1_DATA3	GPIO_SD_B0_05	ALT0
	USDHC1_RESET_B	GPIO_AD_B0_00	ALT6
		GPIO_B1_15	ALT6
		GPIO_EMC_33	ALT2

Table continues on the next page...

Table 10-1. Muxing Options (continued)

	USDHC1_VSELECT	GPIO_EMC_41	ALT6
		GPIO_AD_B1_01 pin 18	ALT6
		GPIO_B1_14	ALT6
		GPIO_EMC_34	ALT2
	USDHC1_WP	GPIO_EMC_36 pin 31	ALT6
		GPIO_AD_B1_00 pin 19	ALT6
		GPIO_B1_13 pin 34 (T4.1)	ALT6
		GPIO_EMC_12	ALT3
	USDHC1_CD_B	GPIO_AD_B1_02 pin 14	ALT6
		GPIO_EMC_35	ALT6
		GPIO_B1_12 pin 35 (T4.1)	ALT6
USDHC2	USDHC2_CMD	GPIO_AD_B1_08 pin 22	ALT6
		GPIO_SD_B1_05	ALT0
	USDHC2_CLK	GPIO_AD_B1_09 pin 23	ALT6
		GPIO_SD_B1_04	ALT0
	USDHC2_DATA0	GPIO_AD_B1_04 pin 40 (T4.1)	ALT6
		GPIO_SD_B1_03	ALT0
	USDHC2_DATA1	GPIO_AD_B1_05 pin 41 (T4.1)	ALT6
		GPIO_SD_B1_02	ALT0
	USDHC2_DATA2	GPIO_AD_B1_06 pin 17	ALT6
		GPIO_SD_B1_01	ALT0
	USDHC2_DATA3	GPIO_AD_B1_07 pin 16	ALT6
		GPIO_SD_B1_00	ALT0
	USDHC2_DATA4	GPIO_AD_B1_12 pin 38 (T4.1)	ALT6
		GPIO_SD_B1_08	ALT0
	USDHC2_DATA5	GPIO_AD_B1_13 pin 39 (T4.1)	ALT6
		GPIO_SD_B1_09	ALT0
	USDHC2_DATA6	GPIO_AD_B1_14 pin 26	ALT6
		GPIO_SD_B1_10	ALT0
	USDHC2_DATA7	GPIO_AD_B1_15 pin 27	ALT6
		GPIO_SD_B1_11	ALT0
	USDHC2_RESET_B	GPIO_AD_B1_11 pin 21	ALT6
		GPIO_EMC_40	ALT6
		GPIO_EMC_11	ALT3
		GPIO_SD_B1_06	ALT0
	USDHC2_WP	GPIO_AD_B1_10 pin 20	ALT6
		GPIO_EMC_37 pin 30	ALT6
	USDHC2_VSELECT	GPIO_EMC_38	ALT6
	USDHC2_CD_B	GPIO_AD_B1_03 pin 15	ALT6
		GPIO_EMC_39	ALT6

Table continues on the next page...

Table 10-1. Muxing Options (continued)

WDOG1	WDOG1_B	GPIO_B1_13 pin 34 (T4.1)	ALT0
		GPIO_AD_B1_10 pin 20	ALT1
		GPIO_EMC_39	ALT4
		GPIO_AD_B0_11	ALT3
		GPIO_AD_B1_00 pin 19	ALT4
	WDOG1_RESET_B_DEB	GPIO_AD_B0_15	ALT7
	WDOG1_ANY	GPIO_AD_B0_14	ALT7
WDOG2	WDOG2_B	GPIO_AD_B0_12 pin 24	ALT3
	WDOG2_RESET_B_DEB	GPIO_B0_03 pin 13	ALT6
XBAR1	XBAR_INOUT02	GPIO_EMC_00	ALT3
		GPIO_B1_14	ALT3
	XBAR_INOUT03	GPIO_EMC_01	ALT3
		GPIO_B1_15	ALT3
	XBAR_INOUT04	GPIO_SD_B0_00	ALT3
		GPIO_EMC_02	ALT3
	XBAR_INOUT05	GPIO_EMC_03	ALT3
		GPIO_SD_B0_01	ALT3
	XBAR_INOUT06	GPIO_SD_B0_02	ALT3
		GPIO_EMC_04 pin 2	ALT3
	XBAR_INOUT07	GPIO_EMC_05 pin 3	ALT3
		GPIO_SD_B0_03	ALT3
	XBAR_INOUT08	GPIO_SD_B0_04	ALT3
		GPIO_EMC_06 pin 4	ALT3
	XBAR_INOUT09	GPIO_EMC_07 pin 33	ALT3
		GPIO_SD_B0_05	ALT3
	XBAR_INOUT10	GPIO_B0_12 pin 32	ALT1
	XBAR_INOUT11	GPIO_B0_13	ALT1
	XBAR_INOUT12	GPIO_B0_14	ALT1
	XBAR_INOUT13	GPIO_B0_15	ALT1
	XBAR_INOUT14	GPIO_B1_00 pin 8	ALT1
		GPIO_AD_B0_00	ALT1
	XBAR_INOUT15	GPIO_AD_B0_01	ALT1
		GPIO_B1_01 pin 7	ALT1
	XBAR_INOUT16	GPIO_B1_02 pin 36 (T4.1)	ALT1
		GPIO_AD_B0_02 pin 1	ALT1
	XBAR_INOUT17	GPIO_AD_B0_03 pin 0	ALT1
		GPIO_EMC_08 pin 5	ALT3
		GPIO_AD_B0_05	ALT6
		GPIO_B1_03 pin 37 (T4.1)	ALT1
	XBAR_INOUT18	GPIO_EMC_35	ALT1

Table continues on the next page...

Table 10-1. Muxing Options (continued)

		GPIO_AD_B0_06	ALT6
	XBAR_INOUT19	GPIO_EMC_14	ALT1
		GPIO_AD_B0_07	ALT6
	XBAR_INOUT20	GPIO_AD_B0_08	ALT6
		GPIO_EMC_15	ALT1
	XBAR_INOUT21	GPIO_EMC_16	ALT1
		GPIO_AD_B0_09	ALT6
	XBAR_INOUT22	GPIO_AD_B0_10	ALT6
		GPIO_EMC_36 pin 31	ALT1
	XBAR_INOUT23	GPIO_EMC_37 pin 30	ALT1
		GPIO_AD_B0_11	ALT6
	XBAR_INOUT24	GPIO_AD_B0_14	ALT1
		GPIO_EMC_12	ALT1
	XBAR_INOUT25	GPIO_AD_B0_15	ALT1
		GPIO_EMC_13	ALT1
XTALOSC	REF_CLK_32K	GPIO_AD_B0_00	ALT2
	REF_CLK_24M	GPIO_AD_B0_01	ALT2
		GPIO_AD_B0_03 pin 0	ALT6
		GPIO_AD_B0_13 pin 25	ALT7
	XTALI	XTALI	No Muxing
	XTALO	XTALO	No Muxing

Chapter 11
IOMUX Controller (IOMUXC)

11.1 Overview

The IOMUX Controller (IOMUXC), together with the IOMUX, enables the IC to share one pad to several functional blocks. This sharing is done by multiplexing the pad's input and output signals.

Every module requires a specific pad setting (such as pull up or keeper), and for each pad, there are up to 8 muxing options (called ALT modes). The pad settings parameters are controlled by the IOMUXC.

The IOMUX consists only of combinatorial logic combined from several basic IOMUX cells. Each basic IOMUX cell handles only one pad signal's muxing.

Figure 11-1 illustrates the IOMUX/IOMUXC connectivity in the system.

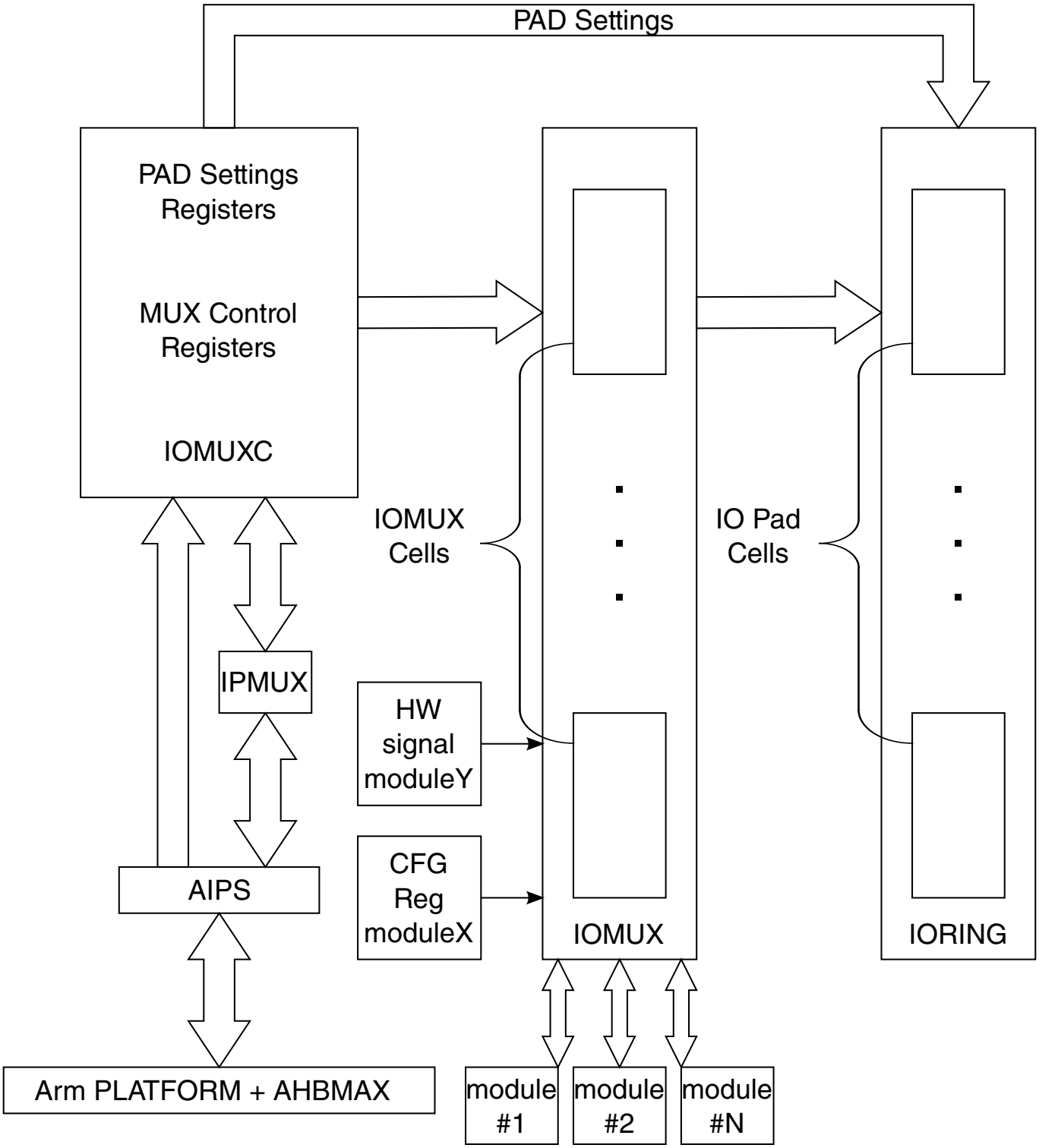


Figure 11-1. IOMUX SoC Level Block Diagram

11.1.1 Block Diagram

The high level illustration of the IO cells is shown in Figure 11-3

11.1.2 Features

The IOMUXC features include:

- 32-bit software mux control registers (IOMUXC_SW_MUX_CTL_PAD_<PAD NAME> or IOMUXC_SW_MUX_CTL_GRP_<GROUP NAME>) to configure 1 of 8 alternate (ALT) MUX_MODE fields of each pad or a predefined group of pads and to enable the forcing of an input path of the pad(s) (SION bit).
- 32-bit software pad control registers (IOMUXC_SW_PAD_CTL_PAD_<PAD_NAME> or IOMUXC_SW_PAD_CTL_GRP_<GROUP NAME>) to configure specific pad settings of each pad, or a predefined group of pads.
- 32-bit general purpose registers - several (GPR0 to GPR*n*) 32-bit registers according to SoC requirements for any usage.
- 32-bit input select control registers to control the input path to a module when more than one pad drives this module input.

Each SW MUX/PAD CTL IOMUXC register handles only one pad or one pad's group.

Only the minimum number of registers required by software are implemented by hardware. For example, if only ALT0 and ALT1 modes are used on Pad x then only one bit register will be generated as the MUX_MODE control field in the software mux control register of Pad x.

The software mux control registers may allow the forcing of pads to become input (input path enabled) regardless of the functional direction driven. This may be useful for loopback and GPIO data capture.

11.2 Functional description

This section provides a complete functional description of the block.

The IOMUXC consists of two sub-blocks:

- IOMUXC_REGISTERS includes all of the IOMUXC registers (see Features).
- IOMUXC_LOGIC includes all of the IOMUXC combinatorial logic (IP interface controls, address decoder, observability muxes).

The IOMUX consists of a number (about the number of pads in the SoC) of basic iomux_cell units. If only one functional mode is required for a specific pad, there is no need for IOMUX and the signals can be connected directly from the module to the I/O. The IOMUX cell is required whenever two or more functional modes are required for a specific pad or when one functional mode and the one test mode are required.

The basic iomux_cell design, which allows two levels of HW signal control (in ALT6 and ALT7 modes - ALT7 gets highest priority) is shown in Figure 11-2.

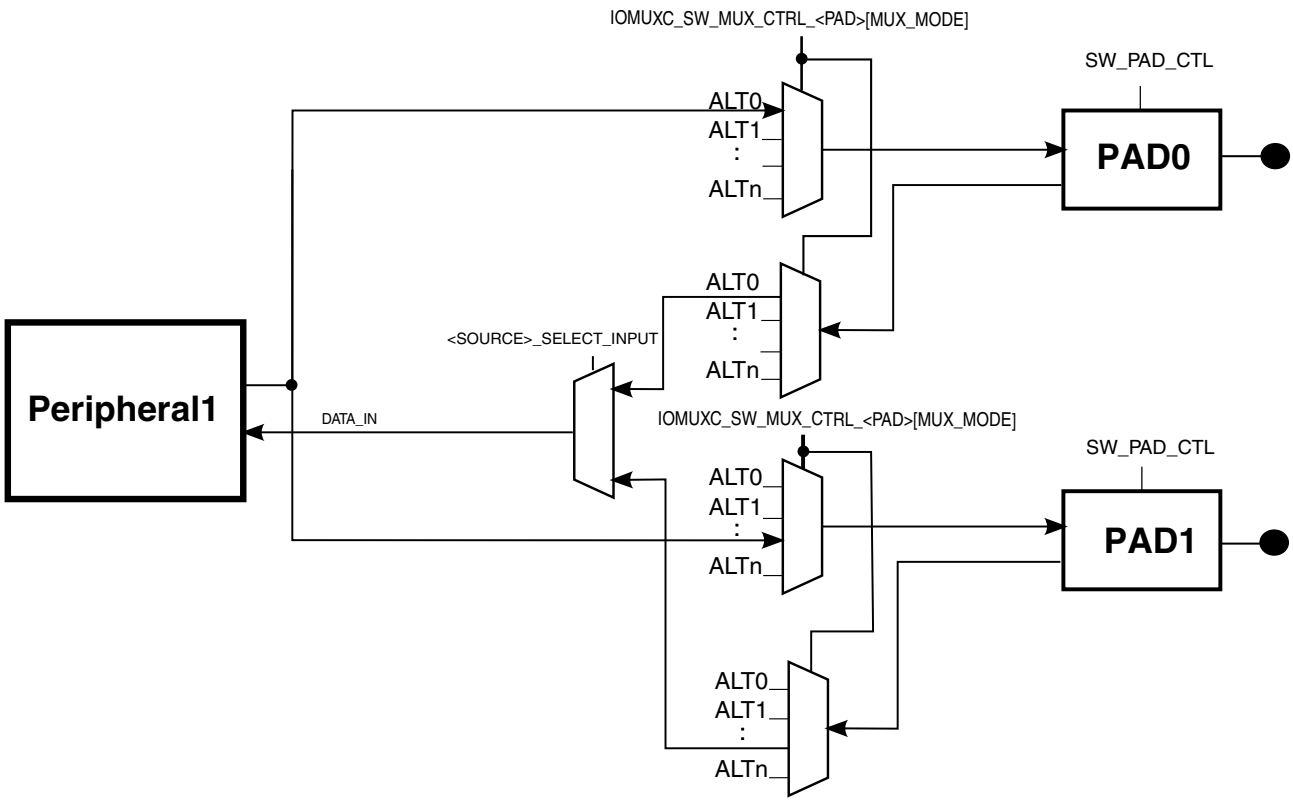


Figure 11-2. IOMUX Cell Block Diagram

11.2.1 ALT6 and ALT7 extended muxing modes

The ALT7 and ALT6 extended muxing modes allow any signal in the system (such as fuse, pad input, JTAG, or software register) to override any software configuration and to force the ALT6/ALT7 muxing mode.

It also allows an IOMUX software register to control a group of pads.

11.2.1 refers to an internal capability of the IOMUX hardware. You can't really make use of the "ALT6 and ALT7 extended muxing modes" from software.

11.2.2 SW Loopback through SION bit

A limited option exists to override the default pad functionality and force the input path to be active (ipp_ibe==1'b1) regardless of the value driven by the corresponding module. This can be done by setting the SION (Software Input On) bit in the IOMUXC_SW_MUX_CTL register (when available) to "1".

Uses include:

- LoopBack - Module x drives the pad and also receives pad value as an input.

11.2.3 Daisy chain - multi pads driving same module input pin

In some cases, more than one pad may drive a single module input pin. Such cases require the addition of one more level of IOMUXing; all of these input signals are muxed, and a dedicated software controlled register controls the mux in order to select the required input path.

A module port involved in "daisy chain" requires two software configuration commands, one for selecting the mode for this pad (programable via the IOMUXC_SW_MUX_CTL_<PAD> registers) and one for defining it as the input path (via the daisy chain registers).

This means that a module port involved in "daisy chain" requires two software configuration commands, one for selecting the mode for this pad (programable via the IOMUXC_SW_MUX_CTL_<PAD> registers) and one for defining it as the input path (via the daisy chain registers). The daisy chain is illustrated in the figure below.

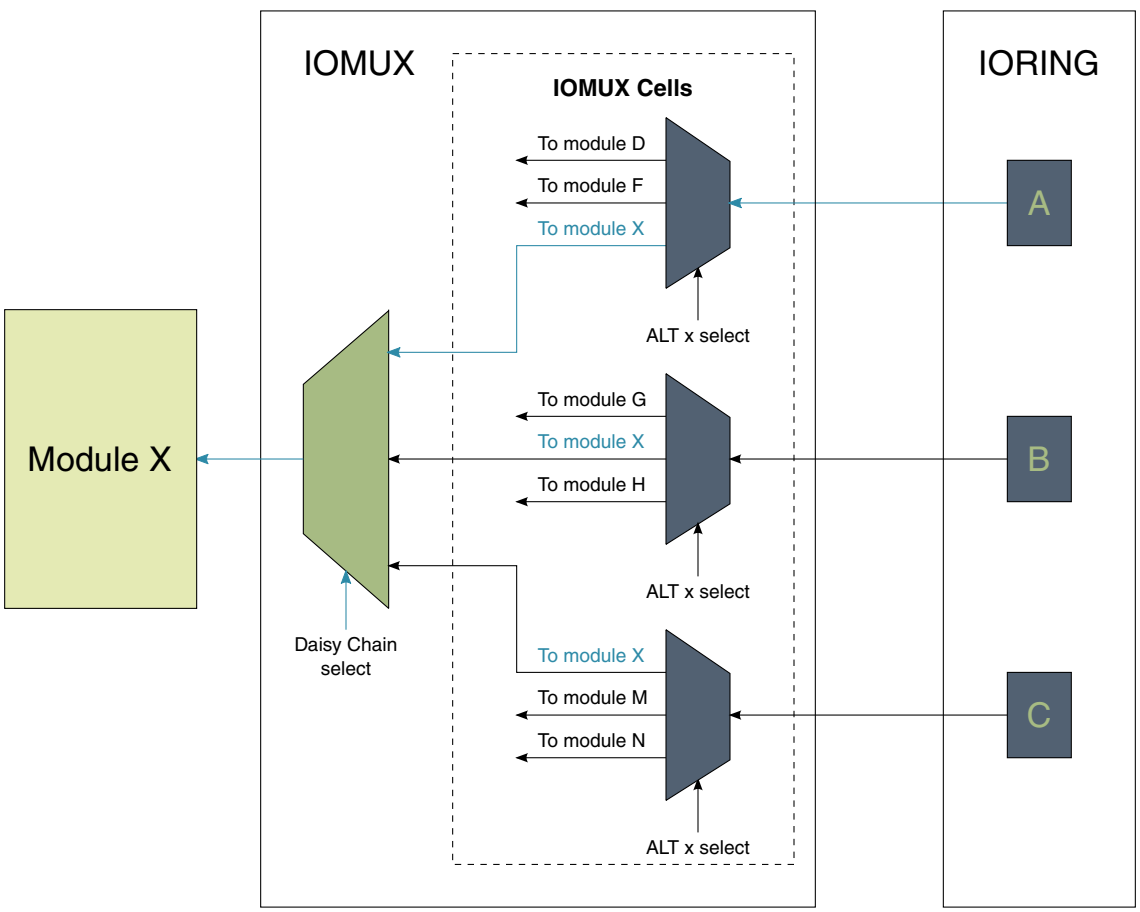


Figure 11-3. Daisy chain illustration

11.2.4 Clocks

The table found here describes the clock sources for IOMUXC.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 11-1. IOMUXC Clocks

Clock name	Clock Root	Description
ipg_clk_s	ipg_clk_root	Peripheral access clock

11.3 IOMUXC GPR Memory Map/Register Definition

IOMUXC_GPR memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400A_C000	GPR0 General Purpose Register (IOMUXC_GPR_GPR0)	32	R/W	0000_0000h	11.3.1/324
400A_C004	GPR1 General Purpose Register (IOMUXC_GPR_GPR1)	32	R/W	0000_0000h	11.3.2/325
400A_C008	GPR2 General Purpose Register (IOMUXC_GPR_GPR2)	32	R/W	0000_0000h	11.3.3/328
400A_C00C	GPR3 General Purpose Register (IOMUXC_GPR_GPR3)	32	R/W	0000_F0F0h	11.3.4/331
400A_C010	GPR4 General Purpose Register (IOMUXC_GPR_GPR4)	32	R/W	0000_0000h	11.3.5/335
400A_C014	GPR5 General Purpose Register (IOMUXC_GPR_GPR5)	32	R/W	0000_0000h	11.3.6/339
400A_C018	GPR6 General Purpose Register (IOMUXC_GPR_GPR6)	32	R/W	0000_0000h	11.3.7/341
400A_C01C	GPR7 General Purpose Register (IOMUXC_GPR_GPR7)	32	R/W	0000_0000h	11.3.8/345
400A_C020	GPR8 General Purpose Register (IOMUXC_GPR_GPR8)	32	R/W	0000_0000h	11.3.9/349
400A_C024	GPR9 General Purpose Register (IOMUXC_GPR_GPR9)	32	R/W	0000_0000h	11.3.10/352
400A_C028	GPR10 General Purpose Register (IOMUXC_GPR_GPR10)	32	R/W	0000_0007h	11.3.11/353
400A_C02C	GPR11 General Purpose Register (IOMUXC_GPR_GPR11)	32	R/W	0000_0000h	11.3.12/355
400A_C030	GPR12 General Purpose Register (IOMUXC_GPR_GPR12)	32	R/W	0000_0000h	11.3.13/356
400A_C034	GPR13 General Purpose Register (IOMUXC_GPR_GPR13)	32	R/W	0000_0000h	11.3.14/358
400A_C038	GPR14 General Purpose Register (IOMUXC_GPR_GPR14)	32	R/W	0000_0000h	11.3.15/360
400A_C03C	GPR15 General Purpose Register (IOMUXC_GPR_GPR15)	32	R/W	FFFF_FFFFh	11.3.16/361
400A_C040	GPR16 General Purpose Register (IOMUXC_GPR_GPR16)	32	R/W	0020_0003h	11.3.17/362
400A_C044	GPR17 General Purpose Register (IOMUXC_GPR_GPR17)	32	R/W	0000_0000h	11.3.18/363
400A_C048	GPR18 General Purpose Register (IOMUXC_GPR_GPR18)	32	R/W	0000_0000h	11.3.19/363
400A_C04C	GPR19 General Purpose Register (IOMUXC_GPR_GPR19)	32	R/W	0000_0000h	11.3.20/364
400A_C050	GPR20 General Purpose Register (IOMUXC_GPR_GPR20)	32	R/W	0000_0000h	11.3.21/365
400A_C054	GPR21 General Purpose Register (IOMUXC_GPR_GPR21)	32	R/W	0000_0000h	11.3.22/366
400A_C058	GPR22 General Purpose Register (IOMUXC_GPR_GPR22)	32	R/W	0000_0000h	11.3.23/367

Table continues on the next page...

IOMUXC_GPR memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400A_C05C	GPR23 General Purpose Register (IOMUXC_GPR_GPR23)	32	R/W	0000_0000h	11.3.24/368
400A_C060	GPR24 General Purpose Register (IOMUXC_GPR_GPR24)	32	R/W	0000_0000h	11.3.25/369
400A_C064	GPR25 General Purpose Register (IOMUXC_GPR_GPR25)	32	R/W	0000_0000h	11.3.26/370
400A_C068	GPR26 General Purpose Register (IOMUXC_GPR_GPR26)	32	R/W	0000_0000h	11.3.27/371
400A_C06C	GPR27 General Purpose Register (IOMUXC_GPR_GPR27)	32	R/W	0000_0000h	11.3.28/371
400A_C070	GPR28 General Purpose Register (IOMUXC_GPR_GPR28)	32	R/W	0000_0000h	11.3.29/372
400A_C074	GPR29 General Purpose Register (IOMUXC_GPR_GPR29)	32	R/W	0000_0000h	11.3.30/372
400A_C078	GPR30 General Purpose Register (IOMUXC_GPR_GPR30)	32	R/W	0000_0000h	11.3.31/373
400A_C07C	GPR31 General Purpose Register (IOMUXC_GPR_GPR31)	32	R/W	0000_0000h	11.3.32/373
400A_C080	GPR32 General Purpose Register (IOMUXC_GPR_GPR32)	32	R/W	0000_0000h	11.3.33/374
400A_C084	GPR33 General Purpose Register (IOMUXC_GPR_GPR33)	32	R/W	0000_0007h	11.3.34/374
400A_C088	GPR34 General Purpose Register (IOMUXC_GPR_GPR34)	32	R/W	0000_0000h	11.3.35/376

11.3.1 GPR0 General Purpose Register (IOMUXC_GPR_GPR0)

GPR Register

Address: 400A_C000h base + 0h offset = 400A_C000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_GPR_GPR0 field descriptions

Field	Description
-	Reserved Reserved. General purpose bits to be used by SoC integration. Bit field type: DEFAULT

11.3.2 GPR1 General Purpose Register (IOMUXC_GPR_GPR1)

GPR Register

Address: 400A_C000h base + 4h offset = 400A_C004h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CM7_FORCE_HCLK_EN	Reserved							ENET_IPG_CLK_S_EN	EXC_MON	SAI3_MCLK_DIR	SAI2_MCLK_DIR	SAI1_MCLK_DIR	ENET2_TX_CLK_DIR	ENET1_TX_CLK_DIR	Reserved
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved	ENET2_CLK_SEL	ENET1_CLK_SEL	GINT	SAI3_MCLK3_SEL		SAI2_MCLK3_SEL		SAI1_MCLK3_SEL		SAI1_MCLK2_SEL			SAI1_MCLK1_SEL		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_GPR_GPR1 field descriptions

Field	Description
31 CM7_FORCE_HCLK_EN	Arm CM7 platform AHB clock enable This bitfield determines whether or not the AHB clock is running when CM7 is sleeping. If the AHB clock is not enabled with this bit, the TCM is not accessible. 0 AHB clock is not running (gated) when CM7 is sleeping and TCM is not accessible. 1 AHB clock is running (enabled) when CM7 is sleeping and TCM is accessible.
30–24 -	This field is reserved. Reserved
23 ENET_IPG_CLK_S_EN	ENET and ENET2 ipg_clk_s clock gating enable 0 ipg_clk_s is gated when there is no IPS access 1 ipg_clk_s is always on
22 EXC_MON	Exclusive monitor response select of illegal command This bit sets the bus response value when CM7 exclusive access fails.

Table continues on the next page...

IOMUXC_GPR_GPR1 field descriptions (continued)

Field	Description
	0 OKAY response 1 SLVError response (default)
21 SAI3_MCLK_DIR	sai3.MCLK signal direction control This bitfield controls the sai3.MCLK signal direction. 0 sai3.MCLK is input signal 1 sai3.MCLK is output signal
20 SAI2_MCLK_DIR	sai2.MCLK signal direction control This bitfield controls the sai2.MCLK signal direction. 0 sai2.MCLK is input signal 1 sai2.MCLK is output signal
19 SAI1_MCLK_DIR	sai1.MCLK signal direction control This bitfield controls the sai1.MCLK signal direction. 0 sai1.MCLK is input signal 1 sai1.MCLK is output signal
18 ENET2_TX_ CLK_DIR	ENET2_TX_CLK data direction control 0 ENET2_TX_CLK output driver is disabled 1 ENET2_TX_CLK output driver is enabled
17 ENET1_TX_ CLK_DIR	ENET1_TX_CLK data direction control 0 ENET1_TX_CLK output driver is disabled 1 ENET1_TX_CLK output driver is enabled
16 -	This field is reserved. Reserved
15 -	This field is reserved. Reserved
14 ENET2_CLK_ SEL	ENET2 reference clock mode select. 0 ENET2 TX reference clock driven by ref_enetpll. This clock is also output to pins via the IOMUX. ENET2_REF_CLK function. 1 Gets ENET2 TX reference clock from the ENET2_TX_CLK pin. In this use case, an external OSC provides the clock for both the external PHY and the internal controller.
13 ENET1_CLK_ SEL	ENET1 reference clock mode select. 0 ENET1 TX reference clock driven by ref_enetpll. This clock is also output to pins via the IOMUX. ENET_REF_CLK1 function. 1 Gets ENET1 TX reference clock from the ENET1_TX_CLK pin. In this use case, an external OSC provides the clock for both the external PHY and the internal controller.
12 GINT	Global Interrupt This is the global interrupt "0" bit (connected to Arm M7 IRQ#0 and GPC) 0 Global interrupt request is not asserted. 1 Global interrupt request is asserted.

Table continues on the next page...

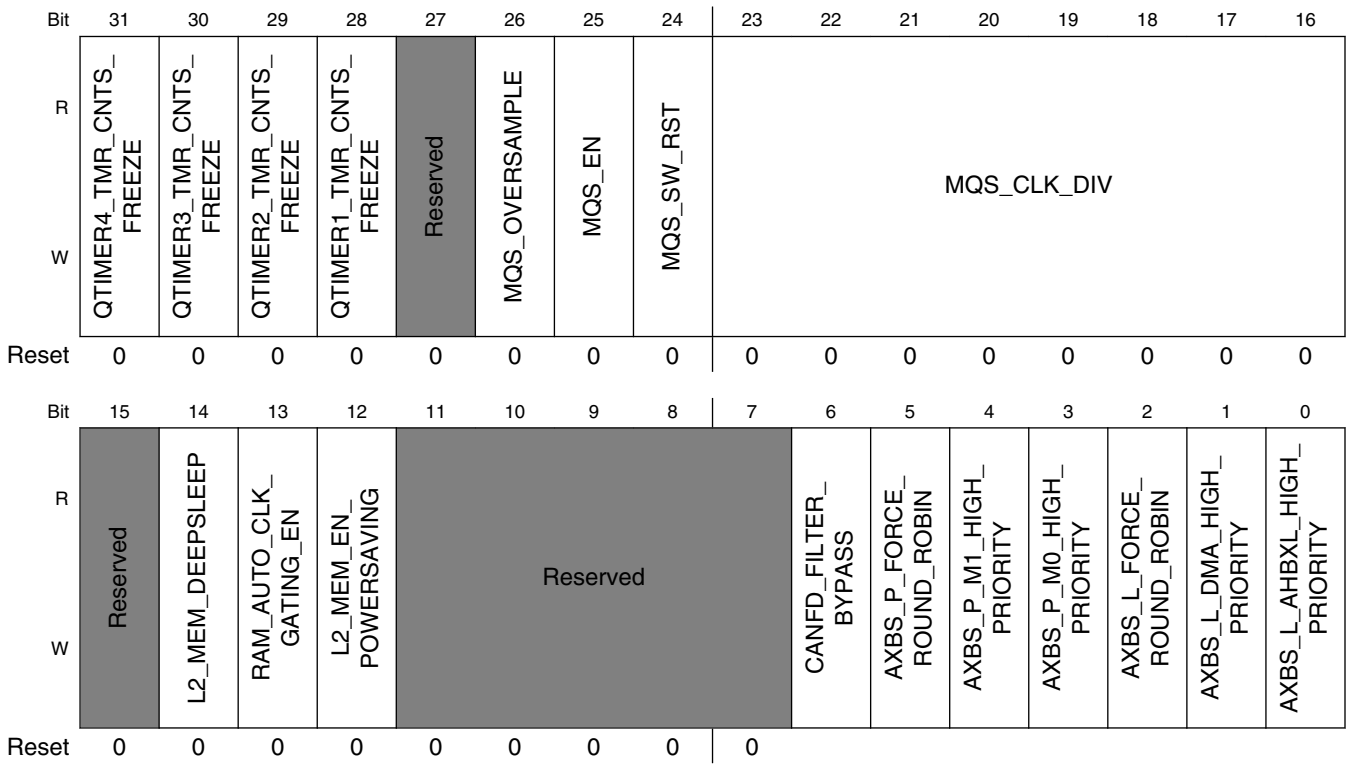
IOMUXC_GPR_GPR1 field descriptions (continued)

Field	Description
11–10 SAI3_MCLK3_SEL	SAI3 MCLK3 source select 00 ccm.spdif0_clk_root 01 iomux.spdif_tx_clk2 10 spdif.spdif_srclk 11 spdif.spdif_outclock
9–8 SAI2_MCLK3_SEL	SAI2 MCLK3 source select 00 ccm.spdif0_clk_root 01 iomux.spdif_tx_clk2 10 spdif.spdif_srclk 11 spdif.spdif_outclock
7–6 SAI1_MCLK3_SEL	SAI1 MCLK3 source select 00 ccm.spdif0_clk_root 01 iomux.spdif_tx_clk2 10 spdif.spdif_srclk 11 spdif.spdif_outclock
5–3 SAI1_MCLK2_SEL	SAI1 MCLK2 source select 000 ccm.ssi1_clk_root 001 ccm.ssi2_clk_root 010 ccm.ssi3_clk_root 011 iomux.sai1_ipg_clk_sai_mclk 100 iomux.sai2_ipg_clk_sai_mclk 101 iomux.sai3_ipg_clk_sai_mclk 110 Reserved 111 Reserved
SAI1_MCLK1_SEL	SAI1 MCLK1 source select 000 ccm.ssi1_clk_root 001 ccm.ssi2_clk_root 010 ccm.ssi3_clk_root 011 iomux.sai1_ipg_clk_sai_mclk 100 iomux.sai2_ipg_clk_sai_mclk 101 iomux.sai3_ipg_clk_sai_mclk 110 Reserved 111 Reserved

11.3.3 GPR2 General Purpose Register (IOMUXC_GPR_GPR2)

GPR Register

Address: 400A_C000h base + 8h offset = 400A_C008h



IOMUXC_GPR_GPR2 field descriptions

Field	Description
31 QTIMER4_TMR_CNTS_FREEZE	QTIMER4 timer counter freeze 0 timer counter work normally 1 reset counter and ouput flags
30 QTIMER3_TMR_CNTS_FREEZE	QTIMER3 timer counter freeze 0 timer counter work normally 1 reset counter and ouput flags
29 QTIMER2_TMR_CNTS_FREEZE	QTIMER2 timer counter freeze 0 timer counter work normally 1 reset counter and ouput flags
28 QTIMER1_TMR_CNTS_FREEZE	QTIMER1 timer counter freeze 0 timer counter work normally 1 reset counter and ouput flags

Table continues on the next page...

IOMUXC_GPR_GPR2 field descriptions (continued)

Field	Description
27 -	This field is reserved. Reserved
26 MQS_ OVERSAMPLE	Used to control the PWM oversampling rate compared with mclk. 0 32 1 64
25 MQS_EN	MQS enable. 0 Disable MQS 1 Enable MQS
24 MQS_SW_RST	MQS software reset 0 Exit software reset for MQS 1 Enable software reset for MQS
23–16 MQS_CLK_DIV	Divider ratio control for mclk from hmclk. mclk frequency = 1/(n+1) * hmclk frequency. 00000000 mclk frequency = hmclk frequency 00000001 mclk frequency = 1/2 * hmclk frequency 00000010 mclk frequency = 1/3 * hmclk frequency 11111111 mclk frequency = 1/256 * hmclk frequency
15 -	This field is reserved. Reserved
14 L2_MEM_ DEEPSLEEP	control how memory enter Deep Sleep mode (shutdown periphery power, but maintain memory contents, outputs of memory are pulled low) 0 no force sleep control supported, memory deep sleep mode only entered when whole system in stop mode 1 force memory into deep sleep mode
13 RAM_AUTO_ CLK_GATING_ EN	Automatically gate off RAM clock when RAM is not accessed. 0 disable automatically gate off RAM clock 1 enable automatically gate off RAM clock
12 L2_MEM_EN_ POWERSAVING	enable power saving features on L2 memory 0 none memory power saving features enabled, SHUTDOWN/DEEPSLEEP/LIGHTSLEEP will have no effect 1 memory power saving features enabled, set SHUTDOWN/DEEPSLEEP/LIGHTSLEEP (priority high to low) to enable power saving levels
11–7 -	This field is reserved. Reserved
6 CANFD_ FILTER_ BYPASS	Disable CANFD filter 0 enable CANFD filter 1 disable CANFD filter
5 AXBS_P_ FORCE_ ROUND_ROBIN	Force Round Robin in AXBS_P. This bit can override master M0 M1 high priority configuration. 0 AXBS_P masters are not arbitored in round robin, depending on M0/M1 master priority settings. 1 AXBS_P masters are arbitored in round robin

Table continues on the next page...

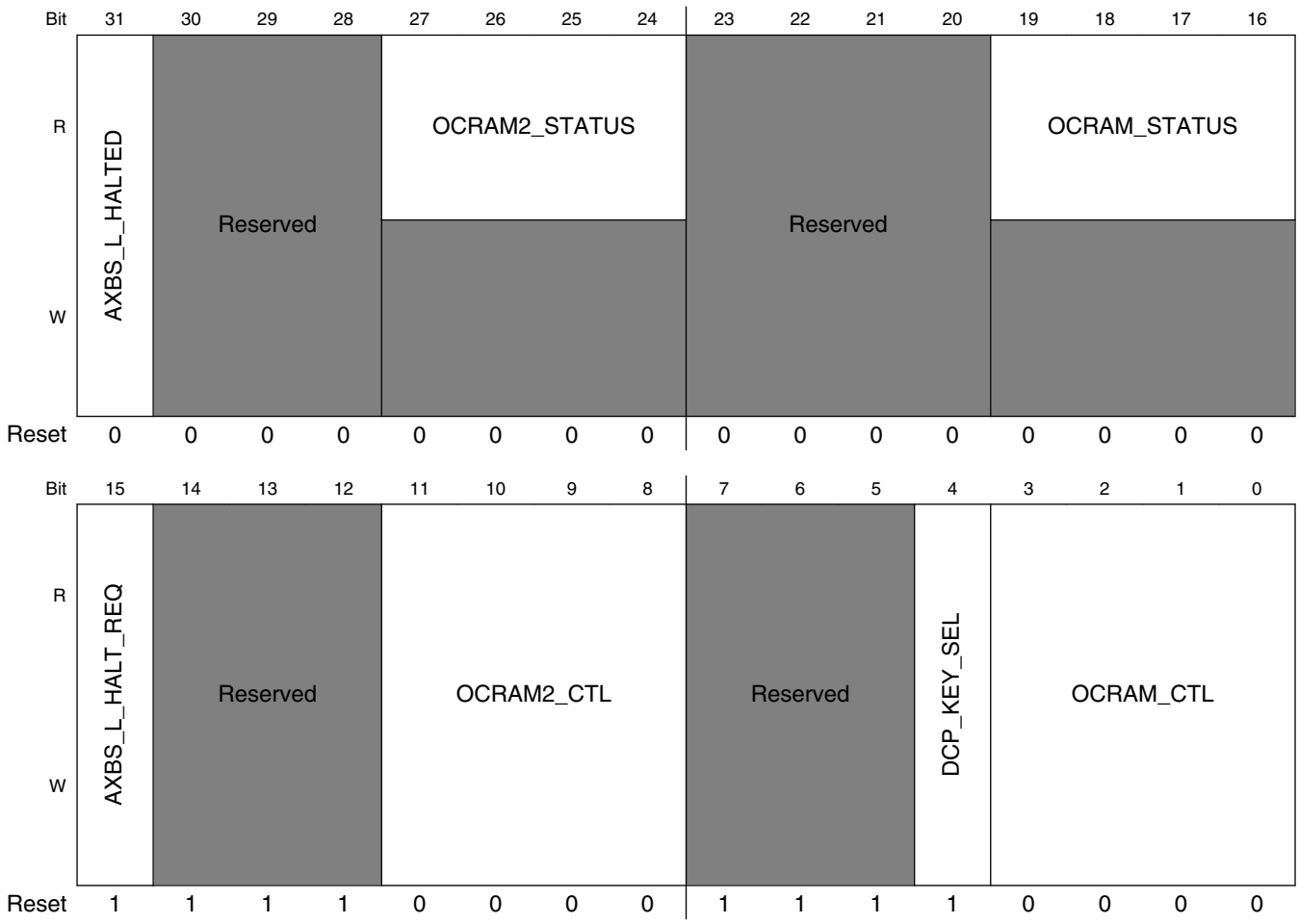
IOMUXC_GPR_GPR2 field descriptions (continued)

Field	Description
4 AXBS_P_M1_ HIGH_PRIORITY	AXBS_P M1 master has higher priority. **NOTE:** Do not set both M1 and M0 to high priority. 0 AXBS_P M1 master does not have high priority 1 AXBS_P M1 master has high priority
3 AXBS_P_M0_ HIGH_PRIORITY	AXBS_P M0 master has higher priority. **NOTE:** Do not set both M1 and M0 to high priority. 0 AXBS_P M0 master doesn't have high priority 1 AXBS_P M0 master has high priority
2 AXBS_L_ FORCE_ ROUND_ROBIN	Force Round Robin in AXBS_L. This bit can override master DMA and AHBXL high priority configuration. 0 AXBS_L masters are not arbitored in round robin, depending on DMA and AHBXL master priority settings. 1 AXBS_L masters are arbitored in round robin
1 AXBS_L_DMA_ HIGH_PRIORITY	AXBS_L DMA master has higher priority. **NOTE:** Do not set both DMA and AHBXL to high priority. 0 AXBS_L DMA master does not have high priority 1 AXBS_L DMA master has high priority
0 AXBS_L_ AHBXL_HIGH_ PRIORITY	AXBS_L AHBXL master has higher priority. **NOTE:** Do not set both DMA and AHBXL to high priority. 0 AXBS_L AHBXL master does not have high priority 1 AXBS_P AHBXL master has high priority

11.3.4 GPR3 General Purpose Register (IOMUXC_GPR_GPR3)

GPR Register

Address: 400A_C000h base + Ch offset = 400A_C00Ch



IOMUXC_GPR_GPR3 field descriptions

Field	Description
31 AXBS_L_ HALTED	This bit shows the status of axbs_l 0 axbs_l is not halted 1 axbs_l is in halted status
30–28 -	This field is reserved. Reserved
27–24 OCRAM2_ STATUS	This field shows the OCRAM2 pipeline settings status, controlled by OCRAM2_CTL bits respectively. When the control bit is changed, the corresponding status bit goes high and keeps high until this new configuration is applied the internal logic. This provides a way for software to detect that the configuration has become valid. The suggested flow for changing the configuration in software is:

Table continues on the next page...

IOMUXC_GPR_GPR3 field descriptions (continued)

Field	Description
	• set/clear the control bit • poll the status bit until it goes to 0 OCRAM2_STATUS[3] shows the write address pipeline status. This bit value reflects the propagation of the respective control bit to OCRAM2 memory. OCRAM2_STATUS[3] shows the write data pipeline status. This bit value reflects the propagation of the respective control bit to OCRAM2 memory. OCRAM2_STATUS[1] shows the read address pipeline status. This bit value reflects the propagation of the respective control bit to OCRAM2 memory. OCRAM2_STATUS[0] shows the read data wait status. This bit value reflects the propagation of the respective control bit to OCRAM2 memory. • 0: read data pipeline configuration valid • 1: read data pipeline control bit changed
23–20 -	This field is reserved. Reserved
19–16 OCRAM_ STATUS	This field shows the OCRAM pipeline settings status, controlled by OCRAM_CTL bits respectively. When the control bit is changed, the corresponding status bit goes high and keeps high until this new configuration is applied the internal logic. This provides a way for software to detect that the configuration has become valid. The suggested flow for changing the configuration in software is: • set/clear the control bit • poll the status bit until it goes to 0 OCRAM_STATUS[19] shows the write address pipeline status. This bit value reflects the propagation of the respective control bit to OCRAM memory. OCRAM_STATUS[18] shows the write data pipeline status. This bit value reflects the propagation of the respective control bit to OCRAM memory. OCRAM_STATUS[17] shows the read address pipeline status. This bit value reflects the propagation of the respective control bit to OCRAM memory. OCRAM_STATUS[16] shows the read data wait status. This bit value reflects the propagation of the respective control bit to OCRAM memory. • 0: read data pipeline configuration valid • 1: read data pipeline control bit changed
15 AXBS_L_HALT_ REQ	Request to halt axbs_l 0 axbs_l normal run 1 request to halt axbs_l
14–12 -	This field is reserved. Reserved
11–8 OCRAM2_CTL	OCRAM2_CTL[3] - write address pipeline control bit. When this feature is enabled, the write address from the AXI master would be delayed 1 cycle before it can be accepted by the on-chip RAM. This can avoid setup time issue for the write access on the memory cell at high frequency. Enabling this feature would cost at most 1 more clock cycle for each AXI write transaction, i.e., at most 1 more clock cycle for each write burst with multiple beats of data. When this feature is disabled, the write address from the AXI master can be accepted by the on-chip RAM without delay, and data can be written to memory at this cycle (if no other access and write data is also ready at this cycle). • 0: write address pipeline is disabled • 1: write address pipeline is enabled

Table continues on the next page...

IOMUXC_GPR_GPR3 field descriptions (continued)

Field	Description
	OCRAM2_CTL[2] - write data pipeline control bit When this feature is enabled, the write data from the AXI master would be delayed 1 cycle before it can be accepted by the on-chip RAM. This can avoid setup time issue for the write access on the memory cell at high frequency. Enabling this feature would cost at most 1 more clock cycle for each AXI write transaction, i.e., at most 1 more clock cycle for each write burst with multiple beats of data. When this feature is disabled, the write data from the AXI master can be accepted by the on-chip RAM without delay, and data can be written to memory at this cycle (if no other access and write address is also ready at this cycle). • 0: write data pipeline is disabled • 1: write data pipeline is enabled OCRAM2_CTL[1] - read address pipeline control bit. When this feature is enabled, the read address from the AXI master would be delayed 1 cycle before it can be accepted by the on-chip RAM. This can avoid setup time issue for the read access on the memory cell at high frequency. Enabling this feature would cost at most 1 more clock cycle for each AXI read transaction, i.e., at most 1 more clock cycle for each read burst with multiple beats of data. When this feature is disabled, the read address from the AXI master can be accepted by the on-chip RAM without delay, and data can become ready for master at next clock cycle (if no other access and no read data wait). • 0: read address pipeline is disabled • 1: read address pipeline is enabled OCRAM2_CTL[0] - read data wait state control bit When thread data wait state is enabled, it will cost 2 cycles for each read access, (each beat of a read burst). This can avoid the potential timing problem caused by the relatively longer memory access time at higher frequency. When this feature is disabled, it only costs 1 clock cycle to finish a read transaction, i.e., get read data back in the next cycle of read request becomes valid on the bus. • 0: read data pipeline is disabled • 1: read data pipeline is enabled
7–5 -	This field is reserved. Reserved
4 DCP_KEY_SEL	Select 128-bit dcp key from 256-bit key from SNVS Master Key 0 Select [127:0] from SNVS Master Key as dcp key 1 Select [255:128] from SNVS Master Key as dcp key
OCRAM_CTL	OCRAM_CTL[3] - write address pipeline control bit. When this feature is enabled, the write address from the AXI master would be delayed 1 cycle before it can be accepted by the on-chip RAM. This can avoid setup time issue for the write access on the memory cell at high frequency. Enabling this feature would cost at most 1 more clock cycle for each AXI write transaction, i.e., at most 1 more clock cycle for each write burst with multiple beats of data. When this feature is disabled, the write address from the AXI master can be accepted by the on-chip RAM without delay, and data can be written to memory at this cycle (if no other access and write data is also ready at this cycle). • 0: write address pipeline is disabled • 1: write address pipeline is enabled OCRAM_CTL[2] - write data pipeline control bit

Table continues on the next page...

IOMUXC_GPR_GPR3 field descriptions (continued)

Field	Description
	When this feature is enabled, the write data from the AXI master would be delayed 1 cycle before it can be accepted by the on-chip RAM. This can avoid setup time issue for the write access on the memory cell at high frequency. Enabling this feature would cost at most 1 more clock cycle for each AXI write transaction, i.e., at most 1 more clock cycle for each write burst with multiple beats of data. When this feature is disabled, the write data from the AXI master can be accepted by the on-chip RAM without delay, and data can be written to memory at this cycle (if no other access and write address is also ready at this cycle). • 0: write data pipeline is disabled • 1: write data pipeline is enabled OCRAM_CTL[1] - read address pipeline control bit. When this feature is enabled, the read address from the AXI master would be delayed 1 cycle before it can be accepted by the on-chip RAM. This can avoid setup time issue for the read access on the memory cell at high frequency. Enabling this feature would cost at most 1 more clock cycle for each AXI read transaction, i.e., at most 1 more clock cycle for each read burst with multiple beats of data. When this feature is disabled, the read address from the AXI master can be accepted by the on-chip RAM without delay, and data can become ready for master at next clock cycle (if no other access and no read data wait). • 0: read address pipeline is disabled • 1: read address pipeline is enabled OCRAM_CTL[0] - read data wait state control bit When thread data wait state is enabled, it will cost 2 cycles for each read access, (each beat of a read burst). This can avoid the potential timing problem caused by the relatively longer memory access time at higher frequency. When this feature is disabled, it only costs 1 clock cycle to finish a read transaction, i.e., get read data back in the next cycle of read request becomes valid on the bus. • 0: read data pipeline is disabled • 1: read data pipeline is enabled

11.3.5 GPR4 General Purpose Register (IOMUXC_GPR_GPR4)

GPR Register

Address: 400A_C000h base + 10h offset = 400A_C010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	FLEXSPI2_STOP_ACK	FLEXIO3_STOP_ACK	FLEXIO2_STOP_ACK	FLEXIO1_STOP_ACK	FLEXSPI_STOP_ACK	PIT_STOP_ACK	SEMC_STOP_ACK	ENET2_STOP_ACK	SAI3_STOP_ACK	SAI2_STOP_ACK	SAI1_STOP_ACK	ENET_STOP_ACK	TRNG_STOP_ACK	CAN2_STOP_ACK	CAN1_STOP_ACK	EDMA_STOP_ACK
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FLEXSPI2_STOP_REQ	FLEXIO3_STOP_REQ	FLEXIO2_STOP_REQ	FLEXIO1_STOP_REQ	FLEXSPI_STOP_REQ	PIT_STOP_REQ	SEMC_STOP_REQ	ENET2_STOP_REQ	SAI3_STOP_REQ	SAI2_STOP_REQ	SAI1_STOP_REQ	ENET_STOP_REQ	TRNG_STOP_REQ	CAN2_STOP_REQ	CAN1_STOP_REQ	EDMA_STOP_REQ
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_GPR_GPR4 field descriptions

Field	Description
31 FLEXSPI2_ STOP_ACK	FLEXSPI2 stop acknowledge 0 FLEXSPI2 stop acknowledge is not asserted 1 FLEXSPI2 stop acknowledge is asserted
30 FLEXIO3_STOP_ ACK	On-platform FLEXIO3 stop acknowledge 0 FLEXIO3 stop acknowledge is not asserted 1 FLEXIO3 stop acknowledge is asserted
29 FLEXIO2_STOP_ ACK	FLEXIO2 stop acknowledge 0 FLEXIO2 stop acknowledge is not asserted 1 FLEXIO2 stop acknowledge is asserted (FLEXIO2 is in STOP mode)
28 FLEXIO1_STOP_ ACK	FLEXIO1 stop acknowledge 0 FLEXIO1 stop acknowledge is not asserted 1 FLEXIO1 stop acknowledge is asserted
27 FLEXSPI_STOP_ ACK	FLEXSPI stop acknowledge 0 FLEXSPI stop acknowledge is not asserted 1 FLEXSPI stop acknowledge is asserted
26 PIT_STOP_ACK	PIT stop acknowledge 0 PIT stop acknowledge is not asserted 1 PIT stop acknowledge is asserted
25 SEMC_STOP_ ACK	SEMC stop acknowledge 0 SEMC stop acknowledge is not asserted 1 SEMC stop acknowledge is asserted
24 ENET2_STOP_ ACK	ENET2 stop acknowledge. 0 ENET2 stop acknowledge is not asserted 1 ENET2 stop acknowledge is asserted
23 SAI3_STOP_ ACK	SAI3 stop acknowledge 0 SAI3 stop acknowledge is not asserted 1 SAI3 stop acknowledge is asserted
22 SAI2_STOP_ ACK	SAI2 stop acknowledge 0 SAI2 stop acknowledge is not asserted 1 SAI2 stop acknowledge is asserted
21 SAI1_STOP_ ACK	SAI1 stop acknowledge 0 SAI1 stop acknowledge is not asserted 1 SAI1 stop acknowledge is asserted
20 ENET_STOP_ ACK	ENET stop acknowledge. 0 ENET1 stop acknowledge is not asserted 1 ENET1 stop acknowledge is asserted

Table continues on the next page...

IOMUXC_GPR_GPR4 field descriptions (continued)

Field	Description
19 TRNG_STOP_ACK	TRNG stop acknowledge 0 TRNG stop acknowledge is not asserted 1 TRNG stop acknowledge is asserted
18 CAN2_STOP_ACK	CAN2 stop acknowledge. 0 CAN2 stop acknowledge is not asserted 1 CAN2 stop acknowledge is asserted
17 CAN1_STOP_ACK	CAN1 stop acknowledge. 0 CAN1 stop acknowledge is not asserted 1 CAN1 stop acknowledge is asserted
16 EDMA_STOP_ACK	EDMA stop acknowledge. This is a status (read-only) bit 0 EDMA stop acknowledge is not asserted 1 EDMA stop acknowledge is asserted (EDMA is in STOP mode).
15 FLEXSPI2_STOP_REQ	FlexSPI2 stop request. 0 stop request off 1 stop request on
14 FLEXIO3_STOP_REQ	On-platform flexio3 stop request. 0 stop request off 1 stop request on
13 FLEXIO2_STOP_REQ	FlexIO2 stop request. 0 stop request off 1 stop request on
12 FLEXIO1_STOP_REQ	FlexIO1 stop request. 0 stop request off 1 stop request on
11 FLEXSPI_STOP_REQ	FlexSPI stop request. 0 stop request off 1 stop request on
10 PIT_STOP_REQ	PIT stop request. 0 stop request off 1 stop request on
9 SEMC_STOP_REQ	SEMC stop request. 0 stop request off 1 stop request on
8 ENET2_STOP_REQ	ENET2 stop request. 0 stop request off 1 stop request on

Table continues on the next page...

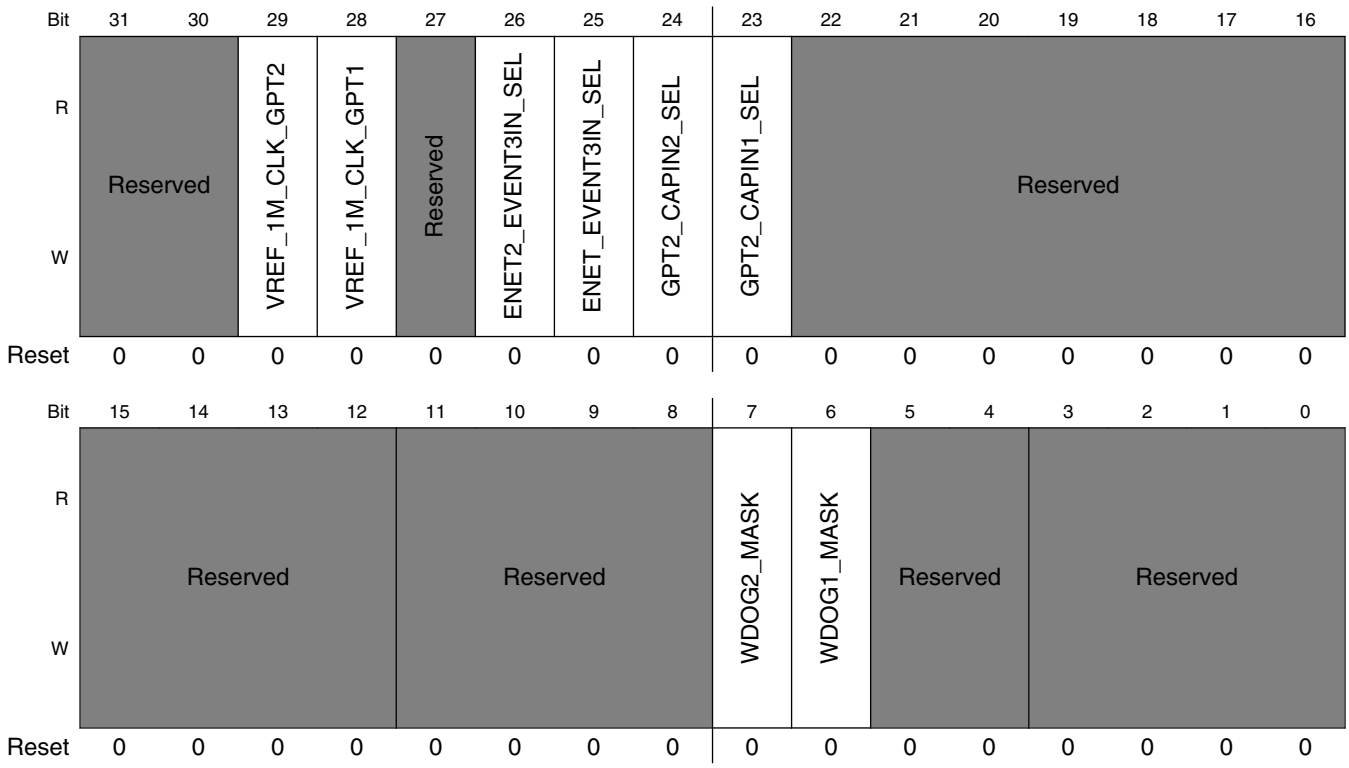
IOMUXC_GPR_GPR4 field descriptions (continued)

Field	Description
7 SAI3_STOP_ REQ	SAI3 stop request. 0 stop request off 1 stop request on
6 SAI2_STOP_ REQ	SAI2 stop request. 0 stop request off 1 stop request on
5 SAI1_STOP_ REQ	SAI1 stop request. 0 stop request off 1 stop request on
4 ENET_STOP_ REQ	ENET stop request. 0 stop request off 1 stop request on
3 TRNG_STOP_ REQ	TRNG stop request. 0 stop request off 1 stop request on
2 CAN2_STOP_ REQ	CAN2 stop request. 0 stop request off 1 stop request on
1 CAN1_STOP_ REQ	CAN1 stop request. 0 stop request off 1 stop request on
0 EDMA_STOP_ REQ	EDMA stop request. 0 stop request off 1 stop request on

11.3.6 GPR5 General Purpose Register (IOMUXC_GPR_GPR5)

GPR Register

Address: 400A_C000h base + 14h offset = 400A_C014h



IOMUXC_GPR_GPR5 field descriptions

Field	Description
31–30 -	This field is reserved. Reserved
29 VREF_1M_CLK_GPT2	GPT2 1 MHz clock source select 0 GPT2 ipg_clk_highfreq driven by IPG_PERCLK 1 GPT2 ipg_clk_highfreq driven by anatop 1 MHz clock
28 VREF_1M_CLK_GPT1	GPT1 1 MHz clock source select 0 GPT1 ipg_clk_highfreq driven by IPG_PERCLK 1 GPT1 ipg_clk_highfreq driven by anatop 1 MHz clock
27 -	This field is reserved. Reserved
26 ENET2_EVENT3IN_SEL	ENET2 input timer event3 source select 0 event3 source input from ENET2_1588_EVENT3_IN 1 event3 source input from GPT2.GPT_COMPARE2

Table continues on the next page...

IOMUXC_GPR_GPR5 field descriptions (continued)

Field	Description
25 ENET_ EVENT3IN_SEL	ENET input timer event3 source select 0 event3 source input from ENET_1588_EVENT3_IN 1 event3 source input from GPT2.GPT_COMPARE1
24 GPT2_CAPIN2_ SEL	GPT2 input capture channel 2 source select 0 source from GPT2_CAPTURE2 1 source from ENET2_1588_EVENT3_OUT (chnnal 3 of IEEE 1588 timer)
23 GPT2_CAPIN1_ SEL	GPT2 input capture channel 1 source select 0 source from GPT2_CAPTURE1 1 source from ENET_1588_EVENT3_OUT (chnnal 3 of IEEE 1588 timer)
22–12 -	This field is reserved. Reserved
11–8 -	This field is reserved. Reserved
7 WDOG2_MASK	WDOG2 Timeout Mask 0 WDOG2 Timeout behaves normally 1 WDOG2 Timeout is masked
6 WDOG1_MASK	WDOG1 Timeout Mask 0 WDOG1 Timeout behaves normally 1 WDOG1 Timeout is masked
5–4 -	This field is reserved. Reserved
-	This field is reserved. Reserved

11.3.7 GPR6 General Purpose Register (IOMUXC_GPR_GPR6)

GPR Register

Address: 400A_C000h base + 18h offset = 400A_C018h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	IOMUXC_XBAR_DIR_SEL_19	IOMUXC_XBAR_DIR_SEL_18	IOMUXC_XBAR_DIR_SEL_17	IOMUXC_XBAR_DIR_SEL_16	IOMUXC_XBAR_DIR_SEL_15	IOMUXC_XBAR_DIR_SEL_14	IOMUXC_XBAR_DIR_SEL_13	IOMUXC_XBAR_DIR_SEL_12	IOMUXC_XBAR_DIR_SEL_11	IOMUXC_XBAR_DIR_SEL_10	IOMUXC_XBAR_DIR_SEL_9	IOMUXC_XBAR_DIR_SEL_8	IOMUXC_XBAR_DIR_SEL_7	IOMUXC_XBAR_DIR_SEL_6	IOMUXC_XBAR_DIR_SEL_5	IOMUXC_XBAR_DIR_SEL_4
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	QTIMER4_TRM3_INPUT_SEL	QTIMER4_TRM2_INPUT_SEL	QTIMER4_TRM1_INPUT_SEL	QTIMER4_TRM0_INPUT_SEL	QTIMER3_TRM3_INPUT_SEL	QTIMER3_TRM2_INPUT_SEL	QTIMER3_TRM1_INPUT_SEL	QTIMER3_TRM0_INPUT_SEL	QTIMER2_TRM3_INPUT_SEL	QTIMER2_TRM2_INPUT_SEL	QTIMER2_TRM1_INPUT_SEL	QTIMER2_TRM0_INPUT_SEL	QTIMER1_TRM3_INPUT_SEL	QTIMER1_TRM2_INPUT_SEL	QTIMER1_TRM1_INPUT_SEL	QTIMER1_TRM0_INPUT_SEL
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_GPR_GPR6 field descriptions

Field	Description
31 IOMUXC_XBAR_DIR_SEL_19	IOMUXC XBAR_INOUT19 function direction select 0 XBAR_INOUT as input 1 XBAR_INOUT as output
30 IOMUXC_XBAR_DIR_SEL_18	IOMUXC XBAR_INOUT18 function direction select 0 XBAR_INOUT as input 1 XBAR_INOUT as output
29 IOMUXC_XBAR_DIR_SEL_17	IOMUXC XBAR_INOUT17 function direction select 0 XBAR_INOUT as input 1 XBAR_INOUT as output
28 IOMUXC_XBAR_DIR_SEL_16	IOMUXC XBAR_INOUT16 function direction select 0 XBAR_INOUT as input 1 XBAR_INOUT as output
27 IOMUXC_XBAR_DIR_SEL_15	IOMUXC XBAR_INOUT15 function direction select 0 XBAR_INOUT as input 1 XBAR_INOUT as output

Table continues on the next page...

IOMUXC_GPR_GPR6 field descriptions (continued)

Field	Description
26 IOMUXC_XBAR_DIR_SEL_14	IOMUXC XBAR_INOUT14 function direction select 0 XBAR_INOUT as input 1 XBAR_INOUT as output
25 IOMUXC_XBAR_DIR_SEL_13	IOMUXC XBAR_INOUT13 function direction select 0 XBAR_INOUT as input 1 XBAR_INOUT as output
24 IOMUXC_XBAR_DIR_SEL_12	IOMUXC XBAR_INOUT12 function direction select 0 XBAR_INOUT as input 1 XBAR_INOUT as output
23 IOMUXC_XBAR_DIR_SEL_11	IOMUXC XBAR_INOUT11 function direction select 0 XBAR_INOUT as input 1 XBAR_INOUT as output
22 IOMUXC_XBAR_DIR_SEL_10	IOMUXC XBAR_INOUT10 function direction select 0 XBAR_INOUT as input 1 XBAR_INOUT as output
21 IOMUXC_XBAR_DIR_SEL_9	IOMUXC XBAR_INOUT9 function direction select 0 XBAR_INOUT as input 1 XBAR_INOUT as output
20 IOMUXC_XBAR_DIR_SEL_8	IOMUXC XBAR_INOUT8 function direction select 0 XBAR_INOUT as input 1 XBAR_INOUT as output
19 IOMUXC_XBAR_DIR_SEL_7	IOMUXC XBAR_INOUT7 function direction select 0 XBAR_INOUT as input 1 XBAR_INOUT as output
18 IOMUXC_XBAR_DIR_SEL_6	IOMUXC XBAR_INOUT6 function direction select 0 XBAR_INOUT as input 1 XBAR_INOUT as output
17 IOMUXC_XBAR_DIR_SEL_5	IOMUXC XBAR_INOUT5 function direction select 0 XBAR_INOUT as input 1 XBAR_INOUT as output
16 IOMUXC_XBAR_DIR_SEL_4	IOMUXC XBAR_INOUT4 function direction select 0 XBAR_INOUT as input 1 XBAR_INOUT as output
15 QTIMER4_TRM3_INPUT_SEL	QTIMER4 TMR3 input select 0 input from IOMUX 1 input from XBAR

Table continues on the next page...

IOMUXC_GPR_GPR6 field descriptions (continued)

Field	Description
14 QTIMER4_ TRM2_INPUT_ SEL	QTIMER4 TMR2 input select 0 input from IOMUX 1 input from XBAR
13 QTIMER4_ TRM1_INPUT_ SEL	QTIMER4 TMR1 input select 0 input from IOMUX 1 input from XBAR
12 QTIMER4_ TRM0_INPUT_ SEL	QTIMER4 TMR0 input select 0 input from IOMUX 1 input from XBAR
11 QTIMER3_ TRM3_INPUT_ SEL	QTIMER3 TMR3 input select 0 input from IOMUX 1 input from XBAR
10 QTIMER3_ TRM2_INPUT_ SEL	QTIMER3 TMR2 input select 0 input from IOMUX 1 input from XBAR
9 QTIMER3_ TRM1_INPUT_ SEL	QTIMER3 TMR1 input select 0 input from IOMUX 1 input from XBAR
8 QTIMER3_ TRM0_INPUT_ SEL	QTIMER3 TMR0 input select 0 input from IOMUX 1 input from XBAR
7 QTIMER2_ TRM3_INPUT_ SEL	QTIMER2 TMR3 input select 0 input from IOMUX 1 input from XBAR
6 QTIMER2_ TRM2_INPUT_ SEL	QTIMER2 TMR2 input select 0 input from IOMUX 1 input from XBAR
5 QTIMER2_ TRM1_INPUT_ SEL	QTIMER2 TMR1 input select 0 input from IOMUX 1 input from XBAR
4 QTIMER2_ TRM0_INPUT_ SEL	QTIMER2 TMR0 input select 0 input from IOMUX 1 input from XBAR
3 QTIMER1_ TRM3_INPUT_ SEL	QTIMER1 TMR3 input select 0 input from IOMUX 1 input from XBAR

Table continues on the next page...

IOMUXC_GPR_GPR6 field descriptions (continued)

Field	Description
2 QTIMER1_ TRM2_INPUT_ SEL	QTIMER1 TMR2 input select 0 input from IOMUX 1 input from XBAR
1 QTIMER1_ TRM1_INPUT_ SEL	QTIMER1 TMR1 input select 0 input from IOMUX 1 input from XBAR
0 QTIMER1_ TRM0_INPUT_ SEL	QTIMER1 TMR0 input select 0 input from IOMUX 1 input from XBAR

11.3.8 GPR7 General Purpose Register (IOMUXC_GPR_GPR7)

GPR Register

Address: 400A_C000h base + 1Ch offset = 400A_C01Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	LPUART8_STOP_ACK	LPUART7_STOP_ACK	LPUART6_STOP_ACK	LPUART5_STOP_ACK	LPUART4_STOP_ACK	LPUART3_STOP_ACK	LPUART2_STOP_ACK	LPUART1_STOP_ACK	LPSPI4_STOP_ACK	LPSPI3_STOP_ACK	LPSPI2_STOP_ACK	LPSPI1_STOP_ACK	LPI2C4_STOP_ACK	LPI2C3_STOP_ACK	LPI2C2_STOP_ACK	LPI2C1_STOP_ACK
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LPUART8_STOP_REQ	LPUART7_STOP_REQ	LPUART6_STOP_REQ	LPUART5_STOP_REQ	LPUART4_STOP_REQ	LPUART3_STOP_REQ	LPUART2_STOP_REQ	LPUART1_STOP_REQ	LPSPI4_STOP_REQ	LPSPI3_STOP_REQ	LPSPI2_STOP_REQ	LPSPI1_STOP_REQ	LPI2C4_STOP_REQ	LPI2C3_STOP_REQ	LPI2C2_STOP_REQ	LPI2C1_STOP_REQ
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_GPR_GPR7 field descriptions

Field	Description
31 LPUART8_ STOP_ACK	LPUART8 stop acknowledge 0 stop acknowledge is not asserted 1 stop acknowledge is asserted (the module is in Stop mode)
30 LPUART7_ STOP_ACK	LPUART7 stop acknowledge 0 stop acknowledge is not asserted 1 stop acknowledge is asserted
29 LPUART6_ STOP_ACK	LPUART6 stop acknowledge 0 stop acknowledge is not asserted 1 stop acknowledge is asserted
28 LPUART5_ STOP_ACK	LPUART5 stop acknowledge 0 stop acknowledge is not asserted 1 stop acknowledge is asserted
27 LPUART4_ STOP_ACK	LPUART4 stop acknowledge 0 stop acknowledge is not asserted 1 stop acknowledge is asserted
26 LPUART3_ STOP_ACK	LPUART3 stop acknowledge 0 stop acknowledge is not asserted 1 stop acknowledge is asserted
25 LPUART2_ STOP_ACK	LPUART1 stop acknowledge 0 stop acknowledge is not asserted 1 stop acknowledge is asserted
24 LPUART1_ STOP_ACK	LPUART1 stop acknowledge 0 stop acknowledge is not asserted 1 stop acknowledge is asserted
23 LPSPI4_STOP_ ACK	LPSPI4 stop acknowledge 0 stop acknowledge is not asserted 1 stop acknowledge is asserted
22 LPSPI3_STOP_ ACK	LPSPI3 stop acknowledge 0 stop acknowledge is not asserted 1 stop acknowledge is asserted
21 LPSPI2_STOP_ ACK	LPSPI2 stop acknowledge 0 stop acknowledge is not asserted 1 stop acknowledge is asserted
20 LPSPI1_STOP_ ACK	LPSPI1 stop acknowledge 0 stop acknowledge is not asserted 1 stop acknowledge is asserted

Table continues on the next page...

IOMUXC_GPR_GPR7 field descriptions (continued)

Field	Description
19 LPI2C4_STOP_ACK	LPI2C4 stop acknowledge 0 stop acknowledge is not asserted 1 stop acknowledge is asserted
18 LPI2C3_STOP_ACK	LPI2C3 stop acknowledge 0 stop acknowledge is not asserted 1 stop acknowledge is asserted
17 LPI2C2_STOP_ACK	LPI2C2 stop acknowledge 0 stop acknowledge is not asserted 1 stop acknowledge is asserted
16 LPI2C1_STOP_ACK	LPI2C1 stop acknowledge 0 stop acknowledge is not asserted 1 stop acknowledge is asserted (the module is in Stop mode)
15 LPUART8_STOP_REQ	LPUART8 stop request 0 stop request off 1 stop request on
14 LPUART7_STOP_REQ	LPUART7 stop request 0 stop request off 1 stop request on
13 LPUART6_STOP_REQ	LPUART6 stop request 0 stop request off 1 stop request on
12 LPUART5_STOP_REQ	LPUART5 stop request 0 stop request off 1 stop request on
11 LPUART4_STOP_REQ	LPUART4 stop request 0 stop request off 1 stop request on
10 LPUART3_STOP_REQ	LPUART3 stop request 0 stop request off 1 stop request on
9 LPUART2_STOP_REQ	LPUART1 stop request 0 stop request off 1 stop request on
8 LPUART1_STOP_REQ	LPUART1 stop request 0 stop request off 1 stop request on

Table continues on the next page...

IOMUXC_GPR_GPR7 field descriptions (continued)

Field	Description
7 LPSPI4_STOP_REQ	LPSPI4 stop request 0 stop request off 1 stop request on
6 LPSPI3_STOP_REQ	LPSPI3 stop request 0 stop request off 1 stop request on
5 LPSPI2_STOP_REQ	LPSPI2 stop request 0 stop request off 1 stop request on
4 LPSPI1_STOP_REQ	LPSPI1 stop request 0 stop request off 1 stop request on
3 LPI2C4_STOP_REQ	LPI2C4 stop request 0 stop request off 1 stop request on
2 LPI2C3_STOP_REQ	LPI2C3 stop request 0 stop request off 1 stop request on
1 LPI2C2_STOP_REQ	LPI2C2 stop request 0 stop request off 1 stop request on
0 LPI2C1_STOP_REQ	LPI2C1 stop request 0 stop request off 1 stop request on

11.3.9 GPR8 General Purpose Register (IOMUXC_GPR_GPR8)

GPR Register

Address: 400A_C000h base + 20h offset = 400A_C020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	LPUART8_IPG_DOZE	LPUART8_IPG_STOP_MODE	LPUART7_IPG_DOZE	LPUART7_IPG_STOP_MODE	LPUART6_IPG_DOZE	LPUART6_IPG_STOP_MODE	LPUART5_IPG_DOZE	LPUART5_IPG_STOP_MODE	LPUART4_IPG_DOZE	LPUART4_IPG_STOP_MODE	LPUART3_IPG_DOZE	LPUART3_IPG_STOP_MODE	LPUART2_IPG_DOZE	LPUART2_IPG_STOP_MODE	LPUART1_IPG_DOZE	LPUART1_IPG_STOP_MODE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	LPSPI4_IPG_DOZE	LPSPI4_IPG_STOP_MODE	LPSPI3_IPG_DOZE	LPSPI3_IPG_STOP_MODE	LPSPI2_IPG_DOZE	LPSPI2_IPG_STOP_MODE	LPSPI1_IPG_DOZE	LPSPI1_IPG_STOP_MODE	LPI2C4_IPG_DOZE	LPI2C4_IPG_STOP_MODE	LPI2C3_IPG_DOZE	LPI2C3_IPG_STOP_MODE	LPI2C2_IPG_DOZE	LPI2C2_IPG_STOP_MODE	LPI2C1_IPG_DOZE	LPI2C1_IPG_STOP_MODE
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_GPR_GPR8 field descriptions

Field	Description
31 LPUART8_IPG_DOZE	LPUART8 ipg_doze mode 0 not in doze mode 1 in doze mode
30 LPUART8_IPG_STOP_MODE	LPUART8 stop mode selection, cannot change when ipg_stop is asserted. 0 the module is functional in Stop mode 1 the module is NOT functional in Stop mode, when this bit is equal to 1 and ipg_stop is asserted
29 LPUART7_IPG_DOZE	LPUART7 ipg_doze mode 0 not in doze mode 1 in doze mode
28 LPUART7_IPG_STOP_MODE	LPUART7 stop mode selection, cannot change when ipg_stop is asserted. 0 the module is functional in Stop mode 1 the module is NOT functional in Stop mode, when this bit is equal to 1 and ipg_stop is asserted
27 LPUART6_IPG_DOZE	LPUART6 ipg_doze mode 0 not in doze mode 1 in doze mode

Table continues on the next page...

IOMUXC_GPR_GPR8 field descriptions (continued)

Field	Description
26 LPUART6_IPG_STOP_MODE	LPUART6 stop mode selection, cannot change when ipg_stop is asserted. 0 the module is functional in Stop mode 1 the module is NOT functional in Stop mode, when this bit is equal to 1 and ipg_stop is asserted
25 LPUART5_IPG_DOZE	LPUART5 ipg_doze mode 0 not in doze mode 1 in doze mode
24 LPUART5_IPG_STOP_MODE	LPUART5 stop mode selection, cannot change when ipg_stop is asserted. 0 the module is functional in Stop mode 1 the module is NOT functional in Stop mode, when this bit is equal to 1 and ipg_stop is asserted
23 LPUART4_IPG_DOZE	LPUART4 ipg_doze mode 0 not in doze mode 1 in doze mode
22 LPUART4_IPG_STOP_MODE	LPUART4 stop mode selection, cannot change when ipg_stop is asserted. 0 the module is functional in Stop mode 1 the module is NOT functional in Stop mode, when this bit is equal to 1 and ipg_stop is asserted
21 LPUART3_IPG_DOZE	LPUART3 ipg_doze mode 0 not in doze mode 1 in doze mode
20 LPUART3_IPG_STOP_MODE	LPUART3 stop mode selection, cannot change when ipg_stop is asserted. 0 the module is functional in Stop mode 1 the module is NOT functional in Stop mode, when this bit is equal to 1 and ipg_stop is asserted
19 LPUART2_IPG_DOZE	LPUART2 ipg_doze mode 0 not in doze mode 1 in doze mode
18 LPUART2_IPG_STOP_MODE	LPUART2 stop mode selection, cannot change when ipg_stop is asserted. 0 the module is functional in Stop mode 1 the module is NOT functional in Stop mode, when this bit is equal to 1 and ipg_stop is asserted
17 LPUART1_IPG_DOZE	LPUART1 ipg_doze mode 0 not in doze mode 1 in doze mode
16 LPUART1_IPG_STOP_MODE	LPUART1 stop mode selection, cannot change when ipg_stop is asserted. 0 the module is functional in Stop mode 1 the module is NOT functional in Stop mode, when this bit is equal to 1 and ipg_stop is asserted
15 LPSPI4_IPG_DOZE	LPSPI4 ipg_doze mode 0 not in doze mode 1 in doze mode

Table continues on the next page...

IOMUXC_GPR_GPR8 field descriptions (continued)

Field	Description
14 LPSPI4_IPG_ STOP_MODE	LPSPI4 stop mode selection, cannot change when ipg_stop is asserted. 0 the module is functional in Stop mode 1 the module is NOT functional in Stop mode, when this bit is equal to 1 and ipg_stop is asserted
13 LPSPI3_IPG_ DOZE	LPSPI3 ipg_doze mode 0 not in doze mode 1 in doze mode
12 LPSPI3_IPG_ STOP_MODE	LPSPI3 stop mode selection, cannot change when ipg_stop is asserted. 0 the module is functional in Stop mode 1 the module is NOT functional in Stop mode, when this bit is equal to 1 and ipg_stop is asserted
11 LPSPI2_IPG_ DOZE	LPSPI2 ipg_doze mode 0 not in doze mode 1 in doze mode
10 LPSPI2_IPG_ STOP_MODE	LPSPI2 stop mode selection, cannot change when ipg_stop is asserted. 0 the module is functional in Stop mode 1 the module is NOT functional in Stop mode, when this bit is equal to 1 and ipg_stop is asserted
9 LPSPI1_IPG_ DOZE	LPSPI1 ipg_doze mode 0 not in doze mode 1 in doze mode
8 LPSPI1_IPG_ STOP_MODE	LPSPI1 stop mode selection, cannot change when ipg_stop is asserted. 0 the module is functional in Stop mode 1 the module is NOT functional in Stop mode, when this bit is equal to 1 and ipg_stop is asserted
7 LPI2C4_IPG_ DOZE	LPI2C4 ipg_doze mode 0 not in doze mode 1 in doze mode
6 LPI2C4_IPG_ STOP_MODE	LPI2C4 stop mode selection, cannot change when ipg_stop is asserted. 0 the module is functional in Stop mode 1 the module is NOT functional in Stop mode, when this bit is equal to 1 and ipg_stop is asserted
5 LPI2C3_IPG_ DOZE	LPI2C3 ipg_doze mode 0 not in doze mode 1 in doze mode
4 LPI2C3_IPG_ STOP_MODE	LPI2C3 stop mode selection, cannot change when ipg_stop is asserted. 0 the module is functional in Stop mode 1 the module is NOT functional in Stop mode, when this bit is equal to 1 and ipg_stop is asserted
3 LPI2C2_IPG_ DOZE	LPI2C2 ipg_doze mode 0 not in doze mode 1 in doze mode

Table continues on the next page...

IOMUXC_GPR_GPR8 field descriptions (continued)

Field	Description
2 LPI2C2_IPG_STOP_MODE	LPI2C2 stop mode selection, cannot change when ipg_stop is asserted. 0 the module is functional in Stop mode 1 the module is NOT functional in Stop mode, when this bit is equal to 1 and ipg_stop is asserted
1 LPI2C1_IPG_DOZE	LPI2C1 ipg_doze mode 0 not in doze mode 1 in doze mode
0 LPI2C1_IPG_STOP_MODE	LPI2C1 stop mode selection, cannot change when ipg_stop is asserted. 0 the module is functional in Stop mode 1 the module is NOT functional in Stop mode, when this bit is equal to 1 and ipg_stop is asserted

11.3.10 GPR9 General Purpose Register (IOMUXC_GPR_GPR9)

GPR Register

Address: 400A_C000h base + 24h offset = 400A_C024h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_GPR_GPR9 field descriptions

Field	Description
-	Reserved This field is reserved. General purpose bits to be used by SoC integration. Bit field type: STICKY

11.3.11 GPR10 General Purpose Register (IOMUXC_GPR_GPR10)

GPR Register

Address: 400A_C000h base + 28h offset = 400A_C028h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	LOCK_OCRAM_TZ_ADDR							LOCK_OCRAM_TZ_EN	Reserved			LOCK_DCPKEY_OCOTP_OR_KEYMUX	Reserved	LOCK_SEC_ERR_RESP	LOCK_DBG_EN	LOCK_NIDEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	OCRAM_TZ_ADDR							OCRAM_TZ_EN	Reserved			DCPKEY_OCOTP_OR_KEYMUX	Reserved	SEC_ERR_RESP	DBG_EN	NIDEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

IOMUXC_GPR_GPR10 field descriptions

Field	Description
31–25 LOCK_OCRAM_TZ_ADDR	Lock OCRAM_TZ_ADDR field for changes. This is a sticky field, once set it cannot be cleared (only by reset). 0 Field is not locked 1 Field is locked (read access only)
24 LOCK_OCRAM_TZ_EN	Lock OCRAM_TZ_EN field for changes. This is a sticky field, once set it cannot be cleared (only by reset). 0 Field is not locked 1 Field is locked (read access only)
23–21 -	This field is reserved. Reserved
20 LOCK_DCPKEY_OCOTP_OR_KEYMUX	Lock DCP Key OCOTP/Key MUX selection bit. This is a sticky field, once set it cannot be cleared (only by reset). 0 Field is not locked 1 Field is locked (read access only)

Table continues on the next page...

IOMUXC_GPR_GPR10 field descriptions (continued)

Field	Description
19 -	This field is reserved. Reserved
18 LOCK_SEC_ ERR_RESP	Lock SEC_ERR_RESP field for changes. This is a sticky field, once set it cannot be cleared (only by reset). 0 Field is not locked 1 Field is locked (read access only)
17 LOCK_DBG_EN	Lock DBG_EN field for changes. This is a sticky field, once set it cannot be cleared (only by reset). 0 Field is not locked 1 Field is locked (read access only)
16 LOCK_NIDEN	Lock NIDEN field for changes. This is a sticky field, once set it cannot be cleared (only by reset). 0 Field is not locked 1 Field is locked (read access only)
15–9 OCRAM_TZ_ ADDR	OCRAM TrustZone (TZ) start address. This is the start address of the secure memory region within the OCRAM memory space is 4 KB granularity. The start address affects the OCRAM transactions only if OCRAM_TZ_EN bit is set. The OCRAM TZ ENDADDR is not configurable and is set to the end of OCRAM memory space.
8 OCRAM_TZ_EN	OCRAM TrustZone (TZ) enable. 0 The TrustZone feature is disabled. Entire OCRAM space is available for all access types (secure/non-secure/user/supervisor). 1 The TrustZone feature is enabled. Access to address in the range specified by [ENDADDR:STARTADDR] follows the execution mode access policy described in CSU chapter.
7–5 -	This field is reserved. Reserved
4 DCPKEY_ OCOTP_OR_ KEYMUX	DCP Key selection bit. 0 Select key from SNVS Master Key. 1 Select key from OCOTP (SW_GP2).
3 -	This field is reserved. Reserved
2 SEC_ERR_ RESP	Security error response enable for all security gaskets (on both AHB and AXI buses) 0 OKEY response 1 SLVError (default)
1 DBG_EN	Arm invasive debug enable 0 Debug turned off. 1 Debug enabled (default).
0 NIDEN	Arm non-secure (non-invasive) debug enable 0 Debug turned off. 1 Debug enabled (default).

11.3.12 GPR11 General Purpose Register (IOMUXC_GPR_GPR11)

GPR Register

Address: 400A_C000h base + 2Ch offset = 400A_C02Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved				BEE_DE_RX_EN				M7_APC_AC_R3_CTRL		M7_APC_AC_R2_CTRL		M7_APC_AC_R1_CTRL		M7_APC_AC_R0_CTRL	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_GPR_GPR11 field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
15–12 -	This field is reserved. Reserved
11–8 BEE_DE_RX_EN	BEE data decryption of memory region-*n* (*n* = 3 to 0). • 0: FlexSPI data decryption disabled • 1: FlexSPI data decryption enabled
7–6 M7_APC_AC_R3_CTRL	Access control of memory region-3 00 No access protection 01 M7 debug protection enabled 10 Reserved 11 Reserved
5–4 M7_APC_AC_R2_CTRL	Access control of memory region-2 00 No access protection 01 M7 debug protection enabled 10 Reserved 11 Reserved
3–2 M7_APC_AC_R1_CTRL	Access control of memory region-1 00 No access protection 01 M7 debug protection enabled 10 Reserved 11 Reserved
M7_APC_AC_R0_CTRL	Access control of memory region-0 00 No access protection 01 M7 debug protection enabled

Table continues on the next page...

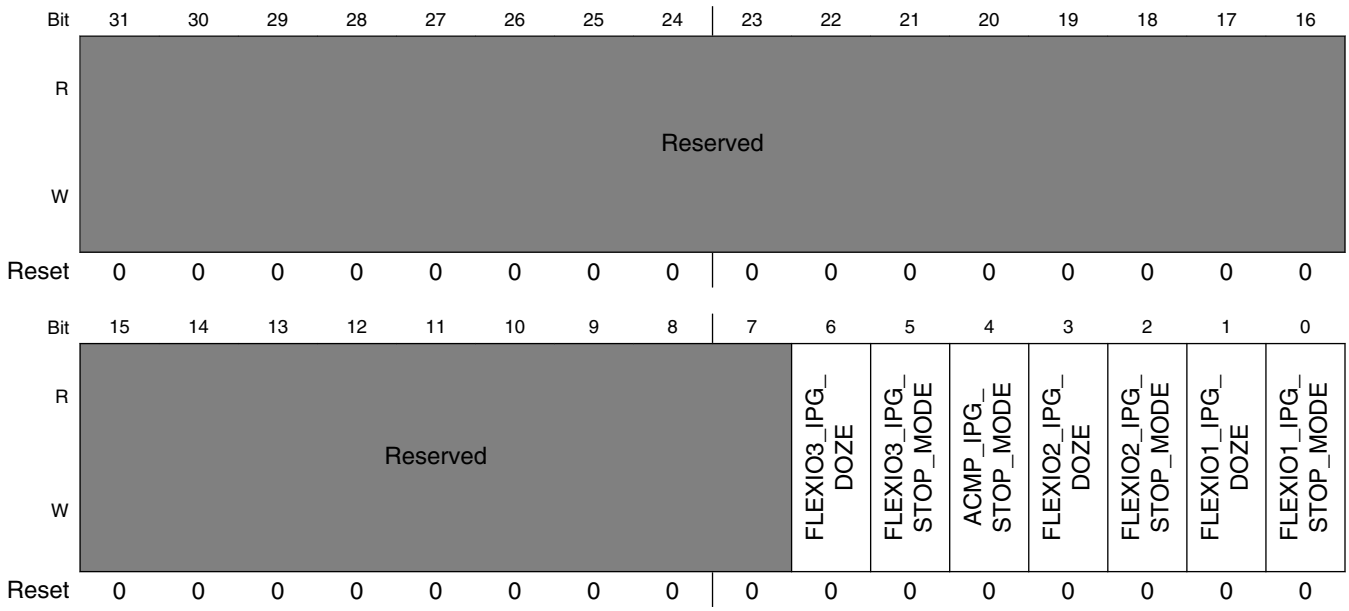
IOMUXC_GPR_GPR11 field descriptions (continued)

Field	Description
	10 Reserved 11 Reserved

11.3.13 GPR12 General Purpose Register (IOMUXC_GPR_GPR12)

GPR Register

Address: 400A_C000h base + 30h offset = 400A_C030h



IOMUXC_GPR_GPR12 field descriptions

Field	Description
31–7 -	This field is reserved. Reserved
6 FLEXIO3_IPG_DOZE	FLEXIO3 ipg_doze mode 0 FLEXIO3 is not in doze mode 1 FLEXIO3 is in doze mode
5 FLEXIO3_IPG_STOP_MODE	FlexIO3 stop mode selection. Cannot change when ipg_stop is asserted. 0 FlexIO3 is functional in Stop mode. 1 When this bit is equal to 1'b1 and ipg_stop is asserted, FlexIO3 is not functional in Stop mode.
4 ACMP_IPG_STOP_MODE	ACMP stop mode selection. Cannot change when ipg_stop is asserted. 0 ACMP is functional in Stop mode. 1 When this bit is equal to 1'b1 and ipg_stop is asserted, ACMP is not functional in Stop mode.

Table continues on the next page...

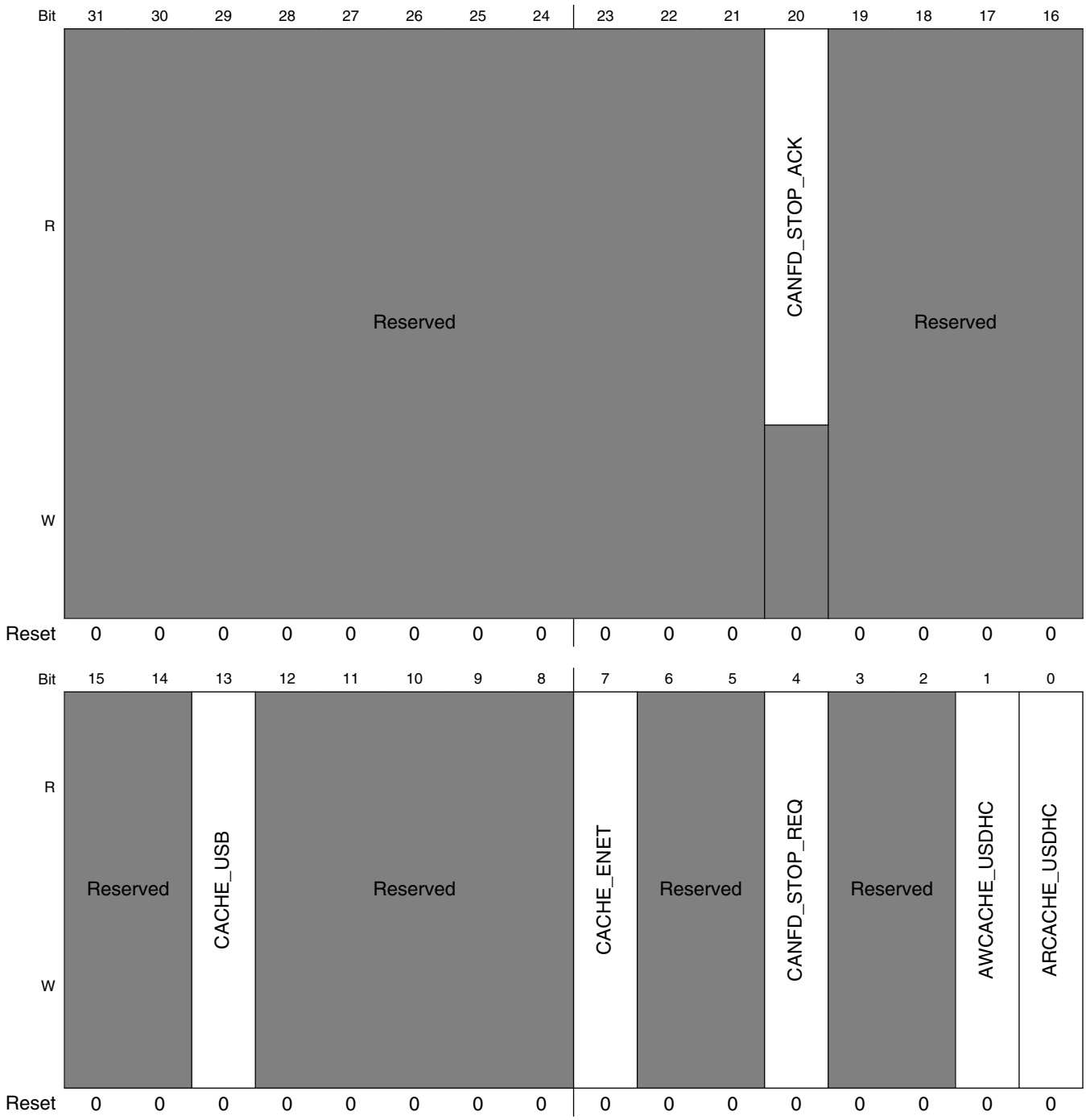
IOMUXC_GPR_GPR12 field descriptions (continued)

Field	Description
3 FLEXIO2_IPG_DOZE	FLEXIO2 ipg_doze mode 0 FLEXIO2 is not in doze mode 1 FLEXIO2 is in doze mode
2 FLEXIO2_IPG_STOP_MODE	FlexIO2 stop mode selection. Cannot change when ipg_stop is asserted. 0 FlexIO2 is functional in Stop mode. 1 When this bit is equal to 1'b1 and ipg_stop is asserted, FlexIO2 is not functional in Stop mode.
1 FLEXIO1_IPG_DOZE	FLEXIO1 ipg_doze mode 0 FLEXIO1 is not in doze mode 1 FLEXIO1 is in doze mode
0 FLEXIO1_IPG_STOP_MODE	FlexIO1 stop mode selection. Cannot change when ipg_stop is asserted. 0 FlexIO1 is functional in Stop mode. 1 When this bit is equal to 1'b1 and ipg_stop is asserted, FlexIO1 is not functional in Stop mode.

11.3.14 GPR13 General Purpose Register (IOMUXC_GPR_GPR13)

GPR Register

Address: 400A_C000h base + 34h offset = 400A_C034h



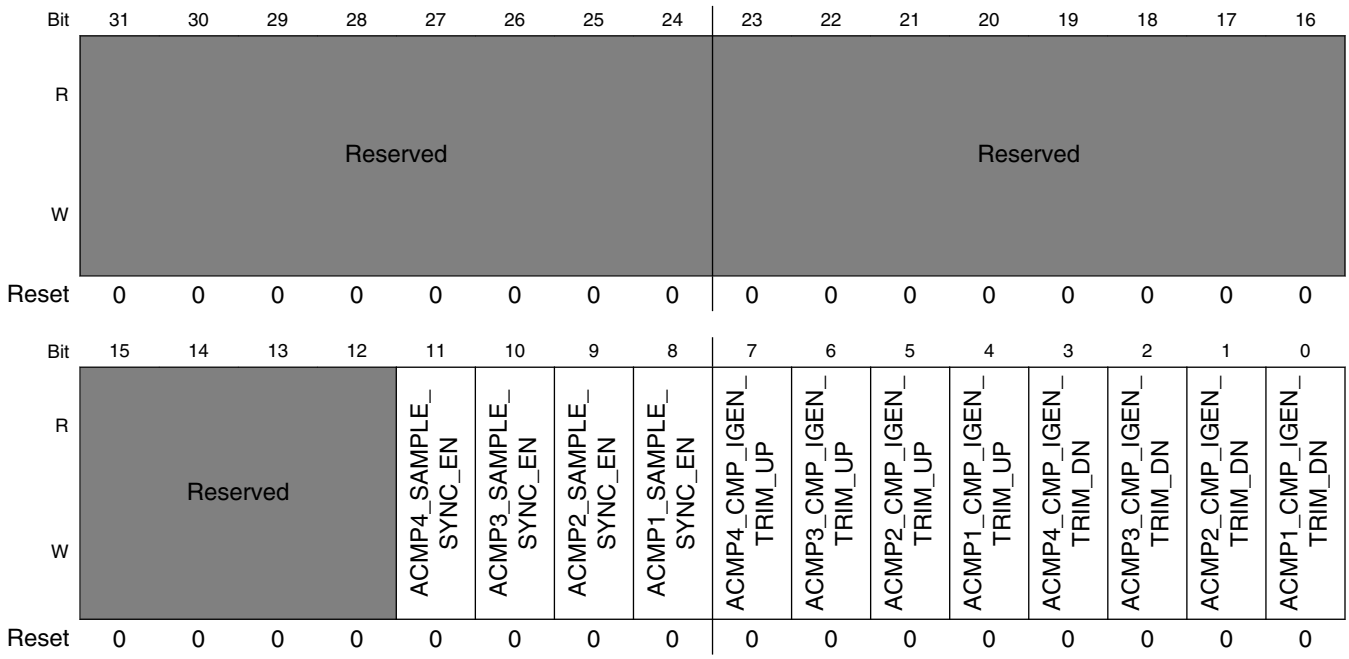
IOMUXC_GPR_GPR13 field descriptions

Field	Description
31–21 -	This field is reserved. Reserved
20 CANFD_STOP_ACK	CANFD stop acknowledge. 0 CANFD stop acknowledge is not asserted 1 CANFD stop acknowledge is asserted
19–14 -	This field is reserved. Reserved
13 CACHE_USB	USB block cacheable attribute value of AXI transactions 0 Cacheable attribute is off for read/write transactions. 1 Cacheable attribute is on for read/write transactions.
12–8 -	This field is reserved. Reserved
7 CACHE_ENET	ENET block cacheable attribute value of AXI transactions 0 Cacheable attribute is off for read/write transactions. 1 Cacheable attribute is on for read/write transactions.
6–5 -	This field is reserved. Reserved
4 CANFD_STOP_REQ	CANFD stop request. 0 stop request off 1 stop request on
3–2 -	This field is reserved. Reserved
1 AWCACHE_USDHC	uSDHC block cacheable attribute value of AXI write transactions 0 Cacheable attribute is off for write transactions. 1 Cacheable attribute is on for write transactions.
0 ARCACHE_USDHC	uSDHC block cacheable attribute value of AXI read transactions 0 Cacheable attribute is off for read transactions. 1 Cacheable attribute is on for read transactions.

11.3.15 GPR14 General Purpose Register (IOMUXC_GPR_GPR14)

GPR Register

Address: 400A_C000h base + 38h offset = 400A_C038h



IOMUXC_GPR_GPR14 field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23–16 -	This field is reserved. Reserved
15–12 -	This field is reserved. Reserved
11 ACMP4_ SAMPLE_ SYNC_EN	ACMP4 sample_lv source select 0 select XBAR output 1 select synced sample_lv
10 ACMP3_ SAMPLE_ SYNC_EN	ACMP3 sample_lv source select 0 select XBAR output 1 select synced sample_lv
9 ACMP2_ SAMPLE_ SYNC_EN	ACMP2 sample_lv source select 0 select XBAR output 1 select synced sample_lv

Table continues on the next page...

IOMUXC_GPR_GPR14 field descriptions (continued)

Field	Description
8 ACMP1_ SAMPLE_ SYNC_EN	ACMP1 sample_lv source select 0 select XBAR output 1 select synced sample_lv
7 ACMP4_CMP_ IGEN_TRIM_UP	increases ACMP4 internal bias current by 30% 0 no increase 1 increases
6 ACMP3_CMP_ IGEN_TRIM_UP	increases ACMP3 internal bias current by 30% 0 no increase 1 increases
5 ACMP2_CMP_ IGEN_TRIM_UP	increases ACMP2 internal bias current by 30% 0 no increase 1 increases
4 ACMP1_CMP_ IGEN_TRIM_UP	increases ACMP1 internal bias current by 30% 0 no increase 1 increases
3 ACMP4_CMP_ IGEN_TRIM_DN	reduces ACMP4 internal bias current by 30% 0 no reduce 1 reduces
2 ACMP3_CMP_ IGEN_TRIM_DN	reduces ACMP3 internal bias current by 30% 0 no reduce 1 reduces
1 ACMP2_CMP_ IGEN_TRIM_DN	reduces ACMP2 internal bias current by 30% 0 no reduce 1 reduces
0 ACMP1_CMP_ IGEN_TRIM_DN	reduces ACMP1 internal bias current by 30% 0 no reduce 1 reduces

11.3.16 GPR15 General Purpose Register (IOMUXC_GPR_GPR15)

GPR Register

Address: 400A_C000h base + 3Ch offset = 400A_C03Ch

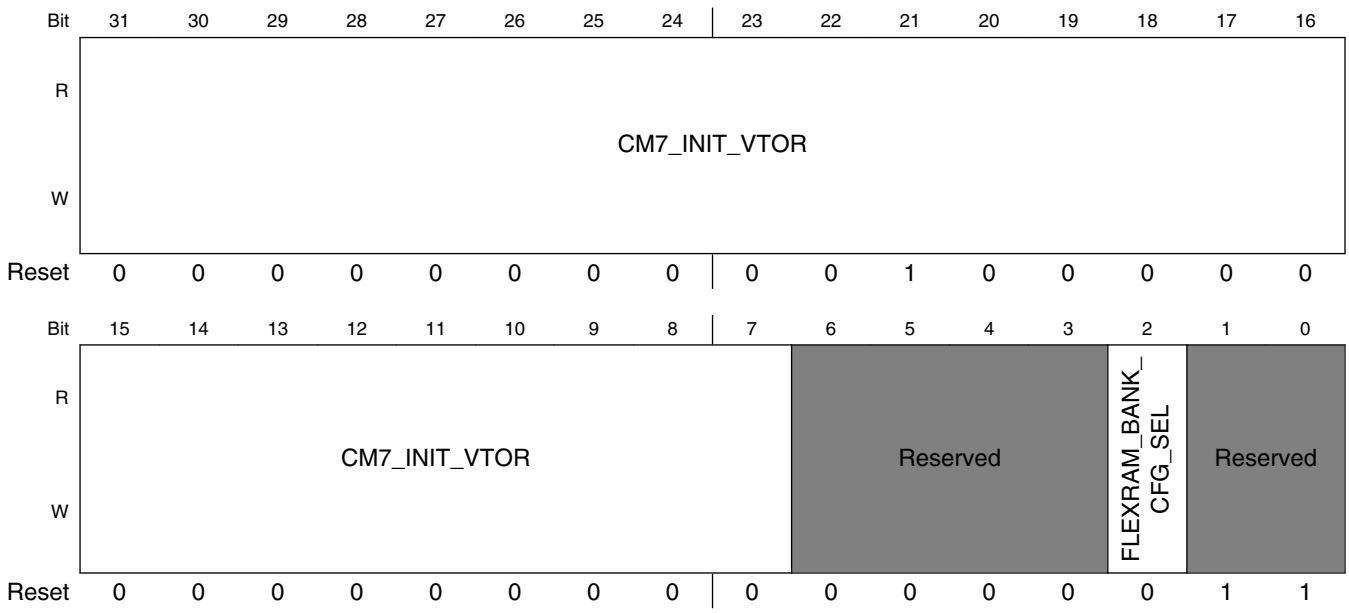
IOMUXC_GPR_GPR15 field descriptions

Field	Description
-	Reserved This field is reserved. General purpose bits to be used by SoC integration. Bit field type: DEFAULT

11.3.17 GPR16 General Purpose Register (IOMUXC_GPR_GPR16)

GPR Register

Address: 400A_C000h base + 40h offset = 400A_C040h



IOMUXC_GPR_GPR16 field descriptions

Field	Description
31–7 CM7_INIT_VTOR	Vector table offset register out of reset. See the Arm v7-M Architecture Reference Manual for more information about the vector table offset register (VTOR).
6–3 -	This field is reserved. Reserved
2 FLEXRAM_BANK_CFG_SEL	FlexRAM bank config source select 0 use fuse value to config 1 use FLEXRAM_BANK_CFG to config
-	This field is reserved. Reserved

11.3.18 GPR17 General Purpose Register (IOMUXC_GPR_GPR17)

GPR Register

Address: 400A_C000h base + 44h offset = 400A_C044h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	FLEXRAM_BANK_CFG																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_GPR_GPR17 field descriptions

Field	Description
FLEXRAM_ BANK_CFG	FlexRAM bank config value GPR_FLEXRAM_BANK_CFG[2*n*+1 : 2*n*], where n = 0, 1, ..., 15 • 00: RAM bank *n* is not used • 01: RAM bank *n* is OCRAM • 10: RAM bank *n* is DTCM • 11: RAM bank *n* is ITCM

11.3.19 GPR18 General Purpose Register (IOMUXC_GPR_GPR18)

GPR Register

Address: 400A_C000h base + 48h offset = 400A_C048h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	M7_APC_AC_R0_BOT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	M7_APC_AC_R0_BOT													Reserved		LOCK_M7_APC_AC_R0_BOT
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

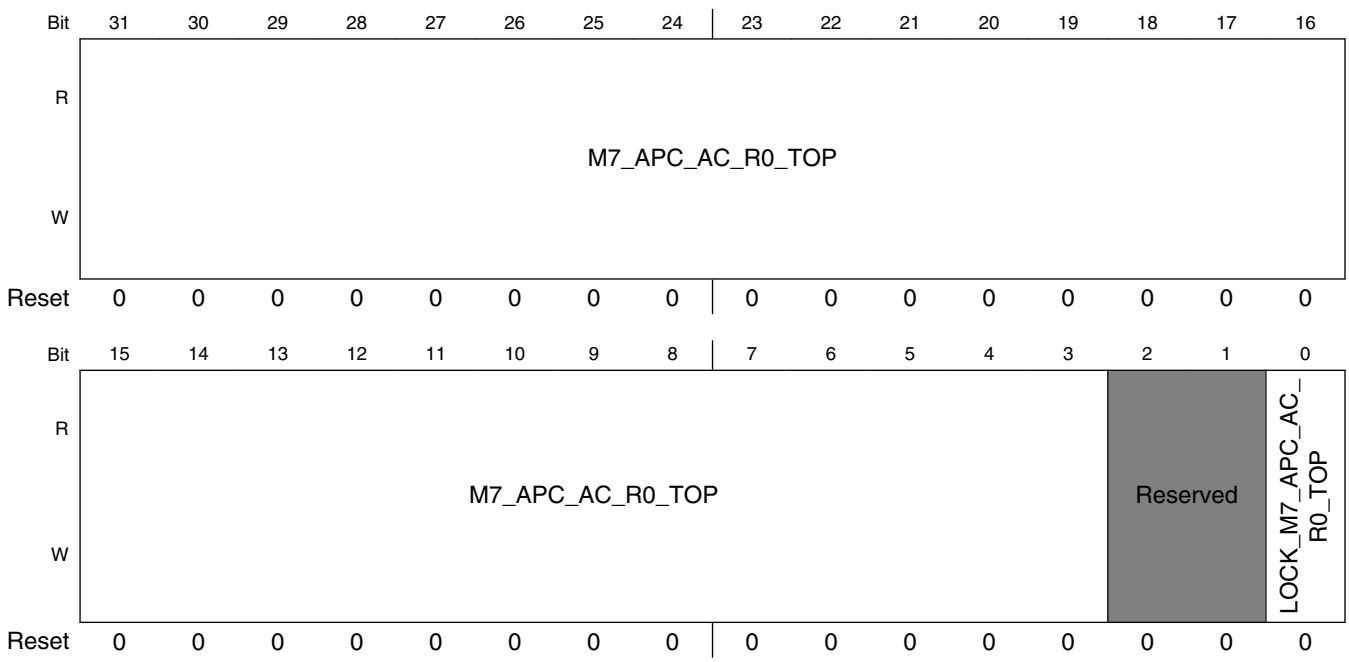
IOMUXC_GPR_GPR18 field descriptions

Field	Description
31–3 M7_APC_AC_R0_BOT	APC end address of memory region-0
2–1 -	This field is reserved. Reserved
0 LOCK_M7_APC_AC_R0_BOT	lock M7_APC_AC_R0_BOT field for changes. This is a sticky field, once set it cannot be cleared (only by reset). 0 Register field [31:1] is not locked 1 Register field [31:1] is locked (read access only)

11.3.20 GPR19 General Purpose Register (IOMUXC_GPR_GPR19)

GPR Register

Address: 400A_C000h base + 4Ch offset = 400A_C04Ch



IOMUXC_GPR_GPR19 field descriptions

Field	Description
31–3 M7_APC_AC_R0_TOP	APC start address of memory region-0
2–1 -	This field is reserved. Reserved

Table continues on the next page...

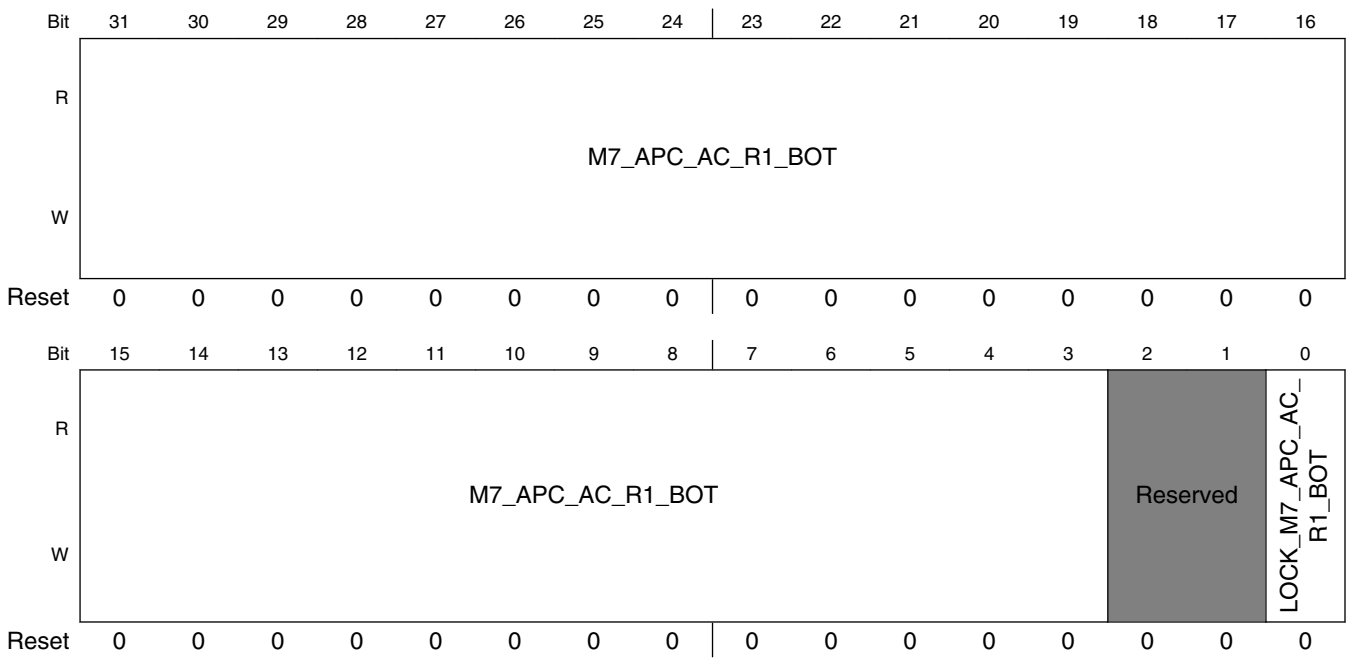
IOMUXC_GPR_GPR19 field descriptions (continued)

Field	Description
0 LOCK_M7_APC_AC_R0_TOP	lock M7_APC_AC_R0_TOP field for changes. This is a sticky field, once set it cannot be cleared (only by reset). 0 Register field [31:1] is not locked 1 Register field [31:1] is locked (read access only)

11.3.21 GPR20 General Purpose Register (IOMUXC_GPR_GPR20)

GPR Register

Address: 400A_C000h base + 50h offset = 400A_C050h



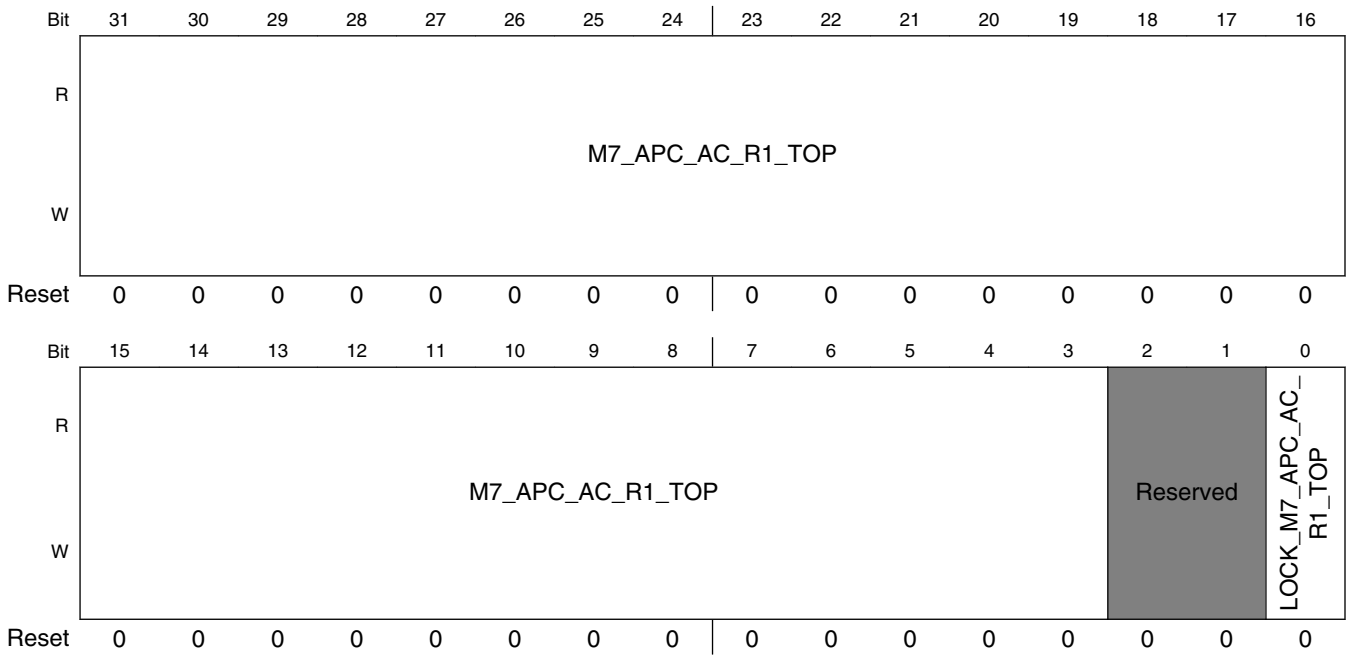
IOMUXC_GPR_GPR20 field descriptions

Field	Description
31–3 M7_APC_AC_R1_BOT	APC end address of memory region-1
2–1 -	This field is reserved. Reserved
0 LOCK_M7_APC_AC_R1_BOT	lock M7_APC_AC_R1_BOT field for changes. This is a sticky field, once set it cannot be cleared (only by reset). 0 Register field [31:1] is not locked 1 Register field [31:1] is locked (read access only)

11.3.22 GPR21 General Purpose Register (IOMUXC_GPR_GPR21)

GPR Register

Address: 400A_C000h base + 54h offset = 400A_C054h



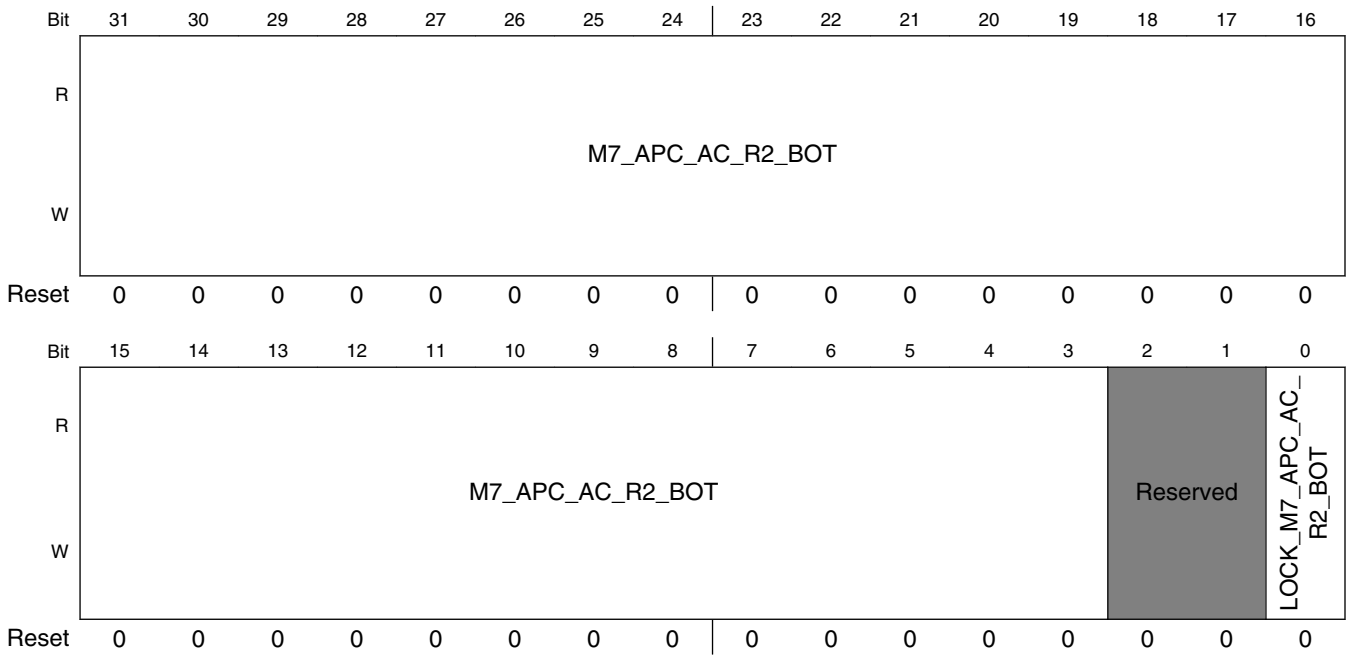
IOMUXC_GPR_GPR21 field descriptions

Field	Description
31–3 M7_APC_AC_R1_TOP	APC start address of memory region-1
2–1 -	This field is reserved. Reserved
0 LOCK_M7_APC_AC_R1_TOP	lock M7_APC_AC_R1_TOP field for changes. This is a sticky field, once set it cannot be cleared (only by reset). 0 Register field [31:1] is not locked 1 Register field [31:1] is locked (read access only)

11.3.23 GPR22 General Purpose Register (IOMUXC_GPR_GPR22)

GPR Register

Address: 400A_C000h base + 58h offset = 400A_C058h



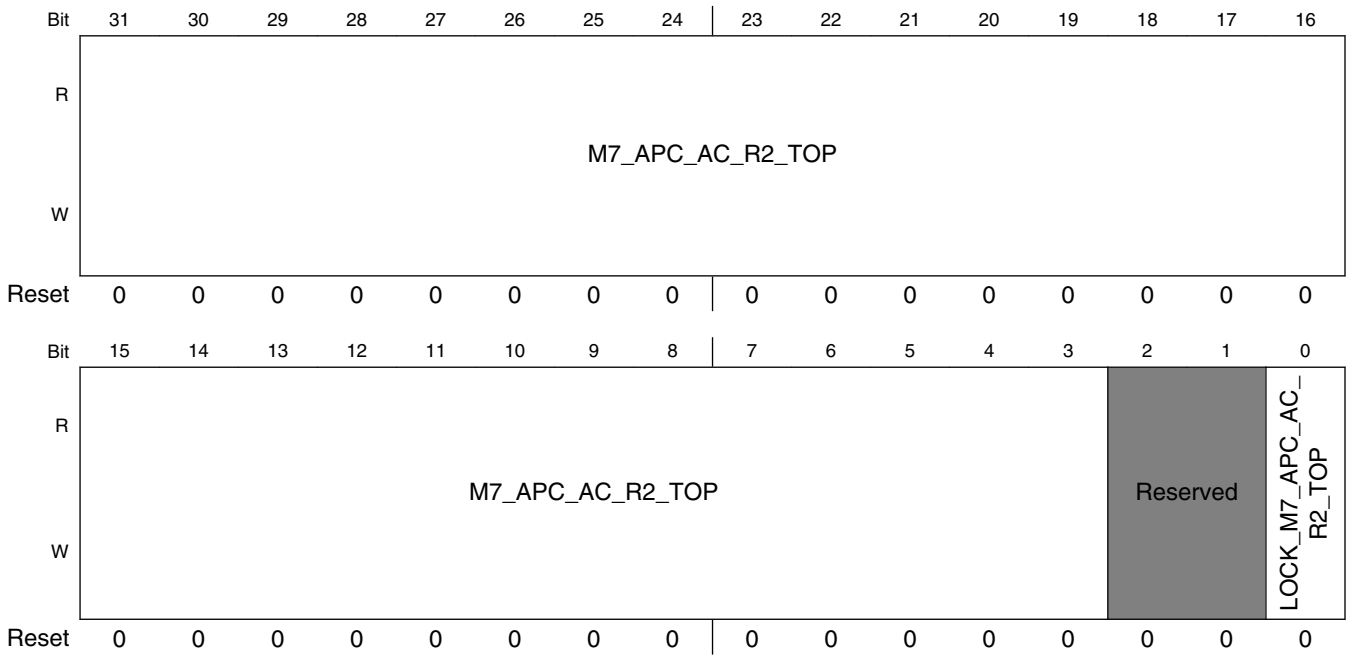
IOMUXC_GPR_GPR22 field descriptions

Field	Description
31–3 M7_APC_AC_R2_BOT	APC end address of memory region-2
2–1 -	This field is reserved. Reserved
0 LOCK_M7_APC_AC_R2_BOT	lock M7_APC_AC_R2_BOT field for changes. This is a sticky field, once set it cannot be cleared (only by reset). 0 Register field [31:1] is not locked 1 Register field [31:1] is locked (read access only)

11.3.24 GPR23 General Purpose Register (IOMUXC_GPR_GPR23)

GPR Register

Address: 400A_C000h base + 5Ch offset = 400A_C05Ch



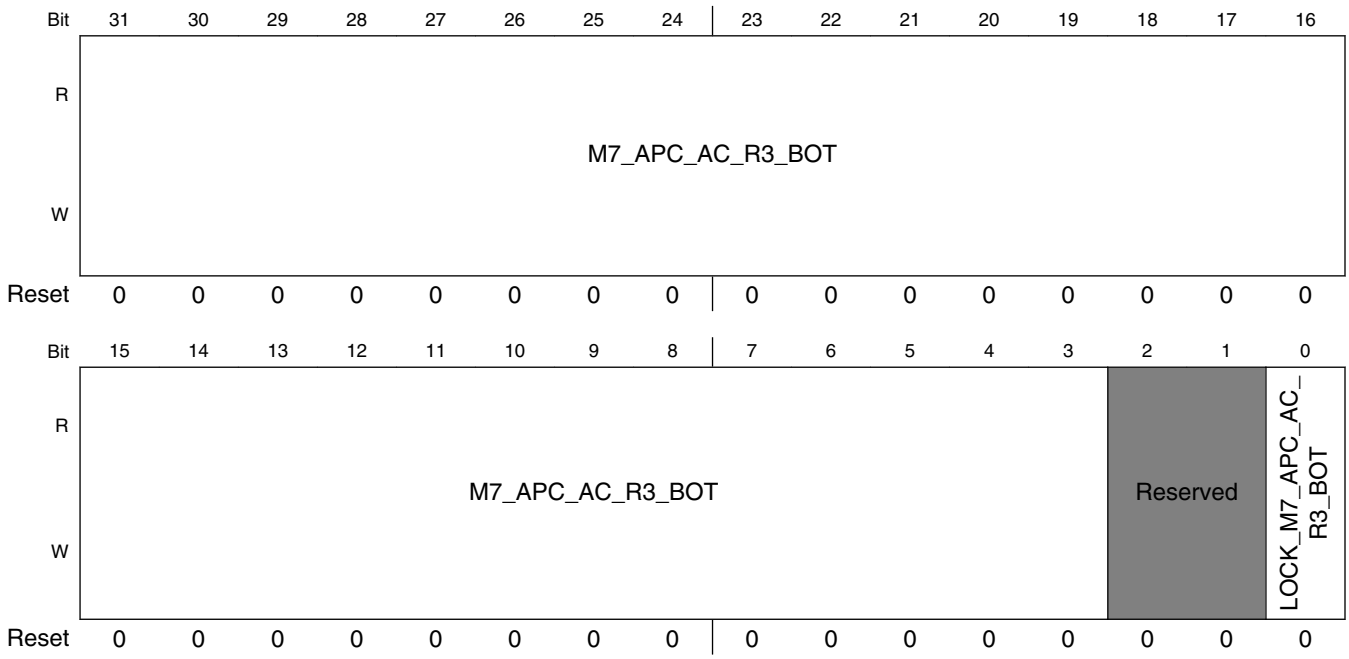
IOMUXC_GPR_GPR23 field descriptions

Field	Description
31–3 M7_APC_AC_R2_TOP	APC start address of memory region-2
2–1 -	This field is reserved. Reserved
0 LOCK_M7_APC_AC_R2_TOP	lock M7_APC_AC_R2_TOP field for changes. This is a sticky field, once set it cannot be cleared (only by reset). 0 Register field [31:1] is not locked 1 Register field [31:1] is locked (read access only)

11.3.25 GPR24 General Purpose Register (IOMUXC_GPR_GPR24)

GPR Register

Address: 400A_C000h base + 60h offset = 400A_C060h



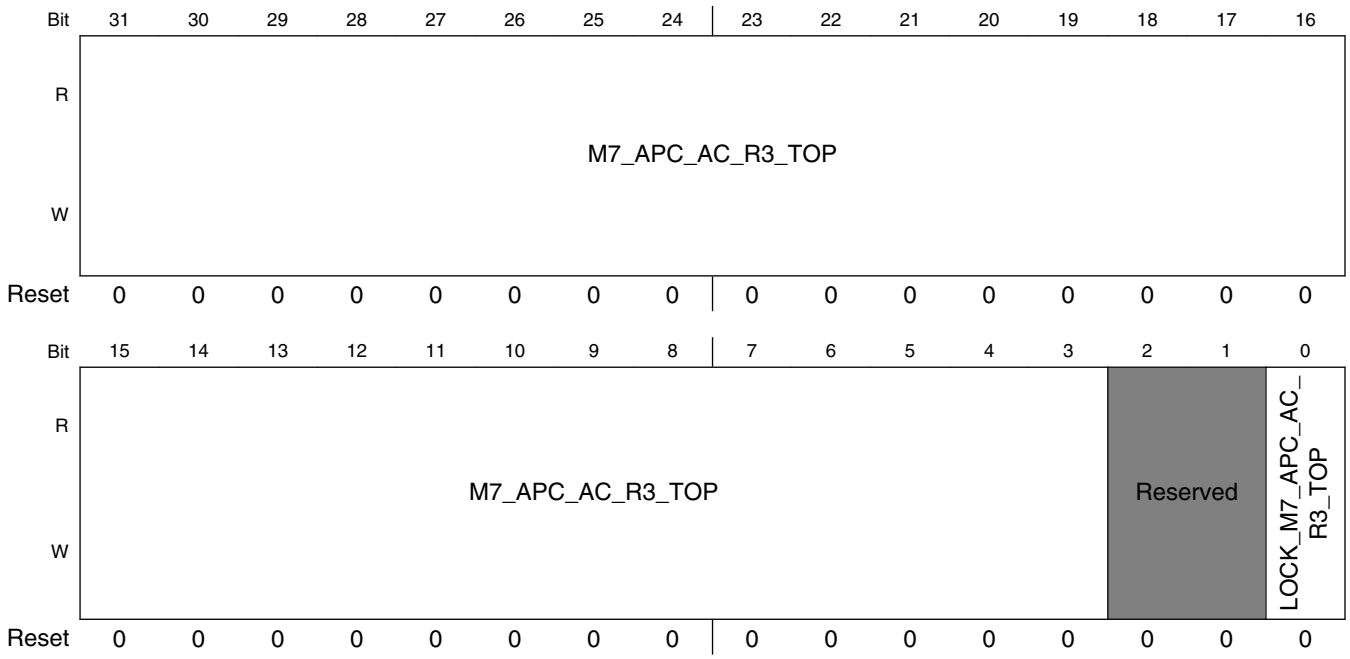
IOMUXC_GPR_GPR24 field descriptions

Field	Description
31–3 M7_APC_AC_R3_BOT	APC end address of memory region-3
2–1 -	This field is reserved. Reserved
0 LOCK_M7_APC_AC_R3_BOT	lock M7_APC_AC_R3_BOT field for changes. This is a sticky field, once set it cannot be cleared (only by reset). 0 Register field [31:1] is not locked 1 Register field [31:1] is locked (read access only)

11.3.26 GPR25 General Purpose Register (IOMUXC_GPR_GPR25)

GPR Register

Address: 400A_C000h base + 64h offset = 400A_C064h



IOMUXC_GPR_GPR25 field descriptions

Field	Description
31–3 M7_APC_AC_R3_TOP	APC start address of memory region-3
2–1 -	This field is reserved. Reserved
0 LOCK_M7_APC_AC_R3_TOP	lock M7_APC_AC_R3_TOP field for changes. This is a sticky field, once set it cannot be cleared (only by reset). 0 Register field [31:1] is not locked 1 Register field [31:1] is locked (read access only)

11.3.27 GPR26 General Purpose Register (IOMUXC_GPR_GPR26)

GPR Register

Address: 400A_C000h base + 68h offset = 400A_C068h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GPIO_MUX1_GPIO_SEL																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_GPR_GPR26 field descriptions

Field	Description
GPIO_MUX1_GPIO_SEL	GPIO1 and GPIO6 share same IO MUX function, GPIO_MUX1 selects one GPIO function. This register controls GPIO_MUX1 to select GPIO1 or GPIO6. For bit *n*, • 0: GPIO1[*n*] is selected; • 1: GPIO6[*n*] is selected.

11.3.28 GPR27 General Purpose Register (IOMUXC_GPR_GPR27)

GPR Register

Address: 400A_C000h base + 6Ch offset = 400A_C06Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GPIO_MUX2_GPIO_SEL																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_GPR_GPR27 field descriptions

Field	Description
GPIO_MUX2_GPIO_SEL	GPIO2 and GPIO7 share same IO MUX function, GPIO_MUX2 selects one GPIO function. This register controls GPIO_MUX2 to select GPIO2 or GPIO7. For bit *n*, • 0: GPIO2[*n*] is selected; • 1: GPIO7[*n*] is selected.

11.3.29 GPR28 General Purpose Register (IOMUXC_GPR_GPR28)

GPR Register

Address: 400A_C000h base + 70h offset = 400A_C070h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GPIO_MUX3_GPIO_SEL																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_GPR_GPR28 field descriptions

Field	Description
GPIO_MUX3_GPIO_SEL	GPIO3 and GPIO8 share same IO MUX function, GPIO_MUX3 selects one GPIO function. This register controls GPIO_MUX3 to select GPIO3 or GPIO8. For bit *n*, • 0: GPIO3[*n*] is selected; • 1: GPIO8[*n*] is selected.

11.3.30 GPR29 General Purpose Register (IOMUXC_GPR_GPR29)

GPR Register

Address: 400A_C000h base + 74h offset = 400A_C074h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GPIO_MUX4_GPIO_SEL																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

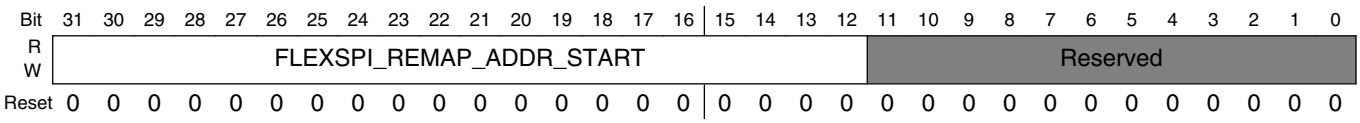
IOMUXC_GPR_GPR29 field descriptions

Field	Description
GPIO_MUX4_GPIO_SEL	GPIO4 and GPIO9 share same IO MUX function, GPIO_MUX4 selects one GPIO function. This register controls GPIO_MUX4 to select GPIO4 or GPIO9. For bit *n*, • 0: GPIO4[*n*] is selected; • 1: GPIO9[*n*] is selected.

11.3.31 GPR30 General Purpose Register (IOMUXC_GPR_GPR30)

GPR Register

Address: 400A_C000h base + 78h offset = 400A_C078h



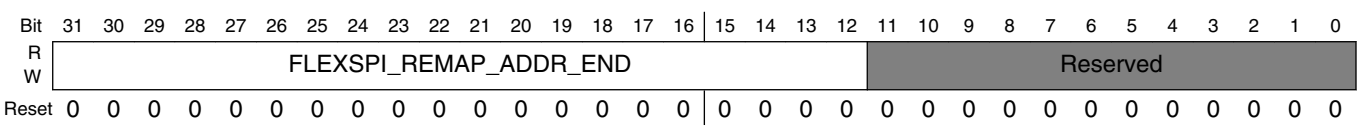
IOMUXC_GPR_GPR30 field descriptions

Field	Description
31–12 FLEXSPI_ REMAP_ADDR_ START	Start address of flexspi1 and flexspi2
-	This field is reserved. Reserved

11.3.32 GPR31 General Purpose Register (IOMUXC_GPR_GPR31)

GPR Register

Address: 400A_C000h base + 7Ch offset = 400A_C07Ch



IOMUXC_GPR_GPR31 field descriptions

Field	Description
31–12 FLEXSPI_ REMAP_ADDR_ END	End address of flexspi1 and flexspi2
-	This field is reserved. Reserved

11.3.33 GPR32 General Purpose Register (IOMUXC_GPR_GPR32)

GPR Register

Address: 400A_C000h base + 80h offset = 400A_C080h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	FLEXSPI_REMAP_ADDR_OFFSET																Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_GPR_GPR32 field descriptions

Field	Description
31–12 FLEXSPI_ REMAP_ADDR_ OFFSET	Offset address of flexspi1 and flexspi2.When ADDR_START[31:12] ≦ Addr_i[31:12] < ADDR_END[31:12], remapped address Addr_o = Addr_i[31:12] + {OFFSET[31:12],12'h0}; Otherwise Addr_o = Addr_i.
-	This field is reserved. Reserved

11.3.34 GPR33 General Purpose Register (IOMUXC_GPR_GPR33)

GPR Register

Address: 400A_C000h base + 84h offset = 400A_C084h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved								LOCK_OCRAM2_TZ_ADDR							LOCK_OCRAM2_TZ_EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								OCRAM2_TZ_ADDR							OCRAM2_TZ_EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1

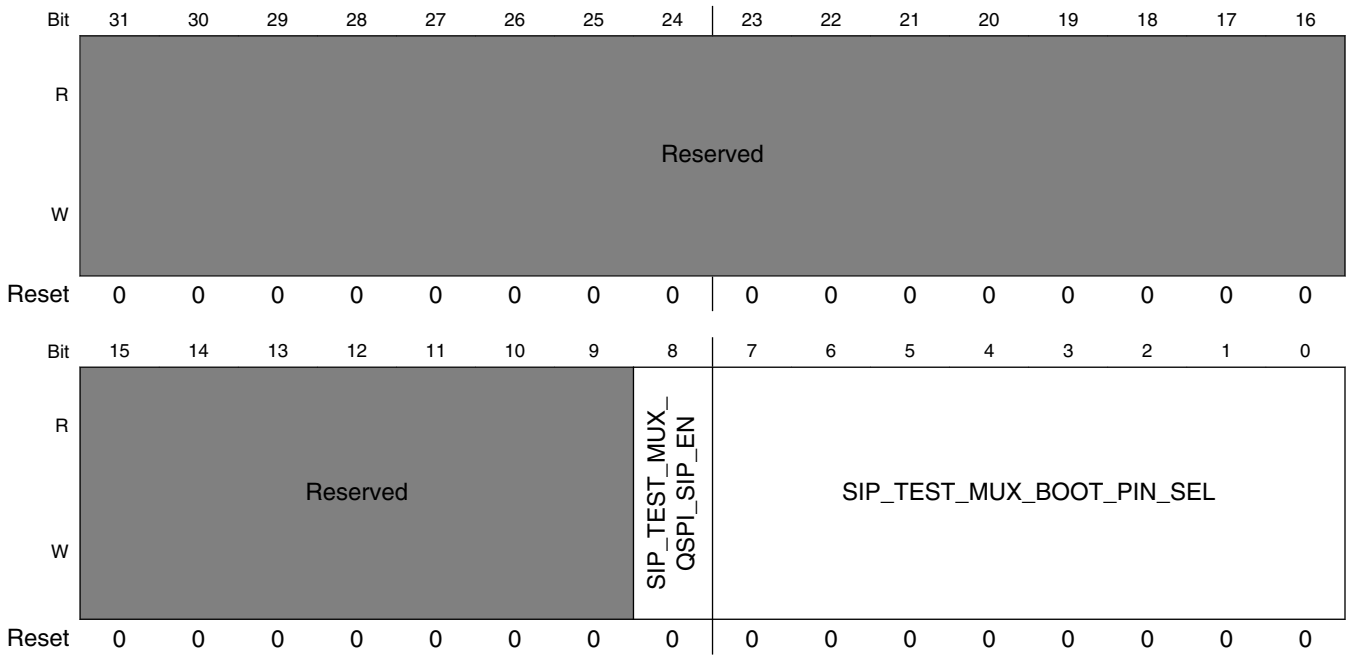
IOMUXC_GPR_GPR33 field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23–17 LOCK_ OCRAM2_TZ_ ADDR	Lock OCRAM2_TZ_ADDR field for changes. This is a sticky field, once set it cannot be cleared (only by reset). 0 Field is not locked 1 Field is locked (read access only)
16 LOCK_ OCRAM2_TZ_ EN	Lock OCRAM2_TZ_EN field for changes. This is a sticky field, once set it cannot be cleared (only by reset). 0 Field is not locked 1 Field is locked (read access only)
15–8 -	This field is reserved. Reserved
7–1 OCRAM2_TZ_ ADDR	OCRAM2 TrustZone (TZ) start address. This is the start address of the secure memory region within the OCRAM2 memory space is 4 KB granularity. The start address affects the OCRAM2 transactions only if OCRAM2_TZ_EN bit is set. The OCRAMe TZ ENDADDR is not configurable and is set to the end of OCRAM2 memory space.
0 OCRAM2_TZ_ EN	OCRAM2 TrustZone (TZ) enable. 0 The TrustZone feature is disabled. Entire OCRAM2 space is available for all access types (secure/non-secure/user/supervisor). 1 The TrustZone feature is enabled. Access to address in the range specified by [ENDADDR:STARTADDR] follows the execution mode access policy described in CSU chapter.

11.3.35 GPR34 General Purpose Register (IOMUXC_GPR_GPR34)

GPR Register

Address: 400A_C000h base + 88h offset = 400A_C088h



IOMUXC_GPR_GPR34 field descriptions

Field	Description
31–9 -	This field is reserved. Reserved
8 SIP_TEST_MUX_QSPI_SIP_EN	Enable SIP_TEST_MUX 0 SIP_TEST_MUX is disabled 1 SIP_TEST_MUX is enabled
SIP_TEST_MUX_BOOT_PIN_SEL	Boot Pin select in SIP_TEST_MUX This field is active only when SIP_TEST_MUX_QSPI_SIP_EN is 1.

11.4 IOMUXC SNVS Memory Map/Register Definition

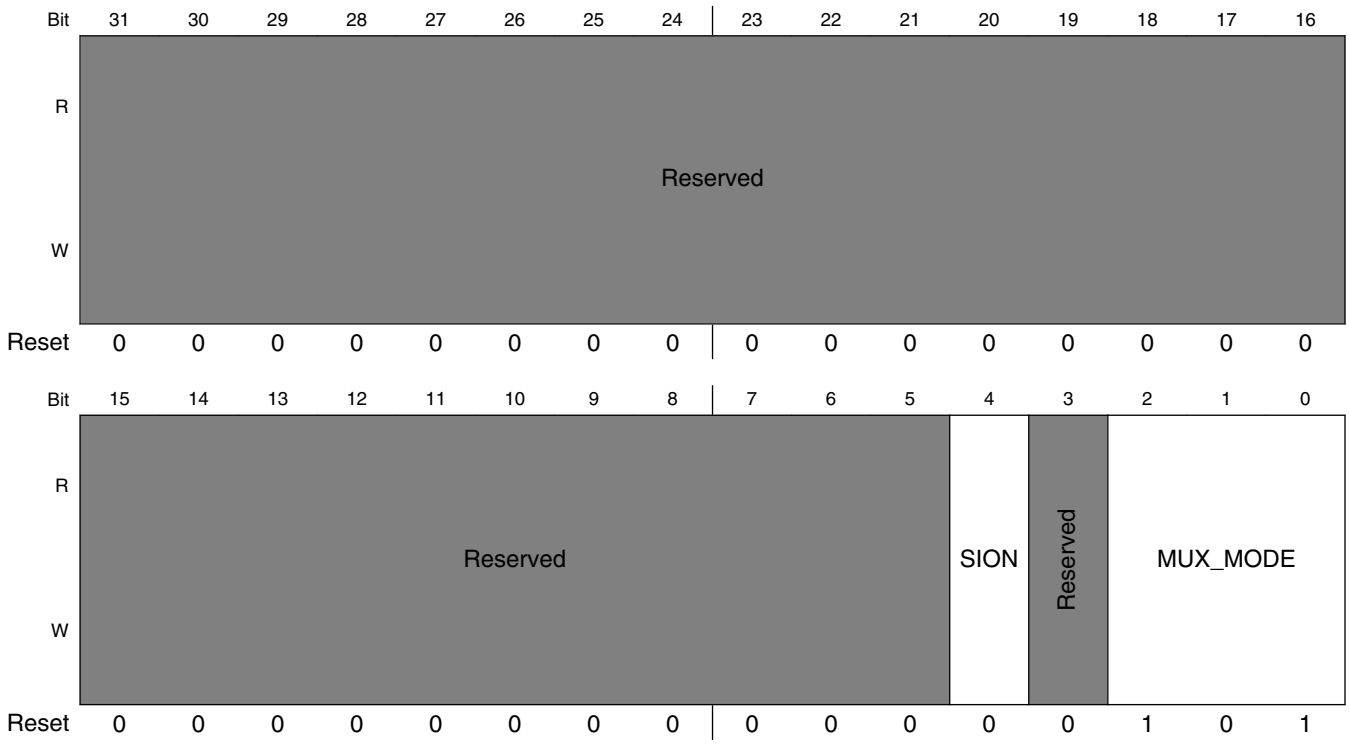
IOMUXC_SNVS memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400A_8000	SW_MUX_CTL_PAD_WAKEUP SW MUX Control Register (IOMUXC_SNVS_SW_MUX_CTL_PAD_WAKEUP)	32	R/W	0000_0005h	11.4.1/378
400A_8004	SW_MUX_CTL_PAD_PMIC_ON_REQ SW MUX Control Register (IOMUXC_SNVS_SW_MUX_CTL_PAD_PMIC_ON_REQ)	32	R/W	0000_0000h	11.4.2/379
400A_8008	SW_MUX_CTL_PAD_PMIC_STBY_REQ SW MUX Control Register (IOMUXC_SNVS_SW_MUX_CTL_PAD_PMIC_STBY_REQ)	32	R/W	0000_0000h	11.4.3/380
400A_800C	SW_PAD_CTL_PAD_TEST_MODE SW PAD Control Register (IOMUXC_SNVS_SW_PAD_CTL_PAD_TEST_MODE)	32	R/W	0000_30A0h	11.4.4/381
400A_8010	SW_PAD_CTL_PAD_POR_B SW PAD Control Register (IOMUXC_SNVS_SW_PAD_CTL_PAD_POR_B)	32	R/W	0001_B0A0h	11.4.5/384
400A_8014	SW_PAD_CTL_PAD_ONOFF SW PAD Control Register (IOMUXC_SNVS_SW_PAD_CTL_PAD_ONOFF)	32	R/W	0001_B0A0h	11.4.6/386
400A_8018	SW_PAD_CTL_PAD_WAKEUP SW PAD Control Register (IOMUXC_SNVS_SW_PAD_CTL_PAD_WAKEUP)	32	R/W	0001_B0A0h	11.4.7/388
400A_801C	SW_PAD_CTL_PAD_PMIC_ON_REQ SW PAD Control Register (IOMUXC_SNVS_SW_PAD_CTL_PAD_PMIC_ON_REQ)	32	R/W	0000_B8A0h	11.4.8/391
400A_8020	SW_PAD_CTL_PAD_PMIC_STBY_REQ SW PAD Control Register (IOMUXC_SNVS_SW_PAD_CTL_PAD_PMIC_STBY_REQ)	32	R/W	0000_A0A0h	11.4.9/393

11.4.1 SW_MUX_CTL_PAD_WAKEUP SW MUX Control Register (IOMUXC_SNVS_SW_MUX_CTL_PAD_WAKEUP)

SW_MUX_CTL Register

Address: 400A_8000h base + 0h offset = 400A_8000h



IOMUXC_SNVS_SW_MUX_CTL_PAD_WAKEUP field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode to Input path no matter what the MUX_MODE functionality is. It is used when the direction of a pin defined by the selected alternative function is an output, but the real logic value on a pin is needed. 1 **ENABLED** — Force input path of pad WAKEUP 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 2 iomux modes to be used for pad: WAKEUP. **Note:** Some functions are available on multiple pins. A given function should not be selected for more than one pin.

Table continues on the next page...

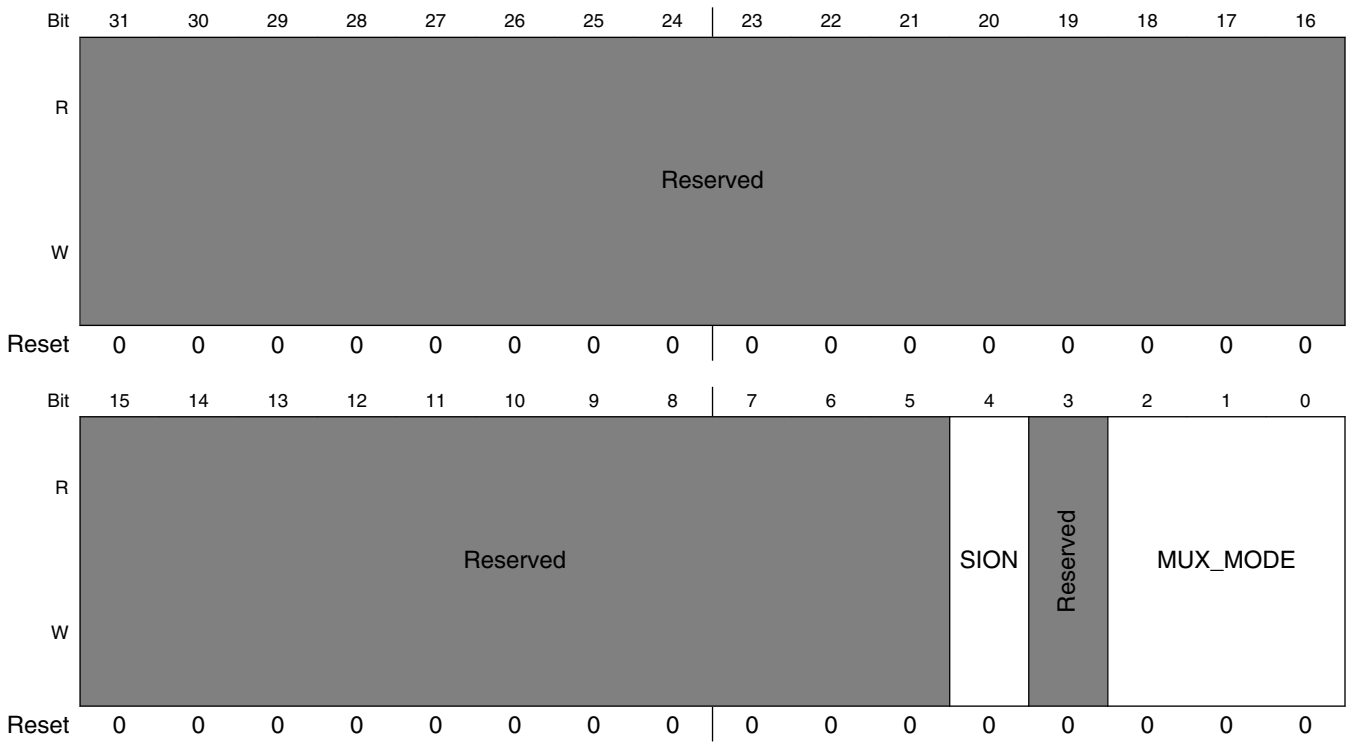
IOMUXC_SNVS_SW_MUX_CTL_PAD_WAKEUP field descriptions (continued)

Field	Description
	101 **ALT5** — Select mux mode: ALT5 mux port: GPIO5_IO00 of instance: gpio5 111 **ALT7** — Select mux mode: ALT7 mux port: NMI of instance: CM7

11.4.2 SW_MUX_CTL_PAD_PMIC_ON_REQ SW MUX Control Register (IOMUXC_SNVS_SW_MUX_CTL_PAD_PMIC_ON_REQ)

SW_MUX_CTL Register

Address: 400A_8000h base + 4h offset = 400A_8004h



IOMUXC_SNVS_SW_MUX_CTL_PAD_PMIC_ON_REQ field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode to Input path no matter what the MUX_MODE functionality is. It is used when the direction of a pin defined by the selected alternative function is an output, but the real logic value on a pin is needed.

Table continues on the next page...

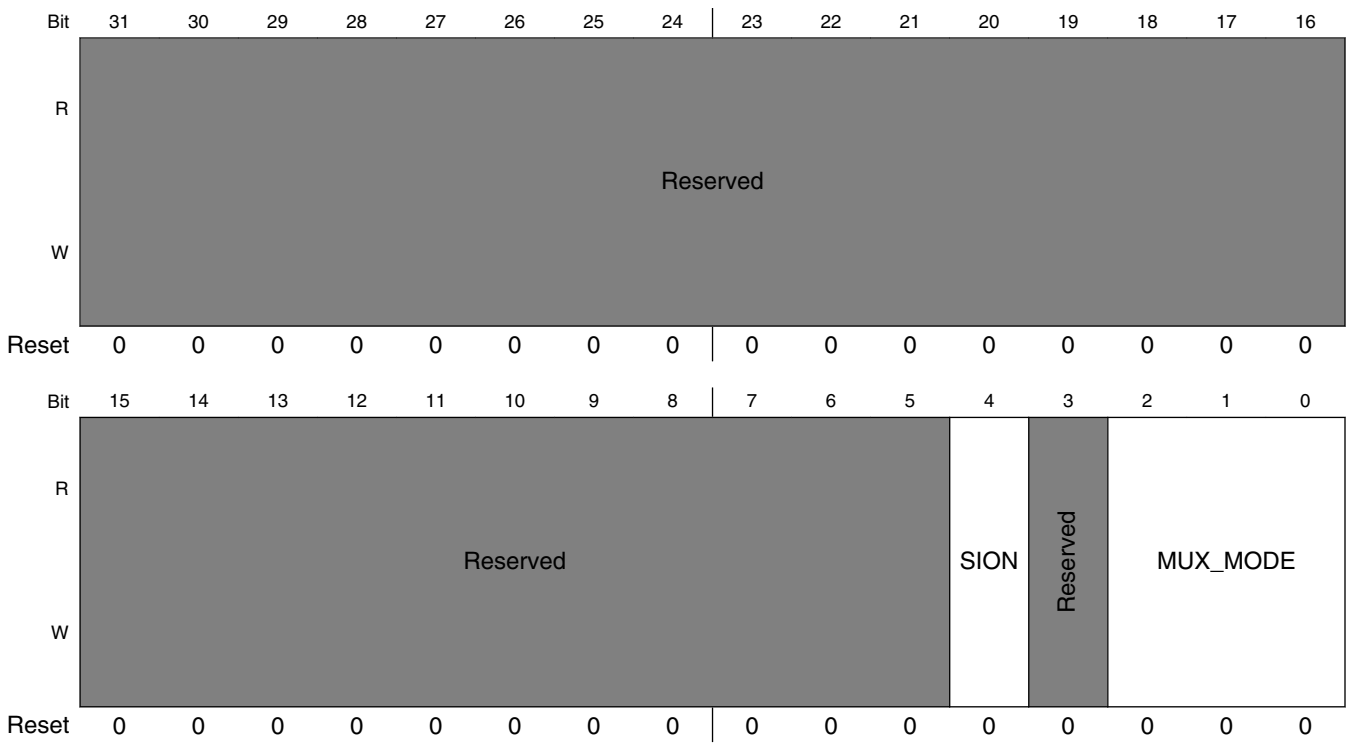
IOMUXC_SNVS_SW_MUX_CTL_PAD_PMIC_ON_REQ field descriptions (continued)

Field	Description
	1 **ENABLED** — Force input path of pad PMIC_ON_REQ 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 2 iomux modes to be used for pad: PMIC_ON_REQ. **Note:** Some functions are available on multiple pins. A given function should not be selected for more than one pin. 000 **ALT0** — Select mux mode: ALT0 mux port: SNVS_PMIC_ON_REQ of instance: snvs 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO5_IO01 of instance: gpio5

11.4.3 SW_MUX_CTL_PAD_PMIC_STBY_REQ SW MUX Control Register (IOMUXC_SNVS_SW_MUX_CTL_PAD_PMIC_STBY_REQ)

SW_MUX_CTL Register

Address: 400A_8000h base + 8h offset = 400A_8008h



IOMUXC_SNVS_SW_MUX_CTL_PAD_PMIC_STBY_REQ field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode to Input path no matter what the MUX_MODE functionality is. It is used when the direction of a pin defined by the selected alternative function is an output, but the real logic value on a pin is needed. 1 **ENABLED** — Force input path of pad PMIC_STBY_REQ 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 2 iomux modes to be used for pad: PMIC_STBY_REQ. **Note:** Some functions are available on multiple pins. A given function should not be selected for more than one pin. 000 **ALT0** — Select mux mode: ALT0 mux port: CCM_PMIC_STBY_REQ of instance: ccm 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO5_IO02 of instance: gpio5

11.4.4 SW_PAD_CTL_PAD_TEST_MODE SW PAD Control Register (IOMUXC_SNVS_SW_PAD_CTL_PAD_TEST_MODE)

SW_PAD_CTL Register

Address: 400A_8000h base + Ch offset = 400A_800Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	1	1	0	0	0	0	1	0	1	0	0	0	0	0

IOMUXC_SNVS_SW_PAD_CTL_PAD_TEST_MODE field descriptions

Field	Description
31–17 -	This field is reserved. Reserved

Table continues on the next page...

IOMUXC_SNVS_SW_PAD_CTL_PAD_TEST_MODE field descriptions (continued)

Field	Description
16 HYS	Hyst. Enable Field Select one out of next values for pad: TEST_MODE The hysteresis (HYS) bit controls whether a pin acts as a Schmitt trigger, which is a comparator remembering its last input state (hysteresis). 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled (CMOS input) 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled (Schmitt Trigger input)
15–14 PUS	Pull Up / Down Config. Field Controls signals to select pull-up or pull-down internal resistance strength. Select one out of next values for pad: TEST_MODE 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Control signal to enable internal pull-up/down resistors or pad keeper functionality. Select one out of next values for pad: TEST_MODE 0 **PUE_0_Keeper** — Keep the previous output value when the output driver is disabled. 1 **PUE_1_Pull** — Pull-up or pull-down (determined by PUS field).
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: TEST_MODE The pull/keeper function for a given pin is controlled by the PKE, PUE and PUS bits. The pull/keeper can be enabled by the pull/keep enable (PKE) bit. When the pull/keeper is enabled, the PUE (pull-up enable) bit selects either a pull-up/pull-down resistor on the output or a keeper device (keep the previous output value). When the pull/keeper is disabled, PUE and PUS have no functionality. 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: TEST_MODE If set to 1, the output driver drives only logic 0. The drain of the internal transistor is open. It means that logic 1 has to be driven by an external component. This option is essential if connection between the pad and an external component is bi-directional. If ODE = 0, then the output driver drives logic 1 and logic 0. 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled (Output is CMOS) 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled (Output is Open Drain)
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field

Table continues on the next page...

IOMUXC_SNVS_SW_PAD_CTL_PAD_TEST_MODE field descriptions (continued)

Field	Description
	SPEED is a selectable bit field that sets electrical characteristics of a pin in a given frequency range. This field provides additional 2-bit slew rate control. These options can either increase the output driver current in the higher frequency range, or reduce the switching noise in the lower frequency range. The operational frequency on GPIO pads is dependent on slew rate (SRE), speed (SPEED), and supply voltage (OVDD). See Operating Frequency table in the GPIO block guide for more details. 10 **Medium** — 100MHz
5–3 DSE	Drive Strength Field Select one out of next values for pad: TEST_MODE The drive strength enable (DSE) can be explained as series resistance between an ideal driver's output and its load. To achieve maximal transferred power, the impedance of the driver has to match the load impedance. Note: Typical values provided, please see GPIO spec for full impedance range. 000 **DSE_0** — HI-Z 001 **DSE_1** — Dual/Single voltage: 262/260 Ohm @ 1.8V, 247/157 Ohm @ 3.3V 010 **DSE_2** — Dual/Single voltage: 134/130 Ohm @ 1.8V, 126/78 Ohm @ 3.3V 011 **DSE_3** — Dual/Single voltage: 88/88 Ohm @ 1.8V, 84/53 Ohm @ 3.3V 100 **DSE_4** — Dual/Single voltage: 62/65 Ohm @ 1.8V, 57/39 Ohm @ 3.3V 101 **DSE_5** — Dual/Single voltage: 51/52 Ohm @ 1.8V, 47/32 Ohm @ 3.3V 110 **DSE_6** — Dual/Single voltage: 43/43 Ohm @ 1.8V, 40/26 Ohm @ 3.3V 111 **DSE_7** — Dual/Single voltage: 37/37 Ohm @ 1.8V, 34/23 Ohm @ 3.3V
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: TEST_MODE This bitfield controls how fast the pin toggles between the two logic states. Since rapidly changing states consume more power and generate spikes, it should be enabled only when necessary. The operational frequency on GPIO pads is dependent on slew rate (SRE), speed (SPEED), and supply voltage (OVDD). See Operating Frequency table in the GPIO block guide for more details. 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.4.5 SW_PAD_CTL_PAD_POR_B SW PAD Control Register (IOMUXC_SNVS_SW_PAD_CTL_PAD_POR_B)

SW_PAD_CTL Register

Address: 400A_8000h base + 10h offset = 400A_8010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	1	0	1	1	0	0	0	0	1	0	1	0	0	0	0	0

IOMUXC_SNVS_SW_PAD_CTL_PAD_POR_B field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: POR_B The hysteresis (HYS) bit controls whether a pin acts as a Schmitt trigger, which is a comparator remembering its last input state (hysteresis). 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled (CMOS input) 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled (Schmitt Trigger input)
15–14 PUS	Pull Up / Down Config. Field Controls signals to select pull-up or pull-down internal resistance strength. Select one out of next values for pad: POR_B 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Control signal to enable internal pull-up/down resistors or pad keeper functionality. Select one out of next values for pad: POR_B 0 **PUE_0_Keeper** — Keep the previous output value when the output driver is disabled. 1 **PUE_1_Pull** — Pull-up or pull-down (determined by PUS field).
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: POR_B

Table continues on the next page...

IOMUXC_SNVS_SW_PAD_CTL_PAD_POR_B field descriptions (continued)

Field	Description
	The pull/keeper function for a given pin is controlled by the PKE, PUE and PUS bits. The pull/keeper can be enabled by the pull/keep enable (PKE) bit. When the pull/keeper is enabled, the PUE (pull-up enable) bit selects either a pull-up/pull-down resistor on the output or a keeper device (keep the previous output value). When the pull/keeper is disabled, PUE and PUS have no functionality. 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: POR_B If set to 1, the output driver drives only logic 0. The drain of the internal transistor is open. It means that logic 1 has to be driven by an external component. This option is essential if connection between the pad and an external component is bi-directional. If ODE = 0, then the output driver drives logic 1 and logic 0. 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled (Output is CMOS) 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled (Output is Open Drain)
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field SPEED is a selectable bit field that sets electrical characteristics of a pin in a given frequency range. This field provides additional 2-bit slew rate control. These options can either increase the output driver current in the higher frequency range, or reduce the switching noise in the lower frequency range. The operational frequency on GPIO pads is dependent on slew rate (SRE), speed (SPEED), and supply voltage (OVDD). See Operating Frequency table in the GPIO block guide for more details. 10 **Medium** — 100MHz
5–3 DSE	Drive Strength Field Select one out of next values for pad: POR_B The drive strength enable (DSE) can be explained as series resistance between an ideal driver's output and its load. To achieve maximal transferred power, the impedance of the driver has to match the load impedance. Note: Typical values provided, please see GPIO spec for full impedance range. 000 **DSE_0** — HI-Z 001 **DSE_1** — Dual/Single voltage: 262/260 Ohm @ 1.8V, 247/157 Ohm @ 3.3V 010 **DSE_2** — Dual/Single voltage: 134/130 Ohm @ 1.8V, 126/78 Ohm @ 3.3V 011 **DSE_3** — Dual/Single voltage: 88/88 Ohm @ 1.8V, 84/53 Ohm @ 3.3V 100 **DSE_4** — Dual/Single voltage: 62/65 Ohm @ 1.8V, 57/39 Ohm @ 3.3V 101 **DSE_5** — Dual/Single voltage: 51/52 Ohm @ 1.8V, 47/32 Ohm @ 3.3V 110 **DSE_6** — Dual/Single voltage: 43/43 Ohm @ 1.8V, 40/26 Ohm @ 3.3V 111 **DSE_7** — Dual/Single voltage: 37/37 Ohm @ 1.8V, 34/23 Ohm @ 3.3V
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: POR_B

Table continues on the next page...

IOMUXC_SNVS_SW_PAD_CTL_PAD_POR_B field descriptions (continued)

Field	Description
	This bitfield controls how fast the pin toggles between the two logic states. Since rapidly changing states consume more power and generate spikes, it should be enabled only when necessary. The operational frequency on GPIO pads is dependent on slew rate (SRE), speed (SPEED), and supply voltage (OVDD). See Operating Frequency table in the GPIO block guide for more details. 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.4.6 SW_PAD_CTL_PAD_ONOFF SW PAD Control Register (IOMUXC_SNVS_SW_PAD_CTL_PAD_ONOFF)

SW_PAD_CTL Register

Address: 400A_8000h base + 14h offset = 400A_8014h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	1	0	1	1	0	0	0	0	1	0	1	0	0	0	0	0

IOMUXC_SNVS_SW_PAD_CTL_PAD_ONOFF field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: ONOFF The hysteresis (HYS) bit controls whether a pin acts as a Schmitt trigger, which is a comparator remembering its last input state (hysteresis). 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled (CMOS input) 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled (Schmitt Trigger input)
15–14 PUS	Pull Up / Down Config. Field Controls signals to select pull-up or pull-down internal resistance strength. Select one out of next values for pad: ONOFF 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up

Table continues on the next page...

IOMUXC_SNVS_SW_PAD_CTL_PAD_ONOFF field descriptions (continued)

Field	Description
	10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Control signal to enable internal pull-up/down resistors or pad keeper functionality. Select one out of next values for pad: ONOFF 0 **PUE_0_Keeper** — Keep the previous output value when the output driver is disabled. 1 **PUE_1_Pull** — Pull-up or pull-down (determined by PUS field).
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: ONOFF The pull/keeper function for a given pin is controlled by the PKE, PUE and PUS bits. The pull/keeper can be enabled by the pull/keep enable (PKE) bit. When the pull/keeper is enabled, the PUE (pull-up enable) bit selects either a pull-up/pull-down resistor on the output or a keeper device (keep the previous output value). When the pull/keeper is disabled, PUE and PUS have no functionality. 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: ONOFF If set to 1, the output driver drives only logic 0. The drain of the internal transistor is open. It means that logic 1 has to be driven by an external component. This option is essential if connection between the pad and an external component is bi-directional. If ODE = 0, then the output driver drives logic 1 and logic 0. 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled (Output is CMOS) 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled (Output is Open Drain)
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field SPEED is a selectable bit field that sets electrical characteristics of a pin in a given frequency range. This field provides additional 2-bit slew rate control. These options can either increase the output driver current in the higher frequency range, or reduce the switching noise in the lower frequency range. The operational frequency on GPIO pads is dependent on slew rate (SRE), speed (SPEED), and supply voltage (OVDD). See Operating Frequency table in the GPIO block guide for more details. 10 **Medium** — 100MHz
5–3 DSE	Drive Strength Field Select one out of next values for pad: ONOFF The drive strength enable (DSE) can be explained as series resistance between an ideal driver's output and its load. To achieve maximal transferred power, the impedance of the driver has to match the load impedance. Note: Typical values provided, please see GPIO spec for full impedance range. 000 **DSE_0** — HI-Z 001 **DSE_1** — Dual/Single voltage: 262/260 Ohm @ 1.8V, 247/157 Ohm @ 3.3V

Table continues on the next page...

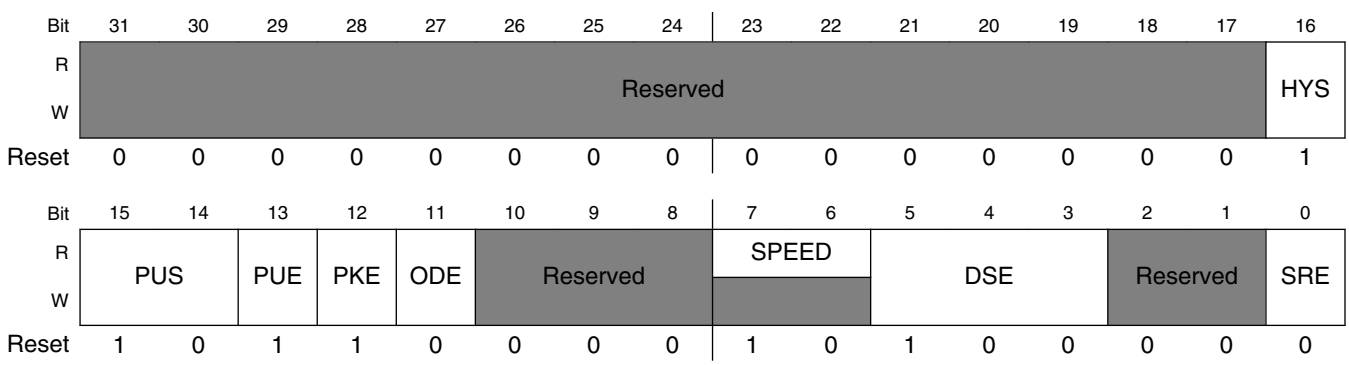
IOMUXC_SNVS_SW_PAD_CTL_PAD_ONOFF field descriptions (continued)

Field	Description
	010 **DSE_2** — Dual/Single voltage: 134/130 Ohm @ 1.8V, 126/78 Ohm @ 3.3V 011 **DSE_3** — Dual/Single voltage: 88/88 Ohm @ 1.8V, 84/53 Ohm @ 3.3V 100 **DSE_4** — Dual/Single voltage: 62/65 Ohm @ 1.8V, 57/39 Ohm @ 3.3V 101 **DSE_5** — Dual/Single voltage: 51/52 Ohm @ 1.8V, 47/32 Ohm @ 3.3V 110 **DSE_6** — Dual/Single voltage: 43/43 Ohm @ 1.8V, 40/26 Ohm @ 3.3V 111 **DSE_7** — Dual/Single voltage: 37/37 Ohm @ 1.8V, 34/23 Ohm @ 3.3V
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: ONOFF This bitfield controls how fast the pin toggles between the two logic states. Since rapidly changing states consume more power and generate spikes, it should be enabled only when necessary. The operational frequency on GPIO pads is dependent on slew rate (SRE), speed (SPEED), and supply voltage (OVDD). See Operating Frequency table in the GPIO block guide for more details. 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.4.7 SW_PAD_CTL_PAD_WAKEUP SW PAD Control Register (IOMUXC_SNVS_SW_PAD_CTL_PAD_WAKEUP)

SW_PAD_CTL Register

Address: 400A_8000h base + 18h offset = 400A_8018h



IOMUXC_SNVS_SW_PAD_CTL_PAD_WAKEUP field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: WAKEUP

Table continues on the next page...

IOMUXC_SNVS_SW_PAD_CTL_PAD_WAKEUP field descriptions (continued)

Field	Description
	The hysteresis (HYS) bit controls whether a pin acts as a Schmitt trigger, which is a comparator remembering its last input state (hysteresis). 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled (CMOS input) 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled (Schmitt Trigger input)
15–14 PUS	Pull Up / Down Config. Field Controls signals to select pull-up or pull-down internal resistance strength. Select one out of next values for pad: WAKEUP 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Control signal to enable internal pull-up/down resistors or pad keeper functionality. Select one out of next values for pad: WAKEUP 0 **PUE_0_Keeper** — Keep the previous output value when the output driver is disabled. 1 **PUE_1_Pull** — Pull-up or pull-down (determined by PUS field).
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: WAKEUP The pull/keeper function for a given pin is controlled by the PKE, PUE and PUS bits. The pull/keeper can be enabled by the pull/keep enable (PKE) bit. When the pull/keeper is enabled, the PUE (pull-up enable) bit selects either a pull-up/pull-down resistor on the output or a keeper device (keep the previous output value). When the pull/keeper is disabled, PUE and PUS have no functionality. 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: WAKEUP If set to 1, the output driver drives only logic 0. The drain of the internal transistor is open. It means that logic 1 has to be driven by an external component. This option is essential if connection between the pad and an external component is bi-directional. If ODE = 0, then the output driver drives logic 1 and logic 0. 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled (Output is CMOS) 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled (Output is Open Drain)
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field SPEED is a selectable bit field that sets electrical characteristics of a pin in a given frequency range. This field provides additional 2-bit slew rate control. These options can either increase the output driver current in the higher frequency range, or reduce the switching noise in the lower frequency range. The operational frequency on GPIO pads is dependent on slew rate (SRE), speed (SPEED), and supply voltage (OVDD). See Operating Frequency table in the GPIO block guide for more details.

Table continues on the next page...

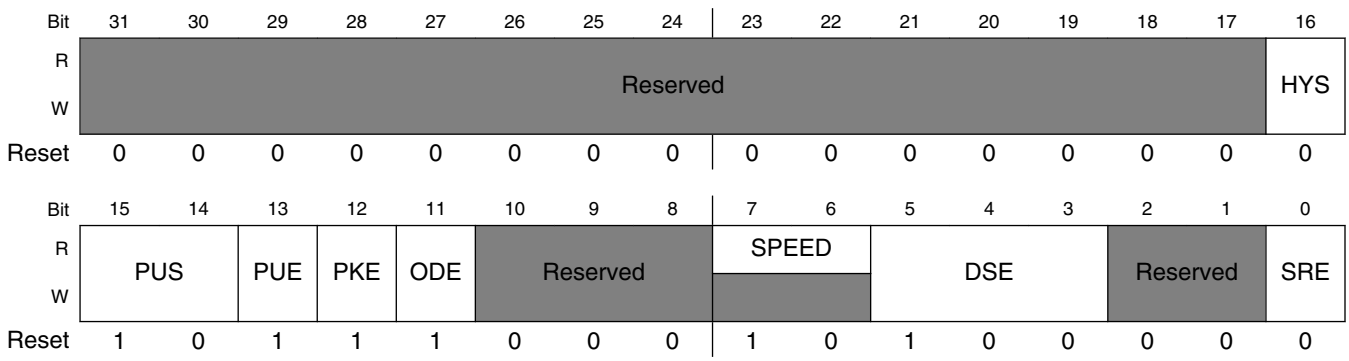
IOMUXC_SNVS_SW_PAD_CTL_PAD_WAKEUP field descriptions (continued)

Field	Description
	10 **Medium** — 100MHz
5–3 DSE	Drive Strength Field Select one out of next values for pad: WAKEUP The drive strength enable (DSE) can be explained as series resistance between an ideal driver's output and its load. To achieve maximal transferred power, the impedance of the driver has to match the load impedance. Note: Typical values provided, please see GPIO spec for full impedance range. 000 **DSE_0** — HI-Z 001 **DSE_1** — Dual/Single voltage: 262/260 Ohm @ 1.8V, 247/157 Ohm @ 3.3V 010 **DSE_2** — Dual/Single voltage: 134/130 Ohm @ 1.8V, 126/78 Ohm @ 3.3V 011 **DSE_3** — Dual/Single voltage: 88/88 Ohm @ 1.8V, 84/53 Ohm @ 3.3V 100 **DSE_4** — Dual/Single voltage: 62/65 Ohm @ 1.8V, 57/39 Ohm @ 3.3V 101 **DSE_5** — Dual/Single voltage: 51/52 Ohm @ 1.8V, 47/32 Ohm @ 3.3V 110 **DSE_6** — Dual/Single voltage: 43/43 Ohm @ 1.8V, 40/26 Ohm @ 3.3V 111 **DSE_7** — Dual/Single voltage: 37/37 Ohm @ 1.8V, 34/23 Ohm @ 3.3V
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: WAKEUP This bitfield controls how fast the pin toggles between the two logic states. Since rapidly changing states consume more power and generate spikes, it should be enabled only when necessary. The operational frequency on GPIO pads is dependent on slew rate (SRE), speed (SPEED), and supply voltage (OVDD). See Operating Frequency table in the GPIO block guide for more details. 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.4.8 SW_PAD_CTL_PAD_PMIC_ON_REQ SW PAD Control Register (IOMUXC_SNVS_SW_PAD_CTL_PAD_PMIC_ON_REQ)

SW_PAD_CTL Register

Address: 400A_8000h base + 1Ch offset = 400A_801Ch



IOMUXC_SNVS_SW_PAD_CTL_PAD_PMIC_ON_REQ field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: PMIC_ON_REQ The hysteresis (HYS) bit controls whether a pin acts as a Schmitt trigger, which is a comparator remembering its last input state (hysteresis). 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled (CMOS input) 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled (Schmitt Trigger input)
15–14 PUS	Pull Up / Down Config. Field Controls signals to select pull-up or pull-down internal resistance strength. Select one out of next values for pad: PMIC_ON_REQ 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Control signal to enable internal pull-up/down resistors or pad keeper functionality. Select one out of next values for pad: PMIC_ON_REQ 0 **PUE_0_Keeper** — Keep the previous output value when the output driver is disabled. 1 **PUE_1_Pull** — Pull-up or pull-down (determined by PUS field).

Table continues on the next page...

IOMUXC_SNVS_SW_PAD_CTL_PAD_PMIC_ON_REQ field descriptions (continued)

Field	Description
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: PMIC_ON_REQ The pull/keeper function for a given pin is controlled by the PKE, PUE and PUS bits. The pull/keeper can be enabled by the pull/keep enable (PKE) bit. When the pull/keeper is enabled, the PUE (pull-up enable) bit selects either a pull-up/pull-down resistor on the output or a keeper device (keep the previous output value). When the pull/keeper is disabled, PUE and PUS have no functionality. 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: PMIC_ON_REQ If set to 1, the output driver drives only logic 0. The drain of the internal transistor is open. It means that logic 1 has to be driven by an external component. This option is essential if connection between the pad and an external component is bi-directional. If ODE = 0, then the output driver drives logic 1 and logic 0. 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled (Output is CMOS) 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled (Output is Open Drain)
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field SPEED is a selectable bit field that sets electrical characteristics of a pin in a given frequency range. This field provides additional 2-bit slew rate control. These options can either increase the output driver current in the higher frequency range, or reduce the switching noise in the lower frequency range. The operational frequency on GPIO pads is dependent on slew rate (SRE), speed (SPEED), and supply voltage (OVDD). See Operating Frequency table in the GPIO block guide for more details. 10 **Medium** — 100MHz
5–3 DSE	Drive Strength Field Select one out of next values for pad: PMIC_ON_REQ The drive strength enable (DSE) can be explained as series resistance between an ideal driver's output and its load. To achieve maximal transferred power, the impedance of the driver has to match the load impedance. Note: Typical values provided, please see GPIO spec for full impedance range. 000 **DSE_0** — HI-Z 001 **DSE_1** — Dual/Single voltage: 262/260 Ohm @ 1.8V, 247/157 Ohm @ 3.3V 010 **DSE_2** — Dual/Single voltage: 134/130 Ohm @ 1.8V, 126/78 Ohm @ 3.3V 011 **DSE_3** — Dual/Single voltage: 88/88 Ohm @ 1.8V, 84/53 Ohm @ 3.3V 100 **DSE_4** — Dual/Single voltage: 62/65 Ohm @ 1.8V, 57/39 Ohm @ 3.3V 101 **DSE_5** — Dual/Single voltage: 51/52 Ohm @ 1.8V, 47/32 Ohm @ 3.3V 110 **DSE_6** — Dual/Single voltage: 43/43 Ohm @ 1.8V, 40/26 Ohm @ 3.3V 111 **DSE_7** — Dual/Single voltage: 37/37 Ohm @ 1.8V, 34/23 Ohm @ 3.3V
2–1 -	This field is reserved. Reserved

Table continues on the next page...

IOMUXC_SNVS_SW_PAD_CTL_PAD_PMIC_ON_REQ field descriptions (continued)

Field	Description
0 SRE	Slew Rate Field Select one out of next values for pad: PMIC_ON_REQ This bitfield controls how fast the pin toggles between the two logic states. Since rapidly changing states consume more power and generate spikes, it should be enabled only when necessary. The operational frequency on GPIO pads is dependent on slew rate (SRE), speed (SPEED), and supply voltage (OVDD). See Operating Frequency table in the GPIO block guide for more details. 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.4.9 SW_PAD_CTL_PAD_PMIC_STBY_REQ SW PAD Control Register (IOMUXC_SNVS_SW_PAD_CTL_PAD_PMIC_STBY_REQ)

SW_PAD_CTL Register

Address: 400A_8000h base + 20h offset = 400A_8020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	1	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0

IOMUXC_SNVS_SW_PAD_CTL_PAD_PMIC_STBY_REQ field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: PMIC_STBY_REQ The hysteresis (HYS) bit controls whether a pin acts as a Schmitt trigger, which is a comparator remembering its last input state (hysteresis). 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled (CMOS input) 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled (Schmitt Trigger input)
15–14 PUS	Pull Up / Down Config. Field Controls signals to select pull-up or pull-down internal resistance strength.

Table continues on the next page...

IOMUXC_SNVS_SW_PAD_CTL_PAD_PMIC_STBY_REQ field descriptions (continued)

Field	Description
	Select one out of next values for pad: PMIC_STBY_REQ 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Control signal to enable internal pull-up/down resistors or pad keeper functionality. Select one out of next values for pad: PMIC_STBY_REQ 0 **PUE_0_Keeper** — Keep the previous output value when the output driver is disabled. 1 **PUE_1_Pull** — Pull-up or pull-down (determined by PUS field).
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: PMIC_STBY_REQ The pull/keeper function for a given pin is controlled by the PKE, PUE and PUS bits. The pull/keeper can be enabled by the pull/keep enable (PKE) bit. When the pull/keeper is enabled, the PUE (pull-up enable) bit selects either a pull-up/pull-down resistor on the output or a keeper device (keep the previous output value). When the pull/keeper is disabled, PUE and PUS have no functionality. 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: PMIC_STBY_REQ If set to 1, the output driver drives only logic 0. The drain of the internal transistor is open. It means that logic 1 has to be driven by an external component. This option is essential if connection between the pad and an external component is bi-directional. If ODE = 0, then the output driver drives logic 1 and logic 0. 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled (Output is CMOS) 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled (Output is Open Drain)
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field SPEED is a selectable bit field that sets electrical characteristics of a pin in a given frequency range. This field provides additional 2-bit slew rate control. These options can either increase the output driver current in the higher frequency range, or reduce the switching noise in the lower frequency range. The operational frequency on GPIO pads is dependent on slew rate (SRE), speed (SPEED), and supply voltage (OVDD). See Operating Frequency table in the GPIO block guide for more details. 10 **Medium** — 100MHz
5–3 DSE	Drive Strength Field Select one out of next values for pad: PMIC_STBY_REQ The drive strength enable (DSE) can be explained as series resistance between an ideal driver's output and its load. To achieve maximal transferred power, the impedance of the driver has to match the load impedance.

Table continues on the next page...

IOMUXC_SNVS_SW_PAD_CTL_PAD_PMIC_STBY_REQ field descriptions (continued)

Field	Description
	Note: Typical values provided, please see GPIO spec for full impedance range. 000 **DSE_0** — HI-Z 001 **DSE_1** — Dual/Single voltage: 262/260 Ohm @ 1.8V, 247/157 Ohm @ 3.3V 010 **DSE_2** — Dual/Single voltage: 134/130 Ohm @ 1.8V, 126/78 Ohm @ 3.3V 011 **DSE_3** — Dual/Single voltage: 88/88 Ohm @ 1.8V, 84/53 Ohm @ 3.3V 100 **DSE_4** — Dual/Single voltage: 62/65 Ohm @ 1.8V, 57/39 Ohm @ 3.3V 101 **DSE_5** — Dual/Single voltage: 51/52 Ohm @ 1.8V, 47/32 Ohm @ 3.3V 110 **DSE_6** — Dual/Single voltage: 43/43 Ohm @ 1.8V, 40/26 Ohm @ 3.3V 111 **DSE_7** — Dual/Single voltage: 37/37 Ohm @ 1.8V, 34/23 Ohm @ 3.3V
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: PMIC_STBY_REQ This bitfield controls how fast the pin toggles between the two logic states. Since rapidly changing states consume more power and generate spikes, it should be enabled only when necessary. The operational frequency on GPIO pads is dependent on slew rate (SRE), speed (SPEED), and supply voltage (OVDD). See Operating Frequency table in the GPIO block guide for more details. 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.5 IOMUXC SNVS GPR Memory Map/Register Definition

IOMUXC_SNVS_GPR memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400A_4000	GPR0 General Purpose Register (IOMUXC_SNVS_GPR_GPR0)	32	R/W	0000_0000h	11.5.1/396
400A_4004	GPR1 General Purpose Register (IOMUXC_SNVS_GPR_GPR1)	32	R/W	0000_0000h	11.5.2/396
400A_4008	GPR2 General Purpose Register (IOMUXC_SNVS_GPR_GPR2)	32	R/W	0000_0000h	11.5.3/396
400A_400C	GPR3 General Purpose Register (IOMUXC_SNVS_GPR_GPR3)	32	R/W	0000_0000h	11.5.4/398

11.5.1 GPR0 General Purpose Register (IOMUXC_SNVS_GPR_GPR0)

GPR Register

Address: 400A_4000h base + 0h offset = 400A_4000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_SNVS_GPR_GPR0 field descriptions

Field	Description
-	This field is reserved. Reserved

11.5.2 GPR1 General Purpose Register (IOMUXC_SNVS_GPR_GPR1)

GPR Register

Address: 400A_4000h base + 4h offset = 400A_4004h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_SNVS_GPR_GPR1 field descriptions

Field	Description
-	This field is reserved. Reserved

11.5.3 GPR2 General Purpose Register (IOMUXC_SNVS_GPR_GPR2)

GPR Register

Address: 400A_4000h base + 8h offset = 400A_4008h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_SNVS_GPR_GPR2 field descriptions

Field	Description
-	This field is reserved. Reserved

11.5.4 GPR3 General Purpose Register (IOMUXC_SNVS_GPR_GPR3)

GPR Register

Address: 400A_4000h base + Ch offset = 400A_400Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved												DCDC_STS_DC_OK	DCDC_OVER_VOL	DCDC_OVER_CUR	DCDC_IN_LOW_VOL
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved												POR_PULL_TYPE		DCDC_STATUS_CAPT_CLR	LPSR_MODE_ENABLE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_SNVS_GPR_GPR3 field descriptions

Field	Description
31–20 -	This field is reserved. Reserved
19 DCDC_STS_DC_OK	DCDC status OK 0 DCDC is ramping up and not ready 1 DCDC is ready
18 DCDC_OVER_VOL	DCDC output over voltage alert 0 No over voltage detected 1 Over voltage detected
17 DCDC_OVER_CUR	DCDC output over current alert 0 No over current detected 1 Over current detected
16 DCDC_IN_LOW_VOL	DCDC_IN low voltage detect. 0 DCDC_IN is ok 1 DCDC_IN is too low
15–4 -	This field is reserved. Reserved
3–2 POR_PULL_TYPE	POR_B pad control 00 100 Ohm pull up enabled for POR_B always 01 Disable pull in SNVS mode, 100 Ohm pull up enabled otherwise 10 Disable pull of POR_B always 11 100 Ohm pull down enabled in SNVS mode, 100 Ohm pull up enabled otherwise
1 DCDC_STATUS_CAPT_CLR	DCDC captured status clear Write logic 1 to clear the 3 bits of DCDC captured status: DCDC_OVER_VOL, DCDC_OVER_CUR, and DCDC_IN_LOW_VOL. Then write 0 to re-enable the 3bits for status capture.
0 LPSR_MODE_ENABLE	Set to enable LPSR mode. 0 SNVS domain will reset when system reset happens 1 SNVS domain will only reset with SNVS POR

11.6 IOMUXC Memory Map/Register Definition

IOMUXC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
401F_8014	SW_MUX_CTL_PAD_GPIO_EMC_00 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_00)	32	R/W	0000_0005h	11.6.1/423
401F_8018	SW_MUX_CTL_PAD_GPIO_EMC_01 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_01)	32	R/W	0000_0005h	11.6.2/424

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
401F_801C	SW_MUX_CTL_PAD_GPIO_EMC_02 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_02)	32	R/W	0000_0005h	11.6.3/425
401F_8020	SW_MUX_CTL_PAD_GPIO_EMC_03 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_03)	32	R/W	0000_0005h	11.6.4/427
401F_8024	SW_MUX_CTL_PAD_GPIO_EMC_04 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_04)	32	R/W	0000_0005h	11.6.5/428
401F_8028	SW_MUX_CTL_PAD_GPIO_EMC_05 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_05)	32	R/W	0000_0005h	11.6.6/429
401F_802C	SW_MUX_CTL_PAD_GPIO_EMC_06 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_06)	32	R/W	0000_0005h	11.6.7/431
401F_8030	SW_MUX_CTL_PAD_GPIO_EMC_07 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_07)	32	R/W	0000_0005h	11.6.8/432
401F_8034	SW_MUX_CTL_PAD_GPIO_EMC_08 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_08)	32	R/W	0000_0005h	11.6.9/433
401F_8038	SW_MUX_CTL_PAD_GPIO_EMC_09 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_09)	32	R/W	0000_0005h	11.6.10/434
401F_803C	SW_MUX_CTL_PAD_GPIO_EMC_10 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_10)	32	R/W	0000_0005h	11.6.11/435
401F_8040	SW_MUX_CTL_PAD_GPIO_EMC_11 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_11)	32	R/W	0000_0005h	11.6.12/436
401F_8044	SW_MUX_CTL_PAD_GPIO_EMC_12 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_12)	32	R/W	0000_0005h	11.6.13/437
401F_8048	SW_MUX_CTL_PAD_GPIO_EMC_13 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_13)	32	R/W	0000_0005h	11.6.14/438
401F_804C	SW_MUX_CTL_PAD_GPIO_EMC_14 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_14)	32	R/W	0000_0005h	11.6.15/439
401F_8050	SW_MUX_CTL_PAD_GPIO_EMC_15 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_15)	32	R/W	0000_0005h	11.6.16/440
401F_8054	SW_MUX_CTL_PAD_GPIO_EMC_16 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_16)	32	R/W	0000_0005h	11.6.17/441
401F_8058	SW_MUX_CTL_PAD_GPIO_EMC_17 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_17)	32	R/W	0000_0005h	11.6.18/442
401F_805C	SW_MUX_CTL_PAD_GPIO_EMC_18 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_18)	32	R/W	0000_0005h	11.6.19/443
401F_8060	SW_MUX_CTL_PAD_GPIO_EMC_19 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_19)	32	R/W	0000_0005h	11.6.20/444
401F_8064	SW_MUX_CTL_PAD_GPIO_EMC_20 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_20)	32	R/W	0000_0005h	11.6.21/446
401F_8068	SW_MUX_CTL_PAD_GPIO_EMC_21 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_21)	32	R/W	0000_0005h	11.6.22/447
401F_806C	SW_MUX_CTL_PAD_GPIO_EMC_22 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_22)	32	R/W	0000_0005h	11.6.23/448
401F_8070	SW_MUX_CTL_PAD_GPIO_EMC_23 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_23)	32	R/W	0000_0005h	11.6.24/449

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
401F_8074	SW_MUX_CTL_PAD_GPIO_EMC_24 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_24)	32	R/W	0000_0005h	11.6.25/450
401F_8078	SW_MUX_CTL_PAD_GPIO_EMC_25 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_25)	32	R/W	0000_0005h	11.6.26/451
401F_807C	SW_MUX_CTL_PAD_GPIO_EMC_26 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_26)	32	R/W	0000_0005h	11.6.27/452
401F_8080	SW_MUX_CTL_PAD_GPIO_EMC_27 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_27)	32	R/W	0000_0005h	11.6.28/453
401F_8084	SW_MUX_CTL_PAD_GPIO_EMC_28 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_28)	32	R/W	0000_0005h	11.6.29/454
401F_8088	SW_MUX_CTL_PAD_GPIO_EMC_29 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_29)	32	R/W	0000_0005h	11.6.30/455
401F_808C	SW_MUX_CTL_PAD_GPIO_EMC_30 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_30)	32	R/W	0000_0005h	11.6.31/456
401F_8090	SW_MUX_CTL_PAD_GPIO_EMC_31 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_31) pin 29	32	R/W	0000_0005h	11.6.32/457
401F_8094	SW_MUX_CTL_PAD_GPIO_EMC_32 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_32) pin 28	32	R/W	0000_0005h	11.6.33/458
401F_8098	SW_MUX_CTL_PAD_GPIO_EMC_33 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_33)	32	R/W	0000_0005h	11.6.34/459
401F_809C	SW_MUX_CTL_PAD_GPIO_EMC_34 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_34)	32	R/W	0000_0005h	11.6.35/460
401F_80A0	SW_MUX_CTL_PAD_GPIO_EMC_35 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_35)	32	R/W	0000_0005h	11.6.36/461
401F_80A4	SW_MUX_CTL_PAD_GPIO_EMC_36 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_36) pin 31	32	R/W	0000_0005h	11.6.37/462
401F_80A8	SW_MUX_CTL_PAD_GPIO_EMC_37 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_37) pin 30	32	R/W	0000_0005h	11.6.38/463
401F_80AC	SW_MUX_CTL_PAD_GPIO_EMC_38 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_38)	32	R/W	0000_0005h	11.6.39/464
401F_80B0	SW_MUX_CTL_PAD_GPIO_EMC_39 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_39)	32	R/W	0000_0005h	11.6.40/465
401F_80B4	SW_MUX_CTL_PAD_GPIO_EMC_40 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_40)	32	R/W	0000_0005h	11.6.41/466
401F_80B8	SW_MUX_CTL_PAD_GPIO_EMC_41 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_41)	32	R/W	0000_0005h	11.6.42/467
401F_80BC	SW_MUX_CTL_PAD_GPIO_AD_B0_00 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_00)	32	R/W	0000_0005h	11.6.43/468
401F_80C0	SW_MUX_CTL_PAD_GPIO_AD_B0_01 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_01)	32	R/W	0000_0005h	11.6.44/469
401F_80C4	SW_MUX_CTL_PAD_GPIO_AD_B0_02 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_02) pin 1	32	R/W	0000_0005h	11.6.45/471

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
401F_80C8	SW_MUX_CTL_PAD_GPIO_AD_B0_03 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_03) pin	32	R/W	0000_0005h	11.6.46/472
401F_80CC	SW_MUX_CTL_PAD_GPIO_AD_B0_04 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_04)	32	R/W	0000_0000h	11.6.47/474
401F_80D0	SW_MUX_CTL_PAD_GPIO_AD_B0_05 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_05)	32	R/W	0000_0000h	11.6.48/475
401F_80D4	SW_MUX_CTL_PAD_GPIO_AD_B0_06 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_06)	32	R/W	0000_0000h	11.6.49/477
401F_80D8	SW_MUX_CTL_PAD_GPIO_AD_B0_07 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_07)	32	R/W	0000_0000h	11.6.50/478
401F_80DC	SW_MUX_CTL_PAD_GPIO_AD_B0_08 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_08)	32	R/W	0000_0000h	11.6.51/480
401F_80E0	SW_MUX_CTL_PAD_GPIO_AD_B0_09 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_09)	32	R/W	0000_0000h	11.6.52/481
401F_80E4	SW_MUX_CTL_PAD_GPIO_AD_B0_10 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_10)	32	R/W	0000_0000h	11.6.53/482
401F_80E8	SW_MUX_CTL_PAD_GPIO_AD_B0_11 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_11)	32	R/W	0000_0000h	11.6.54/483
401F_80EC	SW_MUX_CTL_PAD_GPIO_AD_B0_12 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_12) pin 24	32	R/W	0000_0005h	11.6.55/484
401F_80F0	SW_MUX_CTL_PAD_GPIO_AD_B0_13 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_13) pin 25	32	R/W	0000_0005h	11.6.56/485
401F_80F4	SW_MUX_CTL_PAD_GPIO_AD_B0_14 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_14)	32	R/W	0000_0005h	11.6.57/486
401F_80F8	SW_MUX_CTL_PAD_GPIO_AD_B0_15 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_15)	32	R/W	0000_0005h	11.6.58/487
401F_80FC	SW_MUX_CTL_PAD_GPIO_AD_B1_00 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B1_00) pin 19	32	R/W	0000_0005h	11.6.59/488
401F_8100	SW_MUX_CTL_PAD_GPIO_AD_B1_01 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B1_01) pin 18	32	R/W	0000_0005h	11.6.60/489

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
401F_8104	SW_MUX_CTL_PAD_GPIO_AD_B1_02 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B1_02) pin 14	32	R/W	0000_0005h	11.6.61/490
401F_8108	SW_MUX_CTL_PAD_GPIO_AD_B1_03 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B1_03) pin 15	32	R/W	0000_0005h	11.6.62/491
401F_810C	SW_MUX_CTL_PAD_GPIO_AD_B1_04 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B1_04) pin 40 (T4.1)	32	R/W	0000_0005h	11.6.63/492
401F_8110	SW_MUX_CTL_PAD_GPIO_AD_B1_05 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B1_05) pin 41 (T4.1)	32	R/W	0000_0005h	11.6.64/493
401F_8114	SW_MUX_CTL_PAD_GPIO_AD_B1_06 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B1_06) pin 17	32	R/W	0000_0005h	11.6.65/494
401F_8118	SW_MUX_CTL_PAD_GPIO_AD_B1_07 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B1_07) pin 16	32	R/W	0000_0005h	11.6.66/495
401F_811C	SW_MUX_CTL_PAD_GPIO_AD_B1_08 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B1_08) pin 22	32	R/W	0000_0005h	11.6.67/496
401F_8120	SW_MUX_CTL_PAD_GPIO_AD_B1_09 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B1_09) pin 23	32	R/W	0000_0005h	11.6.68/497
401F_8124	SW_MUX_CTL_PAD_GPIO_AD_B1_10 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B1_10) pin 20	32	R/W	0000_0005h	11.6.69/498
401F_8128	SW_MUX_CTL_PAD_GPIO_AD_B1_11 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B1_11) pin 21	32	R/W	0000_0005h	11.6.70/499
401F_812C	SW_MUX_CTL_PAD_GPIO_AD_B1_12 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B1_12) pin 38 (T4.1)	32	R/W	0000_0005h	11.6.71/500
401F_8130	SW_MUX_CTL_PAD_GPIO_AD_B1_13 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B1_13) pin 39 (T4.1)	32	R/W	0000_0005h	11.6.72/501
401F_8134	SW_MUX_CTL_PAD_GPIO_AD_B1_14 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B1_14) pin 26	32	R/W	0000_0005h	11.6.73/502
401F_8138	SW_MUX_CTL_PAD_GPIO_AD_B1_15 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B1_15) pin 27	32	R/W	0000_0005h	11.6.74/503
401F_813C	SW_MUX_CTL_PAD_GPIO_B0_00 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B0_00) pin 10	32	R/W	0000_0005h	11.6.75/504
401F_8140	SW_MUX_CTL_PAD_GPIO_B0_01 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B0_01) pin 12	32	R/W	0000_0005h	11.6.76/505

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
401F_8144	SW_MUX_CTL_PAD_GPIO_B0_02 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B0_02) pin 11	32	R/W	0000_0005h	11.6.77/506
401F_8148	SW_MUX_CTL_PAD_GPIO_B0_03 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B0_03) pin 13	32	R/W	0000_0005h	11.6.78/507
401F_814C	SW_MUX_CTL_PAD_GPIO_B0_04 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B0_04) pin 40 (MM)	32	R/W	0000_0005h	11.6.79/508
401F_8150	SW_MUX_CTL_PAD_GPIO_B0_05 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B0_05) pin 41 (MM)	32	R/W	0000_0005h	11.6.80/509
401F_8154	SW_MUX_CTL_PAD_GPIO_B0_06 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B0_06) pin 42 (MM)	32	R/W	0000_0005h	11.6.81/510
401F_8158	SW_MUX_CTL_PAD_GPIO_B0_07 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B0_07) pin 43 (MM)	32	R/W	0000_0005h	11.6.82/511
401F_815C	SW_MUX_CTL_PAD_GPIO_B0_08 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B0_08) pin 44 (MM)	32	R/W	0000_0005h	11.6.83/512
401F_8160	SW_MUX_CTL_PAD_GPIO_B0_09 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B0_09) pin 45 (MM)	32	R/W	0000_0005h	11.6.84/513
401F_8164	SW_MUX_CTL_PAD_GPIO_B0_10 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B0_10) pin 6	32	R/W	0000_0005h	11.6.85/514
401F_8168	SW_MUX_CTL_PAD_GPIO_B0_11 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B0_11) pin 9	32	R/W	0000_0005h	11.6.86/515
401F_816C	SW_MUX_CTL_PAD_GPIO_B0_12 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B0_12) pin 32	32	R/W	0000_0005h	11.6.87/516
401F_8170	SW_MUX_CTL_PAD_GPIO_B0_13 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B0_13)	32	R/W	0000_0005h	11.6.88/517
401F_8174	SW_MUX_CTL_PAD_GPIO_B0_14 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B0_14)	32	R/W	0000_0005h	11.6.89/518
401F_8178	SW_MUX_CTL_PAD_GPIO_B0_15 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B0_15)	32	R/W	0000_0005h	11.6.90/519
401F_817C	SW_MUX_CTL_PAD_GPIO_B1_00 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B1_00) pin 8	32	R/W	0000_0005h	11.6.91/520
401F_8180	SW_MUX_CTL_PAD_GPIO_B1_01 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B1_01) pin 7	32	R/W	0000_0005h	11.6.92/521
401F_8184	SW_MUX_CTL_PAD_GPIO_B1_02 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B1_02) pin 36 (T4.1)	32	R/W	0000_0005h	11.6.93/522
401F_8188	SW_MUX_CTL_PAD_GPIO_B1_03 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B1_03) pin 37 (T4.1)	32	R/W	0000_0005h	11.6.94/523
401F_818C	SW_MUX_CTL_PAD_GPIO_B1_04 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B1_04)	32	R/W	0000_0005h	11.6.95/524
401F_8190	SW_MUX_CTL_PAD_GPIO_B1_05 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B1_05)	32	R/W	0000_0005h	11.6.96/525
401F_8194	SW_MUX_CTL_PAD_GPIO_B1_06 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B1_06)	32	R/W	0000_0005h	11.6.97/526
401F_8198	SW_MUX_CTL_PAD_GPIO_B1_07 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B1_07)	32	R/W	0000_0005h	11.6.98/527

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
401F_819C	SW_MUX_CTL_PAD_GPIO_B1_08 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B1_08)	32	R/W	0000_0005h	11.6.99/528
401F_81A0	SW_MUX_CTL_PAD_GPIO_B1_09 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B1_09)	32	R/W	0000_0005h	11.6.100/ 529
401F_81A4	SW_MUX_CTL_PAD_GPIO_B1_10 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B1_10)	32	R/W	0000_0005h	11.6.101/ 530
401F_81A8	SW_MUX_CTL_PAD_GPIO_B1_11 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B1_11)	32	R/W	0000_0005h	11.6.102/ 531
401F_81AC	SW_MUX_CTL_PAD_GPIO_B1_12 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B1_12) pin 35 (T4.1)	32	R/W	0000_0005h	11.6.103/ 532
401F_81B0	SW_MUX_CTL_PAD_GPIO_B1_13 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B1_13) pin 34 (T4.1)	32	R/W	0000_0005h	11.6.104/ 533
401F_81B4	SW_MUX_CTL_PAD_GPIO_B1_14 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B1_14)	32	R/W	0000_0005h	11.6.105/ 534
401F_81B8	SW_MUX_CTL_PAD_GPIO_B1_15 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B1_15)	32	R/W	0000_0005h	11.6.106/ 535
401F_81BC	SW_MUX_CTL_PAD_GPIO_SD_B0_00 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B0_00)	32	R/W	0000_0005h	11.6.107/ 536
401F_81C0	SW_MUX_CTL_PAD_GPIO_SD_B0_01 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B0_01)	32	R/W	0000_0005h	11.6.108/ 537
401F_81C4	SW_MUX_CTL_PAD_GPIO_SD_B0_02 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B0_02)	32	R/W	0000_0005h	11.6.109/ 538
401F_81C8	SW_MUX_CTL_PAD_GPIO_SD_B0_03 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B0_03)	32	R/W	0000_0005h	11.6.110/ 539
401F_81CC	SW_MUX_CTL_PAD_GPIO_SD_B0_04 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B0_04)	32	R/W	0000_0005h	11.6.111/ 540
401F_81D0	SW_MUX_CTL_PAD_GPIO_SD_B0_05 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B0_05)	32	R/W	0000_0005h	11.6.112/ 541
401F_81D4	SW_MUX_CTL_PAD_GPIO_SD_B1_00 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B1_00)	32	R/W	0000_0005h	11.6.113/ 542
401F_81D8	SW_MUX_CTL_PAD_GPIO_SD_B1_01 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B1_01)	32	R/W	0000_0005h	11.6.114/ 543
401F_81DC	SW_MUX_CTL_PAD_GPIO_SD_B1_02 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B1_02)	32	R/W	0000_0005h	11.6.115/ 544
401F_81E0	SW_MUX_CTL_PAD_GPIO_SD_B1_03 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B1_03)	32	R/W	0000_0005h	11.6.116/ 545

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
401F_81E4	SW_MUX_CTL_PAD_GPIO_SD_B1_04 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B1_04)	32	R/W	0000_0005h	11.6.117/ 546
401F_81E8	SW_MUX_CTL_PAD_GPIO_SD_B1_05 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B1_05)	32	R/W	0000_0005h	11.6.118/ 547
401F_81EC	SW_MUX_CTL_PAD_GPIO_SD_B1_06 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B1_06)	32	R/W	0000_0005h	11.6.119/ 548
401F_81F0	SW_MUX_CTL_PAD_GPIO_SD_B1_07 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B1_07)	32	R/W	0000_0005h	11.6.120/ 549
401F_81F4	SW_MUX_CTL_PAD_GPIO_SD_B1_08 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B1_08)	32	R/W	0000_0005h	11.6.121/ 550
401F_81F8	SW_MUX_CTL_PAD_GPIO_SD_B1_09 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B1_09)	32	R/W	0000_0005h	11.6.122/ 551
401F_81FC	SW_MUX_CTL_PAD_GPIO_SD_B1_10 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B1_10)	32	R/W	0000_0005h	11.6.123/ 553
401F_8200	SW_MUX_CTL_PAD_GPIO_SD_B1_11 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B1_11)	32	R/W	0000_0005h	11.6.124/ 554
401F_8204	SW_PAD_CTL_PAD_GPIO_EMC_00 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_00)	32	R/W	0000_10B0h	11.6.125/ 555
401F_8208	SW_PAD_CTL_PAD_GPIO_EMC_01 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_01)	32	R/W	0000_10B0h	11.6.126/ 557
401F_820C	SW_PAD_CTL_PAD_GPIO_EMC_02 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_02)	32	R/W	0000_10B0h	11.6.127/ 559
401F_8210	SW_PAD_CTL_PAD_GPIO_EMC_03 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_03)	32	R/W	0000_10B0h	11.6.128/ 560
401F_8214	SW_PAD_CTL_PAD_GPIO_EMC_04 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_04) pin 2	32	R/W	0000_10B0h	11.6.129/ 562
401F_8218	SW_PAD_CTL_PAD_GPIO_EMC_05 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_05) pin 3	32	R/W	0000_10B0h	11.6.130/ 564
401F_821C	SW_PAD_CTL_PAD_GPIO_EMC_06 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_06) pin 4	32	R/W	0000_10B0h	11.6.131/ 565
401F_8220	SW_PAD_CTL_PAD_GPIO_EMC_07 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_07) pin 33	32	R/W	0000_10B0h	11.6.132/ 567
401F_8224	SW_PAD_CTL_PAD_GPIO_EMC_08 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_08) pin 5	32	R/W	0000_10B0h	11.6.133/ 569
401F_8228	SW_PAD_CTL_PAD_GPIO_EMC_09 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_09)	32	R/W	0000_10B0h	11.6.134/ 570
401F_822C	SW_PAD_CTL_PAD_GPIO_EMC_10 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_10)	32	R/W	0000_10B0h	11.6.135/ 572

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
401F_8230	SW_PAD_CTL_PAD_GPIO_EMC_11 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_11)	32	R/W	0000_10B0h	11.6.136/ 574
401F_8234	SW_PAD_CTL_PAD_GPIO_EMC_12 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_12)	32	R/W	0000_10B0h	11.6.137/ 575
401F_8238	SW_PAD_CTL_PAD_GPIO_EMC_13 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_13)	32	R/W	0000_10B0h	11.6.138/ 577
401F_823C	SW_PAD_CTL_PAD_GPIO_EMC_14 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_14)	32	R/W	0000_10B0h	11.6.139/ 579
401F_8240	SW_PAD_CTL_PAD_GPIO_EMC_15 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_15)	32	R/W	0000_10B0h	11.6.140/ 580
401F_8244	SW_PAD_CTL_PAD_GPIO_EMC_16 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_16)	32	R/W	0000_10B0h	11.6.141/ 582
401F_8248	SW_PAD_CTL_PAD_GPIO_EMC_17 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_17)	32	R/W	0000_10B0h	11.6.142/ 584
401F_824C	SW_PAD_CTL_PAD_GPIO_EMC_18 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_18)	32	R/W	0000_10B0h	11.6.143/ 585
401F_8250	SW_PAD_CTL_PAD_GPIO_EMC_19 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_19)	32	R/W	0000_10B0h	11.6.144/ 587
401F_8254	SW_PAD_CTL_PAD_GPIO_EMC_20 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_20)	32	R/W	0000_10B0h	11.6.145/ 589
401F_8258	SW_PAD_CTL_PAD_GPIO_EMC_21 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_21)	32	R/W	0000_10B0h	11.6.146/ 590
401F_825C	SW_PAD_CTL_PAD_GPIO_EMC_22 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_22)	32	R/W	0000_10B0h	11.6.147/ 592
401F_8260	SW_PAD_CTL_PAD_GPIO_EMC_23 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_23)	32	R/W	0000_10B0h	11.6.148/ 594
401F_8264	SW_PAD_CTL_PAD_GPIO_EMC_24 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_24)	32	R/W	0000_10B0h	11.6.149/ 595
401F_8268	SW_PAD_CTL_PAD_GPIO_EMC_25 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_25)	32	R/W	0000_10B0h	11.6.150/ 597
401F_826C	SW_PAD_CTL_PAD_GPIO_EMC_26 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_26)	32	R/W	0000_10B0h	11.6.151/ 599
401F_8270	SW_PAD_CTL_PAD_GPIO_EMC_27 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_27)	32	R/W	0000_30B0h	11.6.152/ 600
401F_8274	SW_PAD_CTL_PAD_GPIO_EMC_28 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_28)	32	R/W	0000_10B0h	11.6.153/ 602
401F_8278	SW_PAD_CTL_PAD_GPIO_EMC_29 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_29)	32	R/W	0000_10B0h	11.6.154/ 604
401F_827C	SW_PAD_CTL_PAD_GPIO_EMC_30 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_30)	32	R/W	0000_10B0h	11.6.155/ 605
401F_8280	SW_PAD_CTL_PAD_GPIO_EMC_31 SW PAD Control pin 29 Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_31)	32	R/W	0000_10B0h	11.6.156/ 607
401F_8284	SW_PAD_CTL_PAD_GPIO_EMC_32 SW PAD Control pin 28 Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_32)	32	R/W	0000_10B0h	11.6.157/ 609

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
401F_8288	SW_PAD_CTL_PAD_GPIO_EMC_33 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_33)	32	R/W	0000_10B0h	11.6.158/ 610
401F_828C	SW_PAD_CTL_PAD_GPIO_EMC_34 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_34)	32	R/W	0000_10B0h	11.6.159/ 612
401F_8290	SW_PAD_CTL_PAD_GPIO_EMC_35 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_35)	32	R/W	0000_10B0h	11.6.160/ 614
401F_8294	SW_PAD_CTL_PAD_GPIO_EMC_36 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_36) pin 31	32	R/W	0000_10B0h	11.6.161/ 615
401F_8298	SW_PAD_CTL_PAD_GPIO_EMC_37 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_37) pin 30	32	R/W	0000_10B0h	11.6.162/ 617
401F_829C	SW_PAD_CTL_PAD_GPIO_EMC_38 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_38)	32	R/W	0000_10B0h	11.6.163/ 619
401F_82A0	SW_PAD_CTL_PAD_GPIO_EMC_39 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_39)	32	R/W	0000_10B0h	11.6.164/ 620
401F_82A4	SW_PAD_CTL_PAD_GPIO_EMC_40 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_40)	32	R/W	0000_10B0h	11.6.165/ 622
401F_82A8	SW_PAD_CTL_PAD_GPIO_EMC_41 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_41)	32	R/W	0000_10B0h	11.6.166/ 624
401F_82AC	SW_PAD_CTL_PAD_GPIO_AD_B0_00 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_00)	32	R/W	0000_10B0h	11.6.167/ 626
401F_82B0	SW_PAD_CTL_PAD_GPIO_AD_B0_01 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_01)	32	R/W	0000_10B0h	11.6.168/ 628
401F_82B4	SW_PAD_CTL_PAD_GPIO_AD_B0_02 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_02) pin 1	32	R/W	0000_10B0h	11.6.169/ 630
401F_82B8	SW_PAD_CTL_PAD_GPIO_AD_B0_03 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_03) pin 0	32	R/W	0000_10B0h	11.6.170/ 632
401F_82BC	SW_PAD_CTL_PAD_GPIO_AD_B0_04 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_04)	32	R/W	0000_30B0h	11.6.171/ 634
401F_82C0	SW_PAD_CTL_PAD_GPIO_AD_B0_05 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_05)	32	R/W	0000_30B0h	11.6.172/ 636
401F_82C4	SW_PAD_CTL_PAD_GPIO_AD_B0_06 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_06)	32	R/W	0000_70A0h	11.6.173/ 638
401F_82C8	SW_PAD_CTL_PAD_GPIO_AD_B0_07 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_07)	32	R/W	0000_30A0h	11.6.174/ 640
401F_82CC	SW_PAD_CTL_PAD_GPIO_AD_B0_08 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_08)	32	R/W	0000_30A0h	11.6.175/ 642

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
401F_82D0	SW_PAD_CTL_PAD_GPIO_AD_B0_09 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_09)	32	R/W	0000_70A0h	11.6.176/ 644
401F_82D4	SW_PAD_CTL_PAD_GPIO_AD_B0_10 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_10)	32	R/W	0000_90B1h	11.6.177/ 646
401F_82D8	SW_PAD_CTL_PAD_GPIO_AD_B0_11 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_11)	32	R/W	0000_70A0h	11.6.178/ 648
401F_82DC	SW_PAD_CTL_PAD_GPIO_AD_B0_12 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_12) pin 24	32	R/W	0000_10B0h	11.6.179/ 650
401F_82E0	SW_PAD_CTL_PAD_GPIO_AD_B0_13 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_13) pin 25	32	R/W	0000_10B0h	11.6.180/ 652
401F_82E4	SW_PAD_CTL_PAD_GPIO_AD_B0_14 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_14)	32	R/W	0000_10B0h	11.6.181/ 654
401F_82E8	SW_PAD_CTL_PAD_GPIO_AD_B0_15 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_15)	32	R/W	0000_10B0h	11.6.182/ 656
401F_82EC	SW_PAD_CTL_PAD_GPIO_AD_B1_00 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_00) pin 19	32	R/W	0000_10B0h	11.6.183/ 658
401F_82F0	SW_PAD_CTL_PAD_GPIO_AD_B1_01 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_01) pin 18	32	R/W	0000_10B0h	11.6.184/ 660
401F_82F4	SW_PAD_CTL_PAD_GPIO_AD_B1_02 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_02) pin 14	32	R/W	0000_10B0h	11.6.185/ 662
401F_82F8	SW_PAD_CTL_PAD_GPIO_AD_B1_03 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_03) pin 15	32	R/W	0000_10B0h	11.6.186/ 664
401F_82FC	SW_PAD_CTL_PAD_GPIO_AD_B1_04 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_04) pin 40 (T4.1)	32	R/W	0000_10B0h	11.6.187/ 666
401F_8300	SW_PAD_CTL_PAD_GPIO_AD_B1_05 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_05) pin 41 (T4.1)	32	R/W	0000_10B0h	11.6.188/ 668
401F_8304	SW_PAD_CTL_PAD_GPIO_AD_B1_06 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_06) pin 17	32	R/W	0000_10B0h	11.6.189/ 670
401F_8308	SW_PAD_CTL_PAD_GPIO_AD_B1_07 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_07) pin 16	32	R/W	0000_10B0h	11.6.190/ 672

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
401F_830C	SW_PAD_CTL_PAD_GPIO_AD_B1_08 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_08) pin 22	32	R/W	0000_10B0h	11.6.191/ 674
401F_8310	SW_PAD_CTL_PAD_GPIO_AD_B1_09 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_09) pin 23	32	R/W	0000_10B0h	11.6.192/ 676
401F_8314	SW_PAD_CTL_PAD_GPIO_AD_B1_10 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_10) pin 20	32	R/W	0000_10B0h	11.6.193/ 678
401F_8318	SW_PAD_CTL_PAD_GPIO_AD_B1_11 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_11) pin 21	32	R/W	0000_10B0h	11.6.194/ 680
401F_831C	SW_PAD_CTL_PAD_GPIO_AD_B1_12 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_12) pin 38 (T4.1)	32	R/W	0000_10B0h	11.6.195/ 682
401F_8320	SW_PAD_CTL_PAD_GPIO_AD_B1_13 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_13) pin 39 (T4.1)	32	R/W	0000_10B0h	11.6.196/ 684
401F_8324	SW_PAD_CTL_PAD_GPIO_AD_B1_14 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_14) pin 26	32	R/W	0000_10B0h	11.6.197/ 686
401F_8328	SW_PAD_CTL_PAD_GPIO_AD_B1_15 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_15) pin 27	32	R/W	0000_10B0h	11.6.198/ 688
401F_832C	SW_PAD_CTL_PAD_GPIO_B0_00 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_00) pin 10	32	R/W	0000_10B0h	11.6.199/ 689
401F_8330	SW_PAD_CTL_PAD_GPIO_B0_01 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_01) pin 12	32	R/W	0000_10B0h	11.6.200/ 691
401F_8334	SW_PAD_CTL_PAD_GPIO_B0_02 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_02) pin 11	32	R/W	0000_10B0h	11.6.201/ 693
401F_8338	SW_PAD_CTL_PAD_GPIO_B0_03 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_03) pin 13	32	R/W	0000_10B0h	11.6.202/ 694
401F_833C	SW_PAD_CTL_PAD_GPIO_B0_04 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_04) pin 40 (MM)	32	R/W	0000_10B0h	11.6.203/ 696
401F_8340	SW_PAD_CTL_PAD_GPIO_B0_05 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_05) pin 41 (MM)	32	R/W	0000_10B0h	11.6.204/ 698
401F_8344	SW_PAD_CTL_PAD_GPIO_B0_06 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_06) pin 42 (MM)	32	R/W	0000_10B0h	11.6.205/ 699
401F_8348	SW_PAD_CTL_PAD_GPIO_B0_07 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_07) pin 43 (MM)	32	R/W	0000_10B0h	11.6.206/ 701
401F_834C	SW_PAD_CTL_PAD_GPIO_B0_08 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_08) pin 44 (MM)	32	R/W	0000_10B0h	11.6.207/ 703
401F_8350	SW_PAD_CTL_PAD_GPIO_B0_09 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_09) pin 45 (MM)	32	R/W	0000_10B0h	11.6.208/ 704
401F_8354	SW_PAD_CTL_PAD_GPIO_B0_10 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_10) pin 6	32	R/W	0000_10B0h	11.6.209/ 706

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
401F_8358	SW_PAD_CTL_PAD_GPIO_B0_11 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_11) pin 9	32	R/W	0000_10B0h	11.6.210/ 708
401F_835C	SW_PAD_CTL_PAD_GPIO_B0_12 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_12) pin 32	32	R/W	0000_10B0h	11.6.211/ 709
401F_8360	SW_PAD_CTL_PAD_GPIO_B0_13 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_13)	32	R/W	0000_10B0h	11.6.212/ 711
401F_8364	SW_PAD_CTL_PAD_GPIO_B0_14 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_14)	32	R/W	0000_10B0h	11.6.213/ 713
401F_8368	SW_PAD_CTL_PAD_GPIO_B0_15 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_15)	32	R/W	0000_10B0h	11.6.214/ 714
401F_836C	SW_PAD_CTL_PAD_GPIO_B1_00 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_00) pin 8	32	R/W	0000_10B0h	11.6.215/ 716
401F_8370	SW_PAD_CTL_PAD_GPIO_B1_01 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_01) pin 7	32	R/W	0000_10B0h	11.6.216/ 718
401F_8374	SW_PAD_CTL_PAD_GPIO_B1_02 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_02) pin 36 (T4.1)	32	R/W	0000_10B0h	11.6.217/ 719
401F_8378	SW_PAD_CTL_PAD_GPIO_B1_03 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_03) pin 37 (T4.1)	32	R/W	0000_10B0h	11.6.218/ 721
401F_837C	SW_PAD_CTL_PAD_GPIO_B1_04 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_04)	32	R/W	0000_10B0h	11.6.219/ 723
401F_8380	SW_PAD_CTL_PAD_GPIO_B1_05 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_05)	32	R/W	0000_10B0h	11.6.220/ 724
401F_8384	SW_PAD_CTL_PAD_GPIO_B1_06 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_06)	32	R/W	0000_10B0h	11.6.221/ 726
401F_8388	SW_PAD_CTL_PAD_GPIO_B1_07 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_07)	32	R/W	0000_10B0h	11.6.222/ 728
401F_838C	SW_PAD_CTL_PAD_GPIO_B1_08 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_08)	32	R/W	0000_10B0h	11.6.223/ 729
401F_8390	SW_PAD_CTL_PAD_GPIO_B1_09 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_09)	32	R/W	0000_10B0h	11.6.224/ 731
401F_8394	SW_PAD_CTL_PAD_GPIO_B1_10 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_10)	32	R/W	0000_10B0h	11.6.225/ 733
401F_8398	SW_PAD_CTL_PAD_GPIO_B1_11 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_11)	32	R/W	0000_10B0h	11.6.226/ 734
401F_839C	SW_PAD_CTL_PAD_GPIO_B1_12 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_12) pin 35 (T4.1)	32	R/W	0000_10B0h	11.6.227/ 736
401F_83A0	SW_PAD_CTL_PAD_GPIO_B1_13 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_13) pin 34 (T4.1)	32	R/W	0000_10B0h	11.6.228/ 738
401F_83A4	SW_PAD_CTL_PAD_GPIO_B1_14 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_14)	32	R/W	0000_10B0h	11.6.229/ 739
401F_83A8	SW_PAD_CTL_PAD_GPIO_B1_15 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_15)	32	R/W	0000_10B0h	11.6.230/ 741
401F_83AC	SW_PAD_CTL_PAD_GPIO_SD_B0_00 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B0_00)	32	R/W	0000_10B0h	11.6.231/ 743

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
401F_83B0	SW_PAD_CTL_PAD_GPIO_SD_B0_01 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B0_01)	32	R/W	0000_10B0h	11.6.232/ 745
401F_83B4	SW_PAD_CTL_PAD_GPIO_SD_B0_02 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B0_02)	32	R/W	0000_10B0h	11.6.233/ 747
401F_83B8	SW_PAD_CTL_PAD_GPIO_SD_B0_03 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B0_03)	32	R/W	0000_10B0h	11.6.234/ 749
401F_83BC	SW_PAD_CTL_PAD_GPIO_SD_B0_04 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B0_04)	32	R/W	0000_10B0h	11.6.235/ 751
401F_83C0	SW_PAD_CTL_PAD_GPIO_SD_B0_05 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B0_05)	32	R/W	0000_10B0h	11.6.236/ 753
401F_83C4	SW_PAD_CTL_PAD_GPIO_SD_B1_00 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B1_00)	32	R/W	0000_10B0h	11.6.237/ 755
401F_83C8	SW_PAD_CTL_PAD_GPIO_SD_B1_01 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B1_01)	32	R/W	0000_10B0h	11.6.238/ 757
401F_83CC	SW_PAD_CTL_PAD_GPIO_SD_B1_02 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B1_02)	32	R/W	0000_10B0h	11.6.239/ 759
401F_83D0	SW_PAD_CTL_PAD_GPIO_SD_B1_03 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B1_03)	32	R/W	0000_10B0h	11.6.240/ 761
401F_83D4	SW_PAD_CTL_PAD_GPIO_SD_B1_04 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B1_04)	32	R/W	0000_10B0h	11.6.241/ 763
401F_83D8	SW_PAD_CTL_PAD_GPIO_SD_B1_05 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B1_05)	32	R/W	0000_10B0h	11.6.242/ 765
401F_83DC	SW_PAD_CTL_PAD_GPIO_SD_B1_06 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B1_06)	32	R/W	0000_10B0h	11.6.243/ 767
401F_83E0	SW_PAD_CTL_PAD_GPIO_SD_B1_07 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B1_07)	32	R/W	0000_10B0h	11.6.244/ 769
401F_83E4	SW_PAD_CTL_PAD_GPIO_SD_B1_08 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B1_08)	32	R/W	0000_10B0h	11.6.245/ 771
401F_83E8	SW_PAD_CTL_PAD_GPIO_SD_B1_09 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B1_09)	32	R/W	0000_10B0h	11.6.246/ 773

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
401F_83EC	SW_PAD_CTL_PAD_GPIO_SD_B1_10 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B1_10)	32	R/W	0000_10B0h	11.6.247/ 775
401F_83F0	SW_PAD_CTL_PAD_GPIO_SD_B1_11 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B1_11)	32	R/W	0000_10B0h	11.6.248/ 777
401F_83F4	ANATOP_USB_OTG1_ID_SELECT_INPUT DAISY Register (IOMUXC_ANATOP_USB_OTG1_ID_SELECT_INPUT)	32	R/W	0000_0000h	11.6.249/ 779
401F_83F8	ANATOP_USB_OTG2_ID_SELECT_INPUT DAISY Register (IOMUXC_ANATOP_USB_OTG2_ID_SELECT_INPUT)	32	R/W	0000_0000h	11.6.250/ 780
401F_83FC	CCM_PMIC_READY_SELECT_INPUT DAISY Register (IOMUXC_CCM_PMIC_READY_SELECT_INPUT)	32	R/W	0000_0000h	11.6.251/ 780
401F_8400	CSI_DATA02_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA02_SELECT_INPUT)	32	R/W	0000_0000h	11.6.252/ 781
401F_8404	CSI_DATA03_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA03_SELECT_INPUT)	32	R/W	0000_0000h	11.6.253/ 782
401F_8408	CSI_DATA04_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA04_SELECT_INPUT)	32	R/W	0000_0000h	11.6.254/ 783
401F_840C	CSI_DATA05_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA05_SELECT_INPUT)	32	R/W	0000_0000h	11.6.255/ 784
401F_8410	CSI_DATA06_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA06_SELECT_INPUT)	32	R/W	0000_0000h	11.6.256/ 785
401F_8414	CSI_DATA07_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA07_SELECT_INPUT)	32	R/W	0000_0000h	11.6.257/ 786
401F_8418	CSI_DATA08_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA08_SELECT_INPUT)	32	R/W	0000_0000h	11.6.258/ 787
401F_841C	CSI_DATA09_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA09_SELECT_INPUT)	32	R/W	0000_0000h	11.6.259/ 788
401F_8420	CSI_HSYNC_SELECT_INPUT DAISY Register (IOMUXC_CSI_HSYNC_SELECT_INPUT)	32	R/W	0000_0000h	11.6.260/ 788
401F_8424	CSI_PIXCLK_SELECT_INPUT DAISY Register (IOMUXC_CSI_PIXCLK_SELECT_INPUT)	32	R/W	0000_0000h	11.6.261/ 789
401F_8428	CSI_VSYNC_SELECT_INPUT DAISY Register (IOMUXC_CSI_VSYNC_SELECT_INPUT)	32	R/W	0000_0000h	11.6.262/ 790
401F_842C	ENET_IPG_CLK_RMII_SELECT_INPUT DAISY Register (IOMUXC_ENET_IPG_CLK_RMII_SELECT_INPUT)	32	R/W	0000_0000h	11.6.263/ 791
401F_8430	ENET_MDIO_SELECT_INPUT DAISY Register (IOMUXC_ENET_MDIO_SELECT_INPUT)	32	R/W	0000_0000h	11.6.264/ 791
401F_8434	ENET0_RXDATA_SELECT_INPUT DAISY Register (IOMUXC_ENET0_RXDATA_SELECT_INPUT)	32	R/W	0000_0000h	11.6.265/ 792
401F_8438	ENET1_RXDATA_SELECT_INPUT DAISY Register (IOMUXC_ENET1_RXDATA_SELECT_INPUT)	32	R/W	0000_0000h	11.6.266/ 793
401F_843C	ENET_RXEN_SELECT_INPUT DAISY Register (IOMUXC_ENET_RXEN_SELECT_INPUT)	32	R/W	0000_0000h	11.6.267/ 794

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
401F_8440	ENET_RXERR_SELECT_INPUT DAISY Register (IOMUXC_ENET_RXERR_SELECT_INPUT)	32	R/W	0000_0000h	11.6.268/ 795
401F_8444	ENET0_TIMER_SELECT_INPUT DAISY Register (IOMUXC_ENET0_TIMER_SELECT_INPUT)	32	R/W	0000_0000h	11.6.269/ 795
401F_8448	ENET_TXCLK_SELECT_INPUT DAISY Register (IOMUXC_ENET_TXCLK_SELECT_INPUT)	32	R/W	0000_0000h	11.6.270/ 796
401F_844C	FLEXCAN1_RX_SELECT_INPUT DAISY Register (IOMUXC_FLEXCAN1_RX_SELECT_INPUT)	32	R/W	0000_0000h	11.6.271/ 797
401F_8450	FLEXCAN2_RX_SELECT_INPUT DAISY Register (IOMUXC_FLEXCAN2_RX_SELECT_INPUT)	32	R/W	0000_0000h	11.6.272/ 797
401F_8454	FLEXPWM1_PWMA3_SELECT_INPUT DAISY Register (IOMUXC_FLEXPWM1_PWMA3_SELECT_INPUT)	32	R/W	0000_0000h	11.6.273/ 798
401F_8458	FLEXPWM1_PWMA0_SELECT_INPUT DAISY Register (IOMUXC_FLEXPWM1_PWMA0_SELECT_INPUT)	32	R/W	0000_0000h	11.6.274/ 799
401F_845C	FLEXPWM1_PWMA1_SELECT_INPUT DAISY Register (IOMUXC_FLEXPWM1_PWMA1_SELECT_INPUT)	32	R/W	0000_0000h	11.6.275/ 800
401F_8460	FLEXPWM1_PWMA2_SELECT_INPUT DAISY Register (IOMUXC_FLEXPWM1_PWMA2_SELECT_INPUT)	32	R/W	0000_0000h	11.6.276/ 801
401F_8464	FLEXPWM1_PWMB3_SELECT_INPUT DAISY Register (IOMUXC_FLEXPWM1_PWMB3_SELECT_INPUT)	32	R/W	0000_0000h	11.6.277/ 801
401F_8468	FLEXPWM1_PWMB0_SELECT_INPUT DAISY Register (IOMUXC_FLEXPWM1_PWMB0_SELECT_INPUT)	32	R/W	0000_0000h	11.6.278/ 802
401F_846C	FLEXPWM1_PWMB1_SELECT_INPUT DAISY Register (IOMUXC_FLEXPWM1_PWMB1_SELECT_INPUT)	32	R/W	0000_0000h	11.6.279/ 803
401F_8470	FLEXPWM1_PWMB2_SELECT_INPUT DAISY Register (IOMUXC_FLEXPWM1_PWMB2_SELECT_INPUT)	32	R/W	0000_0000h	11.6.280/ 804
401F_8474	FLEXPWM2_PWMA3_SELECT_INPUT DAISY Register (IOMUXC_FLEXPWM2_PWMA3_SELECT_INPUT)	32	R/W	0000_0000h	11.6.281/ 804
401F_8478	FLEXPWM2_PWMA0_SELECT_INPUT DAISY Register (IOMUXC_FLEXPWM2_PWMA0_SELECT_INPUT)	32	R/W	0000_0000h	11.6.282/ 805
401F_847C	FLEXPWM2_PWMA1_SELECT_INPUT DAISY Register (IOMUXC_FLEXPWM2_PWMA1_SELECT_INPUT)	32	R/W	0000_0000h	11.6.283/ 806
401F_8480	FLEXPWM2_PWMA2_SELECT_INPUT DAISY Register (IOMUXC_FLEXPWM2_PWMA2_SELECT_INPUT)	32	R/W	0000_0000h	11.6.284/ 807
401F_8484	FLEXPWM2_PWMB3_SELECT_INPUT DAISY Register (IOMUXC_FLEXPWM2_PWMB3_SELECT_INPUT)	32	R/W	0000_0000h	11.6.285/ 807
401F_8488	FLEXPWM2_PWMB0_SELECT_INPUT DAISY Register (IOMUXC_FLEXPWM2_PWMB0_SELECT_INPUT)	32	R/W	0000_0000h	11.6.286/ 808
401F_848C	FLEXPWM2_PWMB1_SELECT_INPUT DAISY Register (IOMUXC_FLEXPWM2_PWMB1_SELECT_INPUT)	32	R/W	0000_0000h	11.6.287/ 809
401F_8490	FLEXPWM2_PWMB2_SELECT_INPUT DAISY Register (IOMUXC_FLEXPWM2_PWMB2_SELECT_INPUT)	32	R/W	0000_0000h	11.6.288/ 810
401F_8494	FLEXPWM4_PWMA0_SELECT_INPUT DAISY Register (IOMUXC_FLEXPWM4_PWMA0_SELECT_INPUT)	32	R/W	0000_0000h	11.6.289/ 811

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
401F_8498	FLEXPWM4_PWMA1_SELECT_INPUT DAISY Register (IOMUXC_FLEXPWM4_PWMA1_SELECT_INPUT)	32	R/W	0000_0000h	11.6.290/ 812
401F_849C	FLEXPWM4_PWMA2_SELECT_INPUT DAISY Register (IOMUXC_FLEXPWM4_PWMA2_SELECT_INPUT)	32	R/W	0000_0000h	11.6.291/ 813
401F_84A0	FLEXPWM4_PWMA3_SELECT_INPUT DAISY Register (IOMUXC_FLEXPWM4_PWMA3_SELECT_INPUT)	32	R/W	0000_0000h	11.6.292/ 814
401F_84A4	FLEXSPIA_DQS_SELECT_INPUT DAISY Register (IOMUXC_FLEXSPIA_DQS_SELECT_INPUT)	32	R/W	0000_0000h	11.6.293/ 815
401F_84A8	FLEXSPIA_DATA0_SELECT_INPUT DAISY Register (IOMUXC_FLEXSPIA_DATA0_SELECT_INPUT)	32	R/W	0000_0000h	11.6.294/ 816
401F_84AC	FLEXSPIA_DATA1_SELECT_INPUT DAISY Register (IOMUXC_FLEXSPIA_DATA1_SELECT_INPUT)	32	R/W	0000_0000h	11.6.295/ 817
401F_84B0	FLEXSPIA_DATA2_SELECT_INPUT DAISY Register (IOMUXC_FLEXSPIA_DATA2_SELECT_INPUT)	32	R/W	0000_0000h	11.6.296/ 818
401F_84B4	FLEXSPIA_DATA3_SELECT_INPUT DAISY Register (IOMUXC_FLEXSPIA_DATA3_SELECT_INPUT)	32	R/W	0000_0000h	11.6.297/ 819
401F_84B8	FLEXSPIB_DATA0_SELECT_INPUT DAISY Register (IOMUXC_FLEXSPIB_DATA0_SELECT_INPUT)	32	R/W	0000_0000h	11.6.298/ 820
401F_84BC	FLEXSPIB_DATA1_SELECT_INPUT DAISY Register (IOMUXC_FLEXSPIB_DATA1_SELECT_INPUT)	32	R/W	0000_0000h	11.6.299/ 821
401F_84C0	FLEXSPIB_DATA2_SELECT_INPUT DAISY Register (IOMUXC_FLEXSPIB_DATA2_SELECT_INPUT)	32	R/W	0000_0000h	11.6.300/ 822
401F_84C4	FLEXSPIB_DATA3_SELECT_INPUT DAISY Register (IOMUXC_FLEXSPIB_DATA3_SELECT_INPUT)	32	R/W	0000_0000h	11.6.301/ 823
401F_84C8	FLEXSPIA_SCK_SELECT_INPUT DAISY Register (IOMUXC_FLEXSPIA_SCK_SELECT_INPUT)	32	R/W	0000_0000h	11.6.302/ 824
401F_84CC	LPI2C1_SCL_SELECT_INPUT DAISY Register (IOMUXC_LPI2C1_SCL_SELECT_INPUT)	32	R/W	0000_0000h	11.6.303/ 825
401F_84D0	LPI2C1_SDA_SELECT_INPUT DAISY Register (IOMUXC_LPI2C1_SDA_SELECT_INPUT)	32	R/W	0000_0000h	11.6.304/ 826
401F_84D4	LPI2C2_SCL_SELECT_INPUT DAISY Register (IOMUXC_LPI2C2_SCL_SELECT_INPUT)	32	R/W	0000_0000h	11.6.305/ 827
401F_84D8	LPI2C2_SDA_SELECT_INPUT DAISY Register (IOMUXC_LPI2C2_SDA_SELECT_INPUT)	32	R/W	0000_0000h	11.6.306/ 828
401F_84DC	LPI2C3_SCL_SELECT_INPUT DAISY Register (IOMUXC_LPI2C3_SCL_SELECT_INPUT)	32	R/W	0000_0000h	11.6.307/ 828
401F_84E0	LPI2C3_SDA_SELECT_INPUT DAISY Register (IOMUXC_LPI2C3_SDA_SELECT_INPUT)	32	R/W	0000_0000h	11.6.308/ 829
401F_84E4	LPI2C4_SCL_SELECT_INPUT DAISY Register (IOMUXC_LPI2C4_SCL_SELECT_INPUT)	32	R/W	0000_0000h	11.6.309/ 830
401F_84E8	LPI2C4_SDA_SELECT_INPUT DAISY Register (IOMUXC_LPI2C4_SDA_SELECT_INPUT)	32	R/W	0000_0000h	11.6.310/ 831
401F_84EC	LPSPI1_PCS0_SELECT_INPUT DAISY Register (IOMUXC_LPSPI1_PCS0_SELECT_INPUT)	32	R/W	0000_0000h	11.6.311/ 832

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
401F_84F0	LPSPI1_SCK_SELECT_INPUT DAISY Register (IOMUXC_LPSPI1_SCK_SELECT_INPUT)	32	R/W	0000_0000h	11.6.312/ 833
401F_84F4	LPSPI1_SDI_SELECT_INPUT DAISY Register (IOMUXC_LPSPI1_SDI_SELECT_INPUT)	32	R/W	0000_0000h	11.6.313/ 834
401F_84F8	LPSPI1_SDO_SELECT_INPUT DAISY Register (IOMUXC_LPSPI1_SDO_SELECT_INPUT)	32	R/W	0000_0000h	11.6.314/ 835
401F_84FC	LPSPI2_PCS0_SELECT_INPUT DAISY Register (IOMUXC_LPSPI2_PCS0_SELECT_INPUT)	32	R/W	0000_0000h	11.6.315/ 836
401F_8500	LPSPI2_SCK_SELECT_INPUT DAISY Register (IOMUXC_LPSPI2_SCK_SELECT_INPUT)	32	R/W	0000_0000h	11.6.316/ 837
401F_8504	LPSPI2_SDI_SELECT_INPUT DAISY Register (IOMUXC_LPSPI2_SDI_SELECT_INPUT)	32	R/W	0000_0000h	11.6.317/ 838
401F_8508	LPSPI2_SDO_SELECT_INPUT DAISY Register (IOMUXC_LPSPI2_SDO_SELECT_INPUT)	32	R/W	0000_0000h	11.6.318/ 839
401F_850C	LPSPI3_PCS0_SELECT_INPUT DAISY Register (IOMUXC_LPSPI3_PCS0_SELECT_INPUT)	32	R/W	0000_0000h	11.6.319/ 840
401F_8510	LPSPI3_SCK_SELECT_INPUT DAISY Register (IOMUXC_LPSPI3_SCK_SELECT_INPUT)	32	R/W	0000_0000h	11.6.320/ 841
401F_8514	LPSPI3_SDI_SELECT_INPUT DAISY Register (IOMUXC_LPSPI3_SDI_SELECT_INPUT)	32	R/W	0000_0000h	11.6.321/ 842
401F_8518	LPSPI3_SDO_SELECT_INPUT DAISY Register (IOMUXC_LPSPI3_SDO_SELECT_INPUT)	32	R/W	0000_0000h	11.6.322/ 843
401F_851C	LPSPI4_PCS0_SELECT_INPUT DAISY Register (IOMUXC_LPSPI4_PCS0_SELECT_INPUT)	32	R/W	0000_0000h	11.6.323/ 844
401F_8520	LPSPI4_SCK_SELECT_INPUT DAISY Register (IOMUXC_LPSPI4_SCK_SELECT_INPUT)	32	R/W	0000_0000h	11.6.324/ 845
401F_8524	LPSPI4_SDI_SELECT_INPUT DAISY Register (IOMUXC_LPSPI4_SDI_SELECT_INPUT)	32	R/W	0000_0000h	11.6.325/ 846
401F_8528	LPSPI4_SDO_SELECT_INPUT DAISY Register (IOMUXC_LPSPI4_SDO_SELECT_INPUT)	32	R/W	0000_0000h	11.6.326/ 847
401F_852C	LPUART2_RX_SELECT_INPUT DAISY Register (IOMUXC_LPUART2_RX_SELECT_INPUT)	32	R/W	0000_0000h	11.6.327/ 848
401F_8530	LPUART2_TX_SELECT_INPUT DAISY Register (IOMUXC_LPUART2_TX_SELECT_INPUT)	32	R/W	0000_0000h	11.6.328/ 849
401F_8534	LPUART3_CTS_B_SELECT_INPUT DAISY Register (IOMUXC_LPUART3_CTS_B_SELECT_INPUT)	32	R/W	0000_0000h	11.6.329/ 850
401F_8538	LPUART3_RX_SELECT_INPUT DAISY Register (IOMUXC_LPUART3_RX_SELECT_INPUT)	32	R/W	0000_0000h	11.6.330/ 850
401F_853C	LPUART3_TX_SELECT_INPUT DAISY Register (IOMUXC_LPUART3_TX_SELECT_INPUT)	32	R/W	0000_0000h	11.6.331/ 851
401F_8540	LPUART4_RX_SELECT_INPUT DAISY Register (IOMUXC_LPUART4_RX_SELECT_INPUT)	32	R/W	0000_0000h	11.6.332/ 852
401F_8544	LPUART4_TX_SELECT_INPUT DAISY Register (IOMUXC_LPUART4_TX_SELECT_INPUT)	32	R/W	0000_0000h	11.6.333/ 852

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
401F_8548	LPUART5_RX_SELECT_INPUT DAISY Register (IOMUXC_LPUART5_RX_SELECT_INPUT)	32	R/W	0000_0000h	11.6.334/ 853
401F_854C	LPUART5_TX_SELECT_INPUT DAISY Register (IOMUXC_LPUART5_TX_SELECT_INPUT)	32	R/W	0000_0000h	11.6.335/ 854
401F_8550	LPUART6_RX_SELECT_INPUT DAISY Register (IOMUXC_LPUART6_RX_SELECT_INPUT)	32	R/W	0000_0000h	11.6.336/ 855
401F_8554	LPUART6_TX_SELECT_INPUT DAISY Register (IOMUXC_LPUART6_TX_SELECT_INPUT)	32	R/W	0000_0000h	11.6.337/ 856
401F_8558	LPUART7_RX_SELECT_INPUT DAISY Register (IOMUXC_LPUART7_RX_SELECT_INPUT)	32	R/W	0000_0000h	11.6.338/ 857
401F_855C	LPUART7_TX_SELECT_INPUT DAISY Register (IOMUXC_LPUART7_TX_SELECT_INPUT)	32	R/W	0000_0000h	11.6.339/ 858
401F_8560	LPUART8_RX_SELECT_INPUT DAISY Register (IOMUXC_LPUART8_RX_SELECT_INPUT)	32	R/W	0000_0000h	11.6.340/ 858
401F_8564	LPUART8_TX_SELECT_INPUT DAISY Register (IOMUXC_LPUART8_TX_SELECT_INPUT)	32	R/W	0000_0000h	11.6.341/ 859
401F_8568	NMI_GLUE_NMI_SELECT_INPUT DAISY Register (IOMUXC_NMI_SELECT_INPUT)	32	R/W	0000_0000h	11.6.342/ 860
401F_856C	QTIMER2_TIMER0_SELECT_INPUT DAISY Register (IOMUXC_QTIMER2_TIMER0_SELECT_INPUT)	32	R/W	0000_0000h	11.6.343/ 861
401F_8570	QTIMER2_TIMER1_SELECT_INPUT DAISY Register (IOMUXC_QTIMER2_TIMER1_SELECT_INPUT)	32	R/W	0000_0000h	11.6.344/ 862
401F_8574	QTIMER2_TIMER2_SELECT_INPUT DAISY Register (IOMUXC_QTIMER2_TIMER2_SELECT_INPUT)	32	R/W	0000_0000h	11.6.345/ 863
401F_8578	QTIMER2_TIMER3_SELECT_INPUT DAISY Register (IOMUXC_QTIMER2_TIMER3_SELECT_INPUT)	32	R/W	0000_0000h	11.6.346/ 864
401F_857C	QTIMER3_TIMER0_SELECT_INPUT DAISY Register (IOMUXC_QTIMER3_TIMER0_SELECT_INPUT)	32	R/W	0000_0000h	11.6.347/ 864
401F_8580	QTIMER3_TIMER1_SELECT_INPUT DAISY Register (IOMUXC_QTIMER3_TIMER1_SELECT_INPUT)	32	R/W	0000_0000h	11.6.348/ 865
401F_8584	QTIMER3_TIMER2_SELECT_INPUT DAISY Register (IOMUXC_QTIMER3_TIMER2_SELECT_INPUT)	32	R/W	0000_0000h	11.6.349/ 866
401F_8588	QTIMER3_TIMER3_SELECT_INPUT DAISY Register (IOMUXC_QTIMER3_TIMER3_SELECT_INPUT)	32	R/W	0000_0000h	11.6.350/ 866
401F_858C	SAI1_MCLK2_SELECT_INPUT DAISY Register (IOMUXC_SAI1_MCLK2_SELECT_INPUT)	32	R/W	0000_0000h	11.6.351/ 867
401F_8590	SAI1_RX_BCLK_SELECT_INPUT DAISY Register (IOMUXC_SAI1_RX_BCLK_SELECT_INPUT)	32	R/W	0000_0000h	11.6.352/ 868
401F_8594	SAI1_RX_DATA0_SELECT_INPUT DAISY Register (IOMUXC_SAI1_RX_DATA0_SELECT_INPUT)	32	R/W	0000_0000h	11.6.353/ 868
401F_8598	SAI1_RX_DATA1_SELECT_INPUT DAISY Register (IOMUXC_SAI1_RX_DATA1_SELECT_INPUT)	32	R/W	0000_0000h	11.6.354/ 869
401F_859C	SAI1_RX_DATA2_SELECT_INPUT DAISY Register (IOMUXC_SAI1_RX_DATA2_SELECT_INPUT)	32	R/W	0000_0000h	11.6.355/ 870

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
401F_85A0	SAI1_RX_DATA3_SELECT_INPUT DAISY Register (IOMUXC_SAI1_RX_DATA3_SELECT_INPUT)	32	R/W	0000_0000h	11.6.356/ 871
401F_85A4	SAI1_RX_SYNC_SELECT_INPUT DAISY Register (IOMUXC_SAI1_RX_SYNC_SELECT_INPUT)	32	R/W	0000_0000h	11.6.357/ 871
401F_85A8	SAI1_TX_BCLK_SELECT_INPUT DAISY Register (IOMUXC_SAI1_TX_BCLK_SELECT_INPUT)	32	R/W	0000_0000h	11.6.358/ 872
401F_85AC	SAI1_TX_SYNC_SELECT_INPUT DAISY Register (IOMUXC_SAI1_TX_SYNC_SELECT_INPUT)	32	R/W	0000_0000h	11.6.359/ 873
401F_85B0	SAI2_MCLK2_SELECT_INPUT DAISY Register (IOMUXC_SAI2_MCLK2_SELECT_INPUT)	32	R/W	0000_0000h	11.6.360/ 874
401F_85B4	SAI2_RX_BCLK_SELECT_INPUT DAISY Register (IOMUXC_SAI2_RX_BCLK_SELECT_INPUT)	32	R/W	0000_0000h	11.6.361/ 875
401F_85B8	SAI2_RX_DATA0_SELECT_INPUT DAISY Register (IOMUXC_SAI2_RX_DATA0_SELECT_INPUT)	32	R/W	0000_0000h	11.6.362/ 876
401F_85BC	SAI2_RX_SYNC_SELECT_INPUT DAISY Register (IOMUXC_SAI2_RX_SYNC_SELECT_INPUT)	32	R/W	0000_0000h	11.6.363/ 877
401F_85C0	SAI2_TX_BCLK_SELECT_INPUT DAISY Register (IOMUXC_SAI2_TX_BCLK_SELECT_INPUT)	32	R/W	0000_0000h	11.6.364/ 878
401F_85C4	SAI2_TX_SYNC_SELECT_INPUT DAISY Register (IOMUXC_SAI2_TX_SYNC_SELECT_INPUT)	32	R/W	0000_0000h	11.6.365/ 879
401F_85C8	SPDIF_IN_SELECT_INPUT DAISY Register (IOMUXC_SPDIF_IN_SELECT_INPUT)	32	R/W	0000_0000h	11.6.366/ 880
401F_85CC	USB_OTG2_OC_SELECT_INPUT DAISY Register (IOMUXC_USB_OTG2_OC_SELECT_INPUT)	32	R/W	0000_0000h	11.6.367/ 881
401F_85D0	USB_OTG1_OC_SELECT_INPUT DAISY Register (IOMUXC_USB_OTG1_OC_SELECT_INPUT)	32	R/W	0000_0000h	11.6.368/ 882
401F_85D4	USDHC1_CD_B_SELECT_INPUT DAISY Register (IOMUXC_USDHC1_CD_B_SELECT_INPUT)	32	R/W	0000_0000h	11.6.369/ 882
401F_85D8	USDHC1_WP_SELECT_INPUT DAISY Register (IOMUXC_USDHC1_WP_SELECT_INPUT)	32	R/W	0000_0000h	11.6.370/ 883
401F_85DC	USDHC2_CLK_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_CLK_SELECT_INPUT)	32	R/W	0000_0000h	11.6.371/ 884
401F_85E0	USDHC2_CD_B_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_CD_B_SELECT_INPUT)	32	R/W	0000_0000h	11.6.372/ 885
401F_85E4	USDHC2_CMD_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_CMD_SELECT_INPUT)	32	R/W	0000_0000h	11.6.373/ 886
401F_85E8	USDHC2_DATA0_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_DATA0_SELECT_INPUT)	32	R/W	0000_0000h	11.6.374/ 887
401F_85EC	USDHC2_DATA1_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_DATA1_SELECT_INPUT)	32	R/W	0000_0000h	11.6.375/ 888
401F_85F0	USDHC2_DATA2_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_DATA2_SELECT_INPUT)	32	R/W	0000_0000h	11.6.376/ 889
401F_85F4	USDHC2_DATA3_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_DATA3_SELECT_INPUT)	32	R/W	0000_0000h	11.6.377/ 890

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
401F_85F8	USDHC2_DATA4_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_DATA4_SELECT_INPUT)	32	R/W	0000_0000h	11.6.378/ 891
401F_85FC	USDHC2_DATA5_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_DATA5_SELECT_INPUT)	32	R/W	0000_0000h	11.6.379/ 892
401F_8600	USDHC2_DATA6_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_DATA6_SELECT_INPUT)	32	R/W	0000_0000h	11.6.380/ 893
401F_8604	USDHC2_DATA7_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_DATA7_SELECT_INPUT)	32	R/W	0000_0000h	11.6.381/ 894
401F_8608	USDHC2_WP_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_WP_SELECT_INPUT)	32	R/W	0000_0000h	11.6.382/ 895
401F_860C	XBAR1_IN02_SELECT_INPUT DAISY Register (IOMUXC_XBAR1_IN02_SELECT_INPUT)	32	R/W	0000_0000h	11.6.383/ 896
401F_8610	XBAR1_IN03_SELECT_INPUT DAISY Register (IOMUXC_XBAR1_IN03_SELECT_INPUT)	32	R/W	0000_0000h	11.6.384/ 897
401F_8614	XBAR1_IN04_SELECT_INPUT DAISY Register (IOMUXC_XBAR1_IN04_SELECT_INPUT)	32	R/W	0000_0000h	11.6.385/ 898
401F_8618	XBAR1_IN05_SELECT_INPUT DAISY Register (IOMUXC_XBAR1_IN05_SELECT_INPUT)	32	R/W	0000_0000h	11.6.386/ 899
401F_861C	XBAR1_IN06_SELECT_INPUT DAISY Register (IOMUXC_XBAR1_IN06_SELECT_INPUT)	32	R/W	0000_0000h	11.6.387/ 900
401F_8620	XBAR1_IN07_SELECT_INPUT DAISY Register (IOMUXC_XBAR1_IN07_SELECT_INPUT)	32	R/W	0000_0000h	11.6.388/ 901
401F_8624	XBAR1_IN08_SELECT_INPUT DAISY Register (IOMUXC_XBAR1_IN08_SELECT_INPUT)	32	R/W	0000_0000h	11.6.389/ 902
401F_8628	XBAR1_IN09_SELECT_INPUT DAISY Register (IOMUXC_XBAR1_IN09_SELECT_INPUT)	32	R/W	0000_0000h	11.6.390/ 903
401F_862C	XBAR1_IN17_SELECT_INPUT DAISY Register (IOMUXC_XBAR1_IN17_SELECT_INPUT)	32	R/W	0000_0000h	11.6.391/ 903
401F_8630	XBAR1_IN18_SELECT_INPUT DAISY Register (IOMUXC_XBAR1_IN18_SELECT_INPUT)	32	R/W	0000_0000h	11.6.392/ 904
401F_8634	XBAR1_IN20_SELECT_INPUT DAISY Register (IOMUXC_XBAR1_IN20_SELECT_INPUT)	32	R/W	0000_0000h	11.6.393/ 905
401F_8638	XBAR1_IN22_SELECT_INPUT DAISY Register (IOMUXC_XBAR1_IN22_SELECT_INPUT)	32	R/W	0000_0000h	11.6.394/ 906
401F_863C	XBAR1_IN23_SELECT_INPUT DAISY Register (IOMUXC_XBAR1_IN23_SELECT_INPUT)	32	R/W	0000_0000h	11.6.395/ 907
401F_8640	XBAR1_IN24_SELECT_INPUT DAISY Register (IOMUXC_XBAR1_IN24_SELECT_INPUT)	32	R/W	0000_0000h	11.6.396/ 908
401F_8644	XBAR1_IN14_SELECT_INPUT DAISY Register (IOMUXC_XBAR1_IN14_SELECT_INPUT)	32	R/W	0000_0000h	11.6.397/ 909
401F_8648	XBAR1_IN15_SELECT_INPUT DAISY Register (IOMUXC_XBAR1_IN15_SELECT_INPUT)	32	R/W	0000_0000h	11.6.398/ 910
401F_864C	XBAR1_IN16_SELECT_INPUT DAISY Register (IOMUXC_XBAR1_IN16_SELECT_INPUT)	32	R/W	0000_0000h	11.6.399/ 911

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
401F_8650	XBAR1_IN25_SELECT_INPUT DAISY Register (IOMUXC_XBAR1_IN25_SELECT_INPUT)	32	R/W	0000_0000h	11.6.400/ 912
401F_8654	XBAR1_IN19_SELECT_INPUT DAISY Register (IOMUXC_XBAR1_IN19_SELECT_INPUT)	32	R/W	0000_0000h	11.6.401/ 913
401F_8658	XBAR1_IN23_SELECT_INPUT DAISY Register (IOMUXC_XBAR1_IN21_SELECT_INPUT)	32	R/W	0000_0000h	11.6.402/ 914
401F_870C	ENET2_IPG_CLK_RMII_SELECT_INPUT DAISY Register (IOMUXC_ENET2_IPG_CLK_RMII_SELECT_INPUT)	32	R/W	0000_0000h	11.6.403/ 914
401F_8710	ENET2_IPP_IND_MAC0_MDIO_SELECT_INPUT DAISY Register (IOMUXC_ENET2_IPP_IND_MAC0_MDIO_SELECT_INPUT)	32	R/W	0000_0000h	11.6.404/ 915
401F_8714	ENET2_IPP_IND_MAC0_RXDATA_SELECT_INPUT_0 DAISY Register (IOMUXC_ENET2_IPP_IND_MAC0_RXDATA_SELECT_INPUT_0)	32	R/W	0000_0000h	11.6.405/ 916
401F_8718	ENET2_IPP_IND_MAC0_RXDATA_SELECT_INPUT_1 DAISY Register (IOMUXC_ENET2_IPP_IND_MAC0_RXDATA_SELECT_INPUT_1)	32	R/W	0000_0000h	11.6.406/ 916
401F_871C	ENET2_IPP_IND_MAC0_RXEN_SELECT_INPUT DAISY Register (IOMUXC_ENET2_IPP_IND_MAC0_RXEN_SELECT_INPUT)	32	R/W	0000_0000h	11.6.407/ 917
401F_8720	ENET2_IPP_IND_MAC0_RXERR_SELECT_INPUT DAISY Register (IOMUXC_ENET2_IPP_IND_MAC0_RXERR_SELECT_INPUT)	32	R/W	0000_0000h	11.6.408/ 918
401F_8724	ENET2_IPP_IND_MAC0_TIMER_SELECT_INPUT_0 DAISY Register (IOMUXC_ENET2_IPP_IND_MAC0_TIMER_SELECT_INPUT_0)	32	R/W	0000_0000h	11.6.409/ 919
401F_8728	ENET2_IPP_IND_MAC0_TXCLK_SELECT_INPUT DAISY Register (IOMUXC_ENET2_IPP_IND_MAC0_TXCLK_SELECT_INPUT)	32	R/W	0000_0000h	11.6.410/ 920
401F_872C	FLEXSPI2_IPP_IND_DQS_FA_SELECT_INPUT DAISY Register (IOMUXC_FLEXSPI2_IPP_IND_DQS_FA_SELECT_INPUT)	32	R/W	0000_0000h	11.6.411/ 920
401F_8730	FLEXSPI2_IPP_IND_IO_FA_BIT0_SELECT_INPUT DAISY Register (IOMUXC_FLEXSPI2_IPP_IND_IO_FA_BIT0_SELECT_INPUT)	32	R/W	0000_0000h	11.6.412/ 921
401F_8734	FLEXSPI2_IPP_IND_IO_FA_BIT1_SELECT_INPUT DAISY Register (IOMUXC_FLEXSPI2_IPP_IND_IO_FA_BIT1_SELECT_INPUT)	32	R/W	0000_0000h	11.6.413/ 922

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
401F_8738	FLEXSPI2_IPP_IND_IO_FA_BIT2_SELECT_INPUT DAISY Register (IOMUXC_FLEXSPI2_IPP_IND_IO_FA_BIT2_SELECT_INPUT)	32	R/W	0000_0000h	11.6.414/ 922
401F_873C	FLEXSPI2_IPP_IND_IO_FA_BIT3_SELECT_INPUT DAISY Register (IOMUXC_FLEXSPI2_IPP_IND_IO_FA_BIT3_SELECT_INPUT)	32	R/W	0000_0000h	11.6.415/ 923
401F_8740	FLEXSPI2_IPP_IND_IO_FB_BIT0_SELECT_INPUT DAISY Register (IOMUXC_FLEXSPI2_IPP_IND_IO_FB_BIT0_SELECT_INPUT)	32	R/W	0000_0000h	11.6.416/ 924
401F_8744	FLEXSPI2_IPP_IND_IO_FB_BIT1_SELECT_INPUT DAISY Register (IOMUXC_FLEXSPI2_IPP_IND_IO_FB_BIT1_SELECT_INPUT)	32	R/W	0000_0000h	11.6.417/ 925
401F_8748	FLEXSPI2_IPP_IND_IO_FB_BIT2_SELECT_INPUT DAISY Register (IOMUXC_FLEXSPI2_IPP_IND_IO_FB_BIT2_SELECT_INPUT)	32	R/W	0000_0000h	11.6.418/ 926
401F_874C	FLEXSPI2_IPP_IND_IO_FB_BIT3_SELECT_INPUT DAISY Register (IOMUXC_FLEXSPI2_IPP_IND_IO_FB_BIT3_SELECT_INPUT)	32	R/W	0000_0000h	11.6.419/ 927
401F_8750	FLEXSPI2_IPP_IND_SCK_FA_SELECT_INPUT DAISY Register (IOMUXC_FLEXSPI2_IPP_IND_SCK_FA_SELECT_INPUT)	32	R/W	0000_0000h	11.6.420/ 928
401F_8754	FLEXSPI2_IPP_IND_SCK_FB_SELECT_INPUT DAISY Register (IOMUXC_FLEXSPI2_IPP_IND_SCK_FB_SELECT_INPUT)	32	R/W	0000_0000h	11.6.421/ 929
401F_8758	GPT1_IPP_IND_CAPIN1_SELECT_INPUT DAISY Register (IOMUXC_GPT1_IPP_IND_CAPIN1_SELECT_INPUT)	32	R/W	0000_0000h	11.6.422/ 930
401F_875C	GPT1_IPP_IND_CAPIN2_SELECT_INPUT DAISY Register (IOMUXC_GPT1_IPP_IND_CAPIN2_SELECT_INPUT)	32	R/W	0000_0000h	11.6.423/ 931
401F_8760	GPT1_IPP_IND_CLKIN_SELECT_INPUT DAISY Register (IOMUXC_GPT1_IPP_IND_CLKIN_SELECT_INPUT)	32	R/W	0000_0000h	11.6.424/ 932
401F_8764	GPT2_IPP_IND_CAPIN1_SELECT_INPUT DAISY Register (IOMUXC_GPT2_IPP_IND_CAPIN1_SELECT_INPUT)	32	R/W	0000_0000h	11.6.425/ 933
401F_8768	GPT2_IPP_IND_CAPIN2_SELECT_INPUT DAISY Register (IOMUXC_GPT2_IPP_IND_CAPIN2_SELECT_INPUT)	32	R/W	0000_0000h	11.6.426/ 934
401F_876C	GPT2_IPP_IND_CLKIN_SELECT_INPUT DAISY Register (IOMUXC_GPT2_IPP_IND_CLKIN_SELECT_INPUT)	32	R/W	0000_0000h	11.6.427/ 935
401F_8770	SAI3_IPG_CLK_SAI_MCLK_SELECT_INPUT_2 DAISY Register (IOMUXC_SAI3_IPG_CLK_SAI_MCLK_SELECT_INPUT_2)	32	R/W	0000_0000h	11.6.428/ 936

Table continues on the next page...

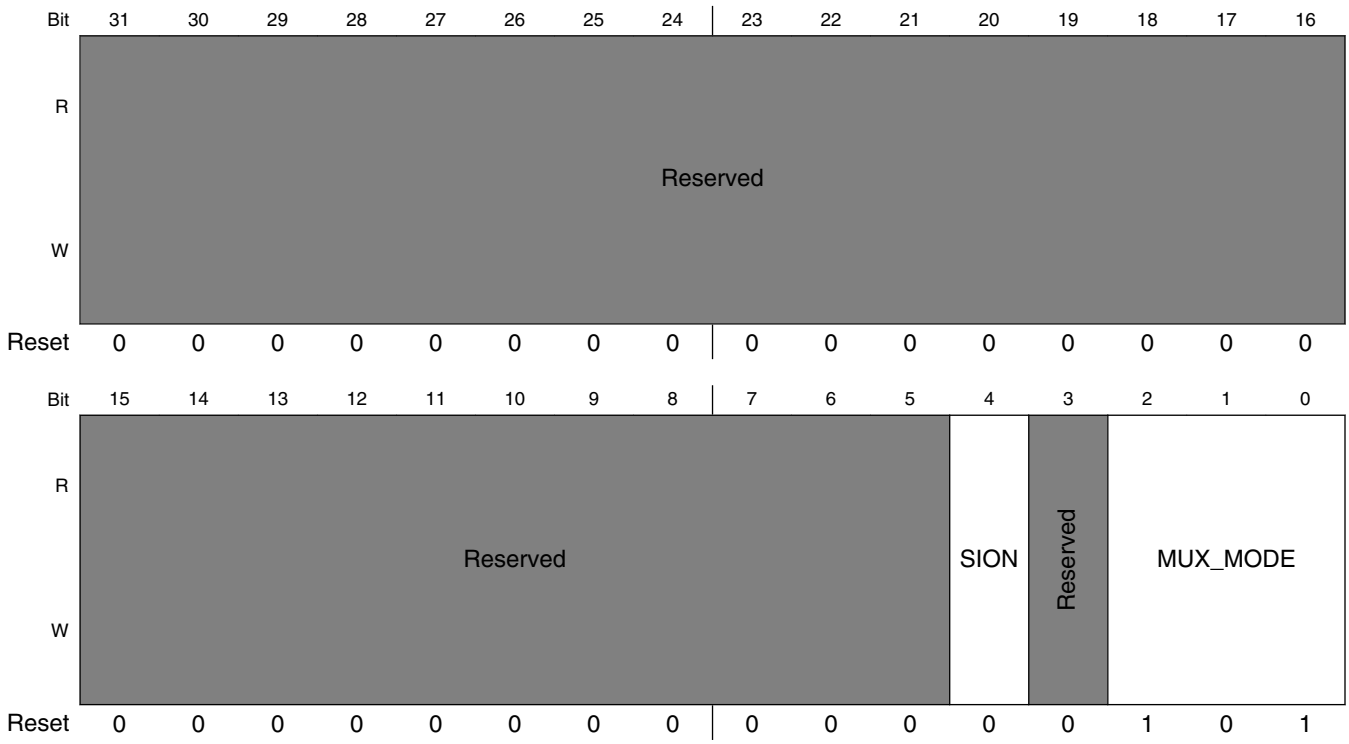
IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
401F_8774	SAI3_IPP_IND_SAI_RXBCLK_SELECT_INPUT DAISY Register (IOMUXC_SAI3_IPP_IND_SAI_RXBCLK_SELECT_INPUT)	32	R/W	0000_0000h	11.6.429/ 937
401F_8778	SAI3_IPP_IND_SAI_RXDATA_SELECT_INPUT_0 DAISY Register (IOMUXC_SAI3_IPP_IND_SAI_RXDATA_SELECT_INPUT_ 0)	32	R/W	0000_0000h	11.6.430/ 938
401F_877C	SAI3_IPP_IND_SAI_RXSYNC_SELECT_INPUT DAISY Register (IOMUXC_SAI3_IPP_IND_SAI_RXSYNC_SELECT_INPUT)	32	R/W	0000_0000h	11.6.431/ 939
401F_8780	SAI3_IPP_IND_SAI_TXBCLK_SELECT_INPUT DAISY Register (IOMUXC_SAI3_IPP_IND_SAI_TXBCLK_SELECT_INPUT)	32	R/W	0000_0000h	11.6.432/ 940
401F_8784	SAI3_IPP_IND_SAI_TXSYNC_SELECT_INPUT DAISY Register (IOMUXC_SAI3_IPP_IND_SAI_TXSYNC_SELECT_INPUT)	32	R/W	0000_0000h	11.6.433/ 941
401F_8788	SEMC_I_IPP_IND_DQS4_SELECT_INPUT DAISY Register (IOMUXC_SEMC_I_IPP_IND_DQS4_SELECT_INPUT)	32	R/W	0000_0000h	11.6.434/ 942
401F_878C	CANFD_IPP_IND_CANRX_SELECT_INPUT DAISY Register (IOMUXC_CANFD_IPP_IND_CANRX_SELECT_INPUT)	32	R/W	0000_0000h	11.6.435/ 942

11.6.1 SW_MUX_CTL_PAD_GPIO_EMC_00 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_00)

SW_MUX_CTL Register

Address: 401F_8000h base + 14h offset = 401F_8014h



IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_00 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_EMC_00 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_EMC_00. 000 **ALT0** — Select mux mode: ALT0 mux port: SEMC_DATA00 of instance: semc 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXPWM4_PWMA00 of instance: flexpwm4 010 **ALT2** — Select mux mode: ALT2 mux port: LPSPI2_SCK of instance: lpspi2

Table continues on the next page...

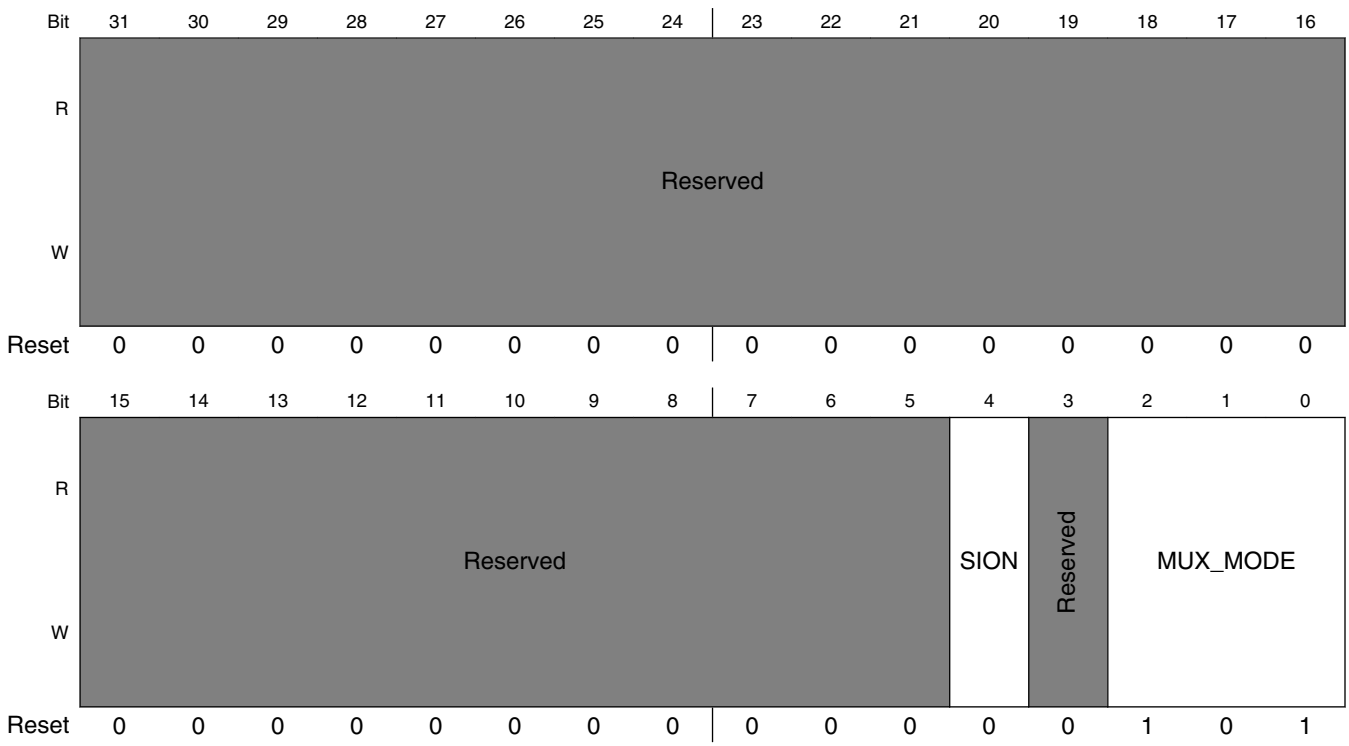
IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_00 field descriptions (continued)

Field	Description
	011 **ALT3** — Select mux mode: ALT3 mux port: XBAR1_XBAR_IN02 of instance: xbar1 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO1_FLEXIO00 of instance: flexio1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO00 of instance: gpio4

11.6.2 SW_MUX_CTL_PAD_GPIO_EMC_01 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_01)

SW_MUX_CTL Register

Address: 401F_8000h base + 18h offset = 401F_8018h



IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_01 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_EMC_01 0 **DISABLED** — Input Path is determined by functionality

Table continues on the next page...

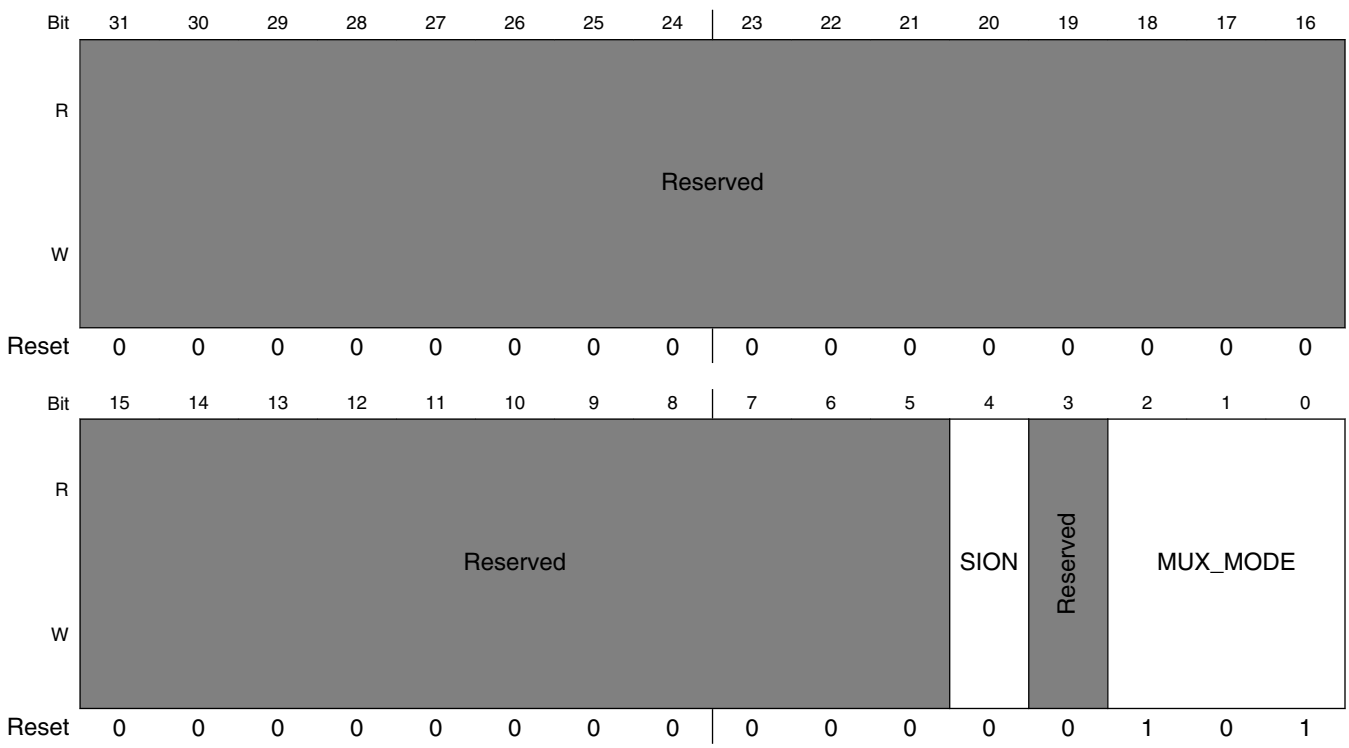
IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_01 field descriptions (continued)

Field	Description
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_EMC_01. 000 **ALT0** — Select mux mode: ALT0 mux port: SEMC_DATA01 of instance: semc 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXPWM4_PWMB00 of instance: flexpwm4 010 **ALT2** — Select mux mode: ALT2 mux port: LPSPI2_PCS0 of instance: lpspi2 011 **ALT3** — Select mux mode: ALT3 mux port: XBAR1_IN03 of instance: xbar1 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO1_FLEXIO01 of instance: flexio1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO01 of instance: gpio4

11.6.3 SW_MUX_CTL_PAD_GPIO_EMC_02 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_02)

SW_MUX_CTL Register

Address: 401F_8000h base + 1Ch offset = 401F_801Ch



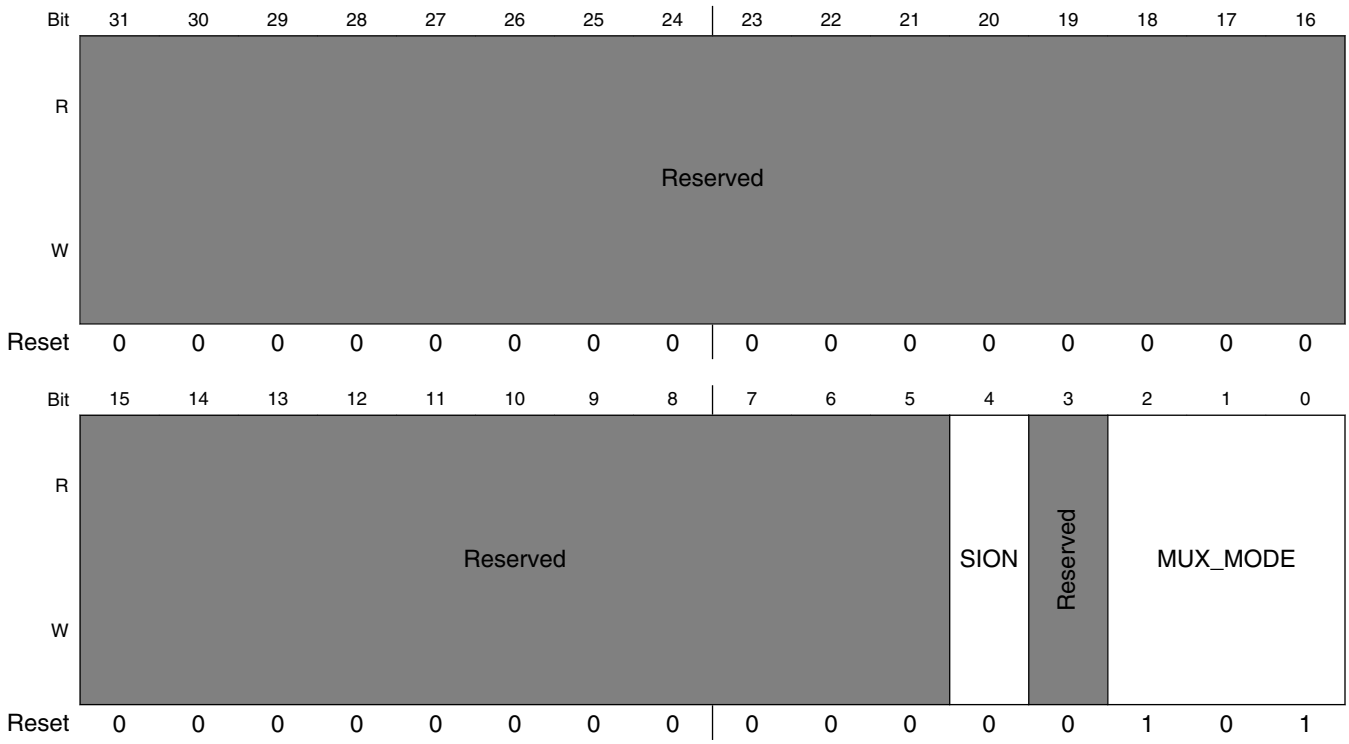
IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_02 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_EMC_02 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_EMC_02. 000 **ALT0** — Select mux mode: ALT0 mux port: SEMC_DATA02 of instance: semc 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXPWM4_PWMA01 of instance: flexpwm4 010 **ALT2** — Select mux mode: ALT2 mux port: LPSPI2_SDO of instance: lpspi2 011 **ALT3** — Select mux mode: ALT3 mux port: XBAR1_INOUT04 of instance: xbar1 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO1_FLEXIO02 of instance: flexio1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO02 of instance: gpio4

11.6.4 SW_MUX_CTL_PAD_GPIO_EMC_03 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_03)

SW_MUX_CTL Register

Address: 401F_8000h base + 20h offset = 401F_8020h



IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_03 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_EMC_03 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_EMC_03. 000 **ALT0** — Select mux mode: ALT0 mux port: SEMC_DATA03 of instance: semc 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXPWM4_PWMB01 of instance: flexpwm4 010 **ALT2** — Select mux mode: ALT2 mux port: LPSPI2_SDI of instance: lpspi2

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_03 field descriptions (continued)

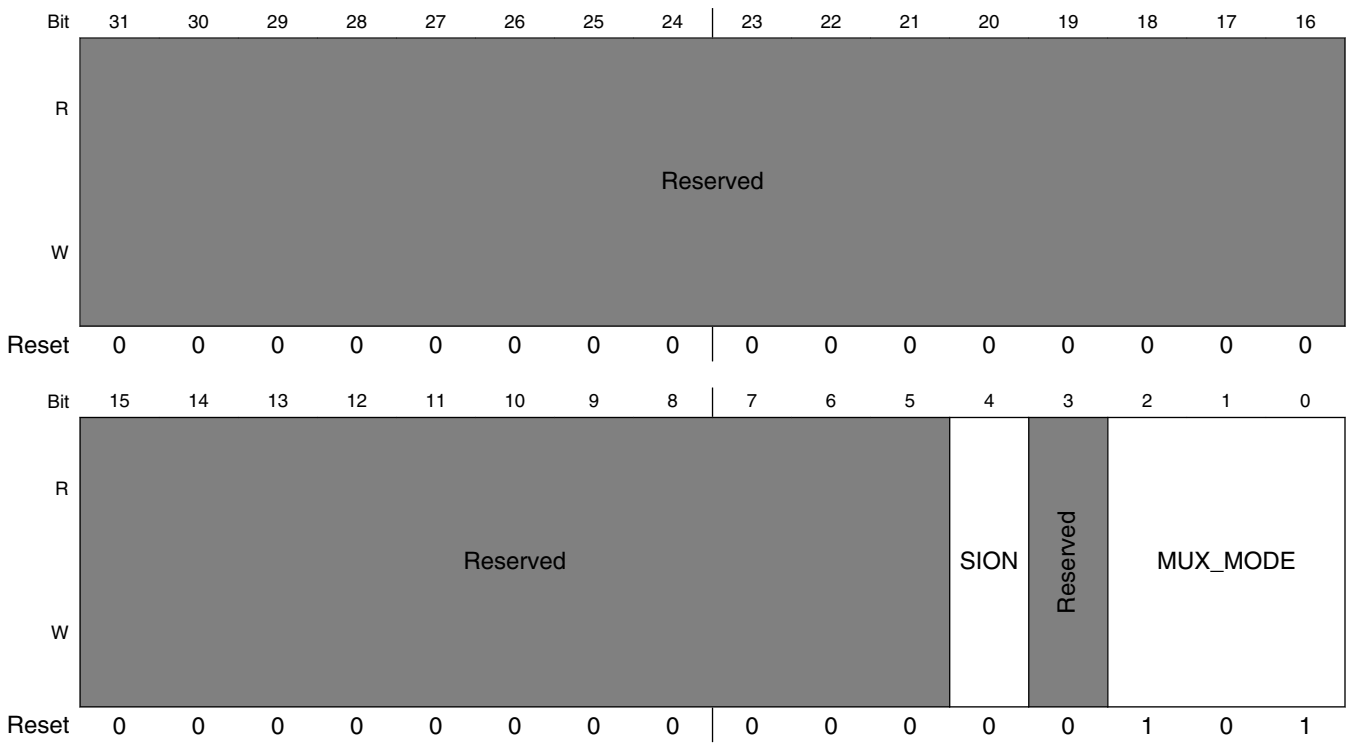
Field	Description
	011 **ALT3** — Select mux mode: ALT3 mux port: XBAR1_INOUT05 of instance: xbar1 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO1_FLEXIO03 of instance: flexio1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO03 of instance: gpio4

11.6.5 SW_MUX_CTL_PAD_GPIO_EMC_04 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_04)

pin 2

SW_MUX_CTL Register

Address: 401F_8000h base + 24h offset = 401F_8024h



IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_04 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_EMC_04 0 **DISABLED** — Input Path is determined by functionality

Table continues on the next page...

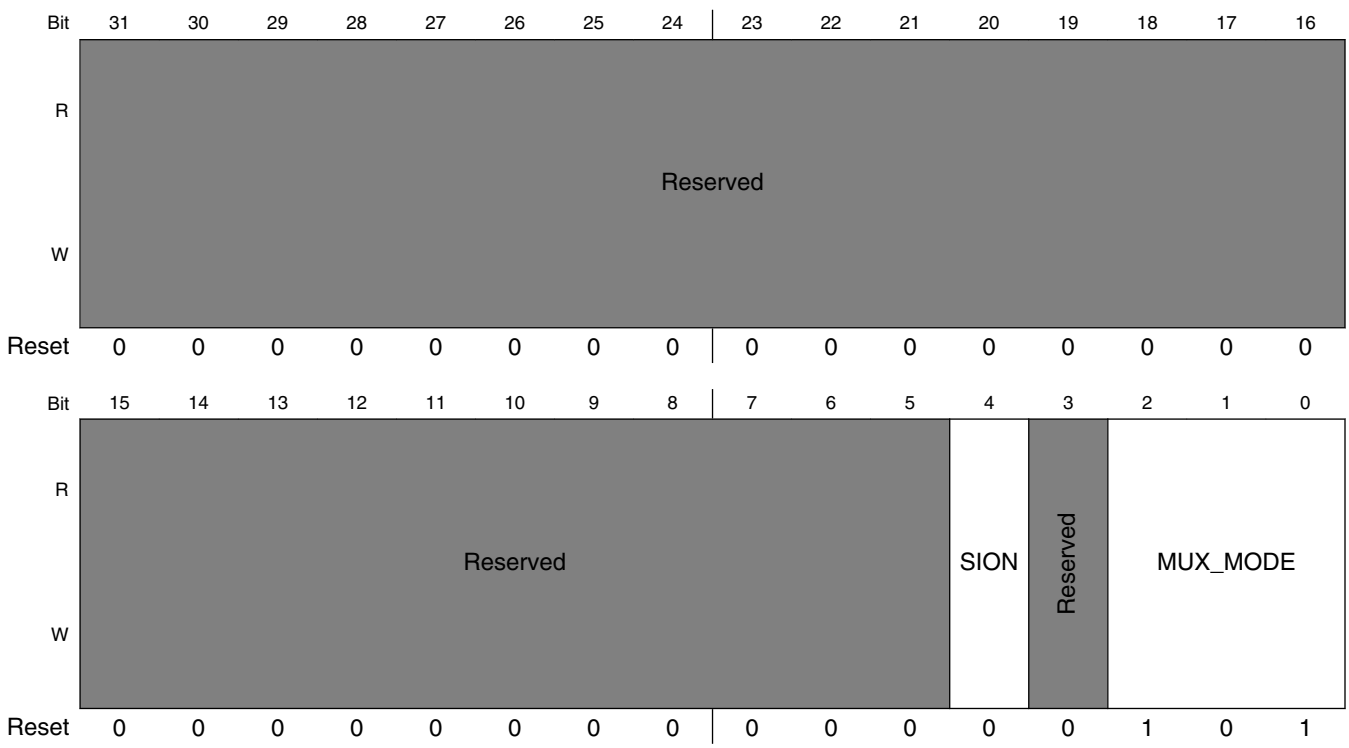
IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_04 field descriptions (continued)

Field	Description
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_EMC_04. pin 2 000 **ALT0** — Select mux mode: ALT0 mux port: SEMC_DATA04 of instance: semc 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXPWM4_PWMA02 of instance: flexpwm4 010 **ALT2** — Select mux mode: ALT2 mux port: SAI2_TX_DATA of instance: sai2 011 **ALT3** — Select mux mode: ALT3 mux port: XBAR1_INOUT06 of instance: xbar1 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO1_FLEXIO04 of instance: flexio1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO04 of instance: gpio4

11.6.6 SW_MUX_CTL_PAD_GPIO_EMC_05 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_05) pin 3

SW_MUX_CTL Register

Address: 401F_8000h base + 28h offset = 401F_8028h



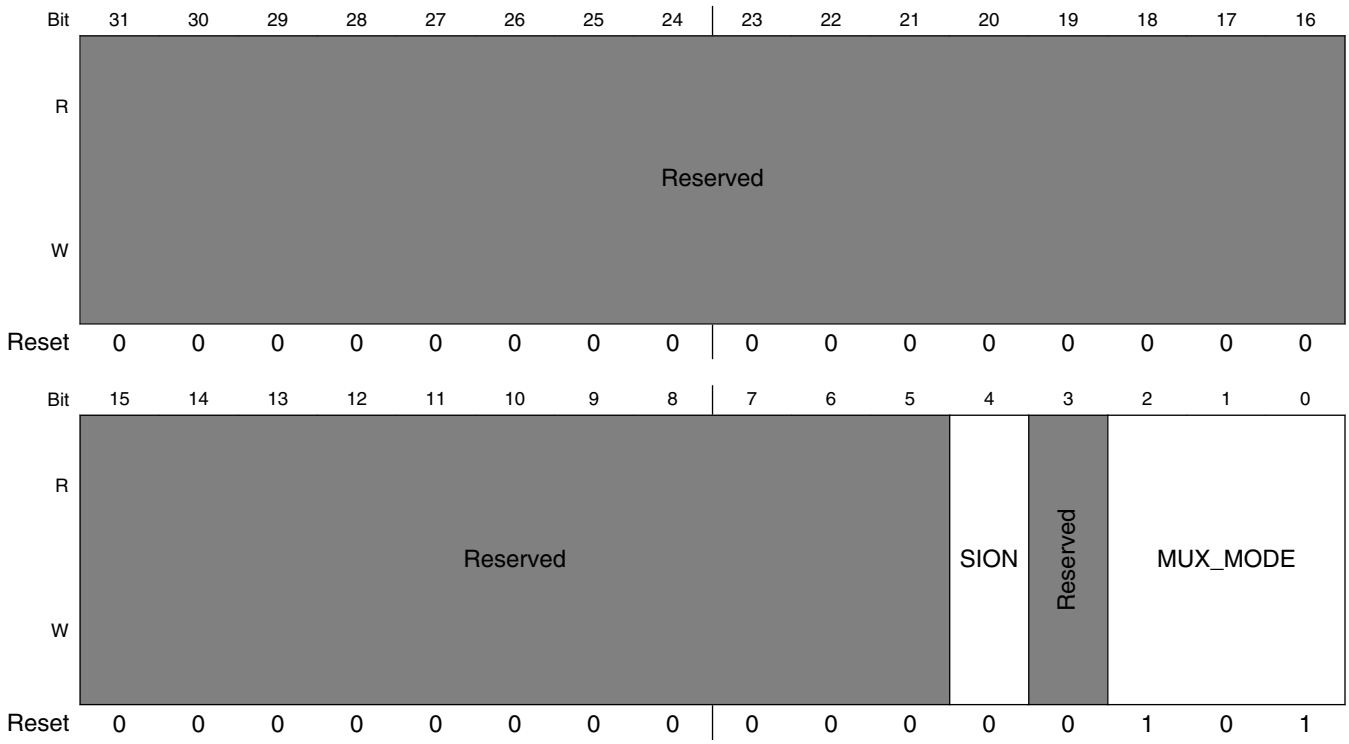
IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_05 field descriptions pin 3

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_EMC_05 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_EMC_05. pin 3 000 **ALT0** — Select mux mode: ALT0 mux port: SEMC_DATA05 of instance: semc 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXPWM4_PWMB02 of instance: flexpwm4 010 **ALT2** — Select mux mode: ALT2 mux port: SAI2_TX_SYNC of instance: sai2 011 **ALT3** — Select mux mode: ALT3 mux port: XBAR1_INOUT07 of instance: xbar1 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO1_FLEXIO05 of instance: flexio1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO05 of instance: gpio4

11.6.7 SW_MUX_CTL_PAD_GPIO_EMC_06 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_06) pin 4

SW_MUX_CTL Register

Address: 401F_8000h base + 2Ch offset = 401F_802Ch



IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_06 field descriptions pin 4

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_EMC_06 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_EMC_06. pin 4 000 **ALT0** — Select mux mode: ALT0 mux port: SEMC_DATA06 of instance: semc 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXPWM2_PWMA00 of instance: flexpwm2 010 **ALT2** — Select mux mode: ALT2 mux port: SAI2_TX_BCLK of instance: sai2

Table continues on the next page...

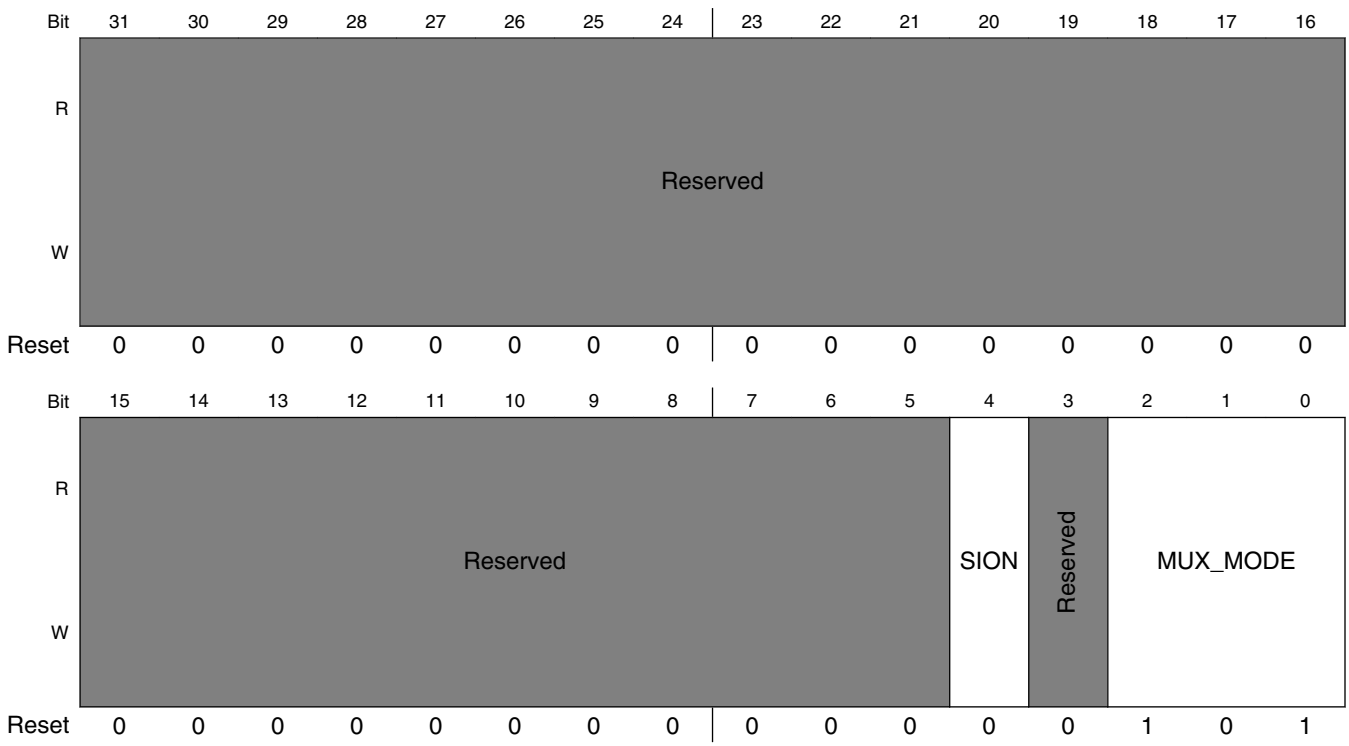
IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_06 field descriptions (continued) pin 4

Field	Description
	011 **ALT3** — Select mux mode: ALT3 mux port: XBAR1_INOUT08 of instance: xbar1 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO1_FLEXIO06 of instance: flexio1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO06 of instance: gpio4

11.6.8 SW_MUX_CTL_PAD_GPIO_EMC_07 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_07) pin 33

SW_MUX_CTL Register

Address: 401F_8000h base + 30h offset = 401F_8030h



IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_07 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_EMC_07 0 **DISABLED** — Input Path is determined by functionality

Table continues on the next page...

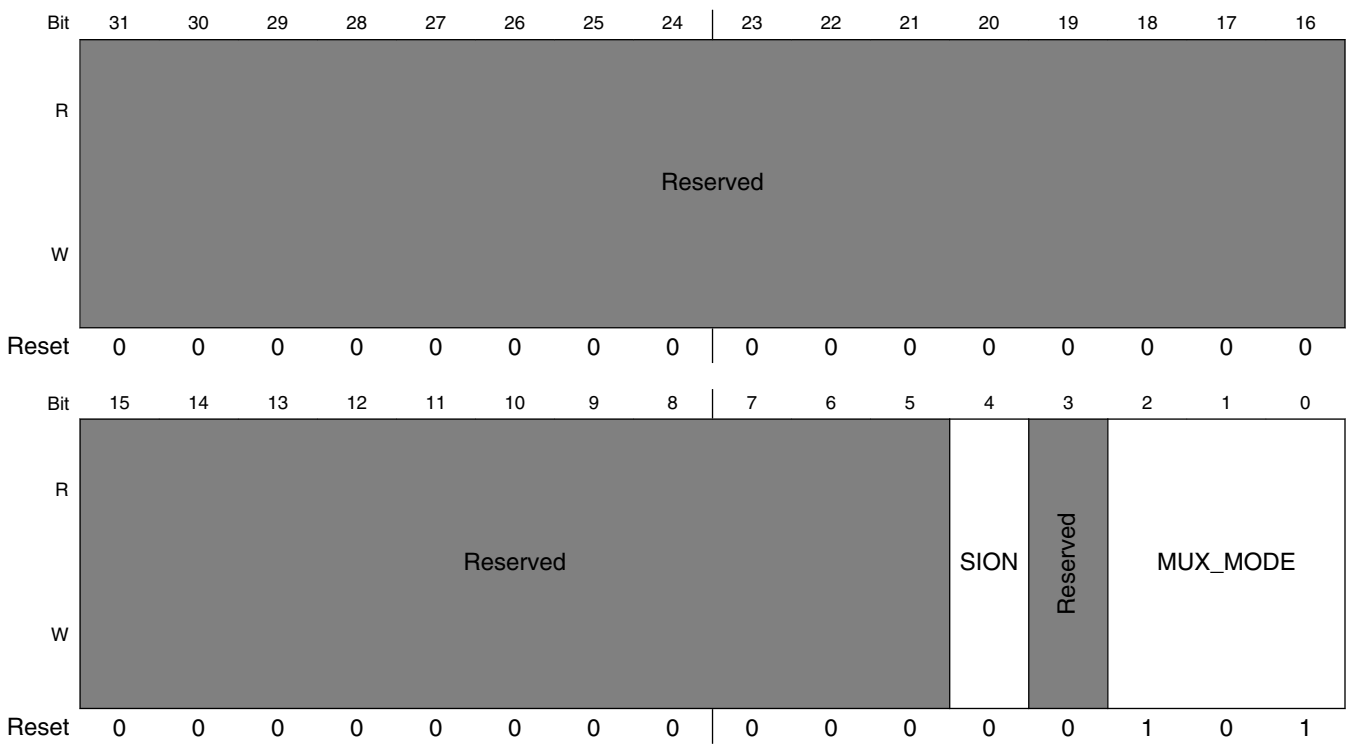
IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_07 field descriptions (continued)

Field	Description
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_EMC_07. pin 33 000 **ALT0** — Select mux mode: ALT0 mux port: SEMC_DATA07 of instance: semc 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXPWM2_PWMB00 of instance: flexpwm2 010 **ALT2** — Select mux mode: ALT2 mux port: SAI2_MCLK of instance: sai2 011 **ALT3** — Select mux mode: ALT3 mux port: XBAR1_INOUT09 of instance: xbar1 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO1_FLEXIO07 of instance: flexio1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO07 of instance: gpio4

11.6.9 SW_MUX_CTL_PAD_GPIO_EMC_08 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_08) pin 5

SW_MUX_CTL Register

Address: 401F_8000h base + 34h offset = 401F_8034h



IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_08 field descriptions pin 5

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_EMC_08 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_EMC_08. 000 **ALT0** — Select mux mode: ALT0 mux port: SEMC_DM00 of instance: semc 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXPWM2_PWMA01 of instance: flexpwm2 010 **ALT2** — Select mux mode: ALT2 mux port: SAI2_RX_DATA of instance: sai2 011 **ALT3** — Select mux mode: ALT3 mux port: XBAR1_INOUT17 of instance: xbar1 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO1_FLEXIO08 of instance: flexio1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO08 of instance: gpio4

11.6.10 SW_MUX_CTL_PAD_GPIO_EMC_09 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_09)

SW_MUX_CTL Register

Address: 401F_8000h base + 38h offset = 401F_8038h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_09 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality.

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_09 field descriptions (continued)

Field	Description
	1 **ENABLED** — Force input path of pad GPIO_EMC_09 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_EMC_09. 000 **ALT0** — Select mux mode: ALT0 mux port: SEMC_ADDR00 of instance: semc 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXPWM2_PWMB01 of instance: flexpwm2 010 **ALT2** — Select mux mode: ALT2 mux port: SAI2_RX_SYNC of instance: sai2 011 **ALT3** — Select mux mode: ALT3 mux port: FLEXCAN2_TX of instance: flexcan2 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO1_FLEXIO09 of instance: flexio1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO09 of instance: gpio4 1000 **ALT8** — Select mux mode: ALT8 mux port: FLEXSPI2_B_SS1_B of instance: flexspi2

11.6.11 SW_MUX_CTL_PAD_GPIO_EMC_10 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_10)

SW_MUX_CTL Register

Address: 401F_8000h base + 3Ch offset = 401F_803Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_10 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_EMC_10 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_EMC_10. 000 **ALT0** — Select mux mode: ALT0 mux port: SEMC_ADDR01 of instance: semc 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXPWM2_PWMA02 of instance: flexpwm2

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_10 field descriptions (continued)

Field	Description
	010 **ALT2** — Select mux mode: ALT2 mux port: SAI2_RX_BCLK of instance: sai2 011 **ALT3** — Select mux mode: ALT3 mux port: FLEXCAN2_RX of instance: flexcan2 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO1_FLEXIO10 of instance: flexio1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO10 of instance: gpio4 1000 **ALT8** — Select mux mode: ALT8 mux port: FLEXSPI2_B_SS0_B of instance: flexspi2

11.6.12 SW_MUX_CTL_PAD_GPIO_EMC_11 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_11)

SW_MUX_CTL Register

Address: 401F_8000h base + 40h offset = 401F_8040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_11 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_EMC_11 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_EMC_11. 000 **ALT0** — Select mux mode: ALT0 mux port: SEMC_ADDR02 of instance: semc 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXPWM2_PWMB02 of instance: flexpwm2 010 **ALT2** — Select mux mode: ALT2 mux port: LPI2C4_SDA of instance: lpi2c4 011 **ALT3** — Select mux mode: ALT3 mux port: USDHC2_RESET_B of instance: usdhc2 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO1_FLEXIO11 of instance: flexio1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO11 of instance: gpio4 1000 **ALT8** — Select mux mode: ALT8 mux port: FLEXSPI2_B_DQS of instance: flexspi2

11.6.13 SW_MUX_CTL_PAD_GPIO_EMC_12 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_12)

SW_MUX_CTL Register

Address: 401F_8000h base + 44h offset = 401F_8044h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_12 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_EMC_12 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_EMC_12. 000 **ALT0** — Select mux mode: ALT0 mux port: SEMC_ADDR03 of instance: semc 001 **ALT1** — Select mux mode: ALT1 mux port: XBAR1_IN24 of instance: xbar1 010 **ALT2** — Select mux mode: ALT2 mux port: LPI2C4_SCL of instance: lpi2c4 011 **ALT3** — Select mux mode: ALT3 mux port: USDHC1_WP of instance: usdhc1 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXPWM1_PWMA03 of instance: flexpwm1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO12 of instance: gpio4 1000 **ALT8** — Select mux mode: ALT8 mux port: FLEXSPI2_B_SCLK of instance: flexspi2

11.6.14 SW_MUX_CTL_PAD_GPIO_EMC_13 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_13)

SW_MUX_CTL Register

Address: 401F_8000h base + 48h offset = 401F_8048h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_13 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_EMC_13 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_EMC_13. 000 **ALT0** — Select mux mode: ALT0 mux port: SEMC_ADDR04 of instance: semc 001 **ALT1** — Select mux mode: ALT1 mux port: XBAR1_IN25 of instance: xbar1 010 **ALT2** — Select mux mode: ALT2 mux port: LPUART3_TX of instance: lpuart3 011 **ALT3** — Select mux mode: ALT3 mux port: MQS_RIGHT of instance: mqs 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXPWM1_PWMB03 of instance: flexpwm1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO13 of instance: gpio4 1000 **ALT8** — Select mux mode: ALT8 mux port: FLEXSPI2_B_DATA00 of instance: flexspi2

11.6.15 SW_MUX_CTL_PAD_GPIO_EMC_14 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_14)

SW_MUX_CTL Register

Address: 401F_8000h base + 4Ch offset = 401F_804Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_14 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_EMC_14 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_EMC_14. 000 **ALT0** — Select mux mode: ALT0 mux port: SEMC_ADDR05 of instance: semc 001 **ALT1** — Select mux mode: ALT1 mux port: XBAR1_INOUT19 of instance: xbar1 010 **ALT2** — Select mux mode: ALT2 mux port: LPUART3_RX of instance: lpuart3 011 **ALT3** — Select mux mode: ALT3 mux port: MQS_LEFT of instance: mqs 100 **ALT4** — Select mux mode: ALT4 mux port: LPSPI2_PCS1 of instance: lpspi2 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO14 of instance: gpio4 1000 **ALT8** — Select mux mode: ALT8 mux port: FLEXSPI2_B_DATA01 of instance: flexspi2

11.6.16 SW_MUX_CTL_PAD_GPIO_EMC_15 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_15)

SW_MUX_CTL Register

Address: 401F_8000h base + 50h offset = 401F_8050h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_15 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_EMC_15 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_EMC_15. 000 **ALT0** — Select mux mode: ALT0 mux port: SEMC_ADDR06 of instance: semc 001 **ALT1** — Select mux mode: ALT1 mux port: XBAR1_IN20 of instance: xbar1 010 **ALT2** — Select mux mode: ALT2 mux port: LPUART3_CTS_B of instance: lpuart3 011 **ALT3** — Select mux mode: ALT3 mux port: SPDIF_OUT of instance: spdif 100 **ALT4** — Select mux mode: ALT4 mux port: QTIMER3_TIMER0 of instance: qtimer3 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO15 of instance: gpio4 1000 **ALT8** — Select mux mode: ALT8 mux port: FLEXSPI2_B_DATA02 of instance: flexspi2

11.6.17 SW_MUX_CTL_PAD_GPIO_EMC_16 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_16)

SW_MUX_CTL Register

Address: 401F_8000h base + 54h offset = 401F_8054h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

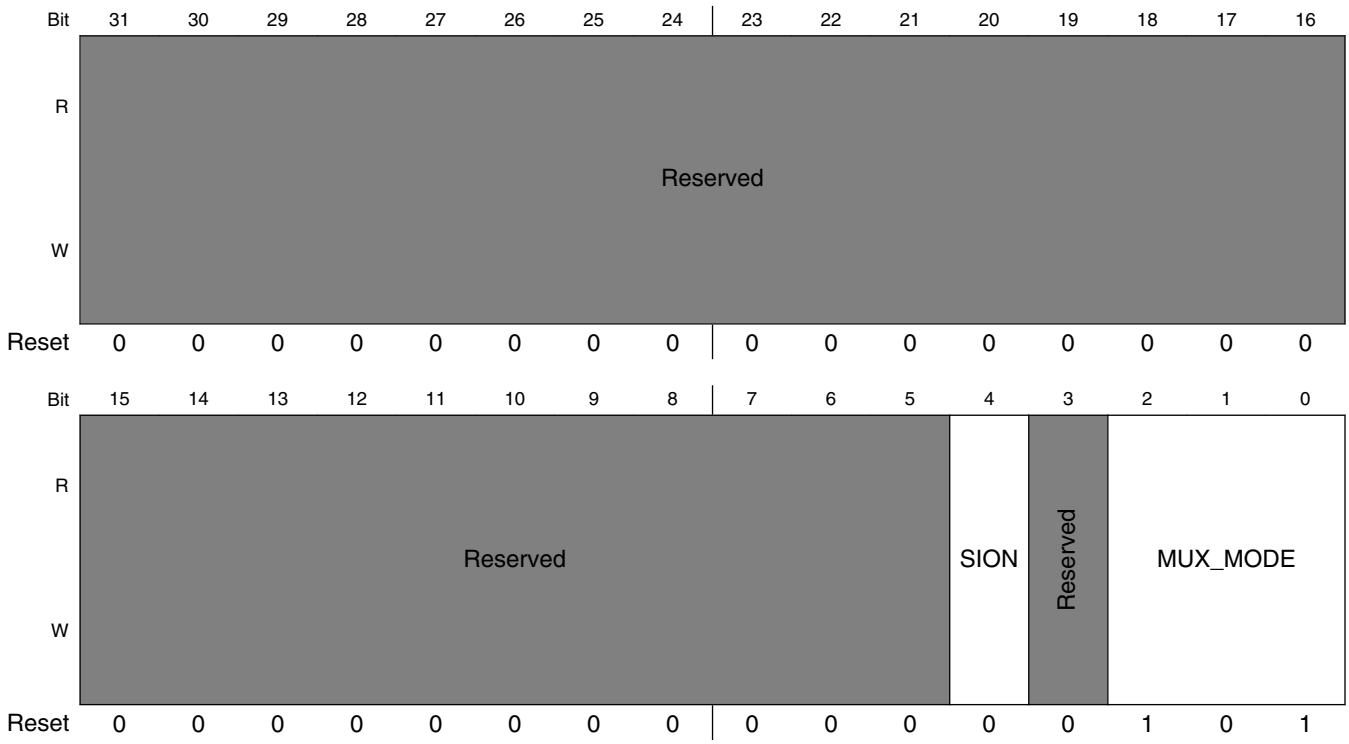
IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_16 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_EMC_16 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_EMC_16. 000 **ALT0** — Select mux mode: ALT0 mux port: SEMC_ADDR07 of instance: semc 001 **ALT1** — Select mux mode: ALT1 mux port: XBAR1_IN21 of instance: xbar1 010 **ALT2** — Select mux mode: ALT2 mux port: LPUART3_RTS_B of instance: lpuart3 011 **ALT3** — Select mux mode: ALT3 mux port: SPDIF_IN of instance: spdif 100 **ALT4** — Select mux mode: ALT4 mux port: QTIMER3_TIMER1 of instance: qtimer3 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO16 of instance: gpio4 1000 **ALT8** — Select mux mode: ALT8 mux port: FLEXSPI2_B_DATA03 of instance: flexspi2

11.6.18 SW_MUX_CTL_PAD_GPIO_EMC_17 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_17)

SW_MUX_CTL Register

Address: 401F_8000h base + 58h offset = 401F_8058h



IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_17 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_EMC_17 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_EMC_17. 000 **ALT0** — Select mux mode: ALT0 mux port: SEMC_ADDR08 of instance: semc 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXPWM4_PWMA03 of instance: flexpwm4 010 **ALT2** — Select mux mode: ALT2 mux port: LPUART4_CTS_B of instance: lpuart4

Table continues on the next page...

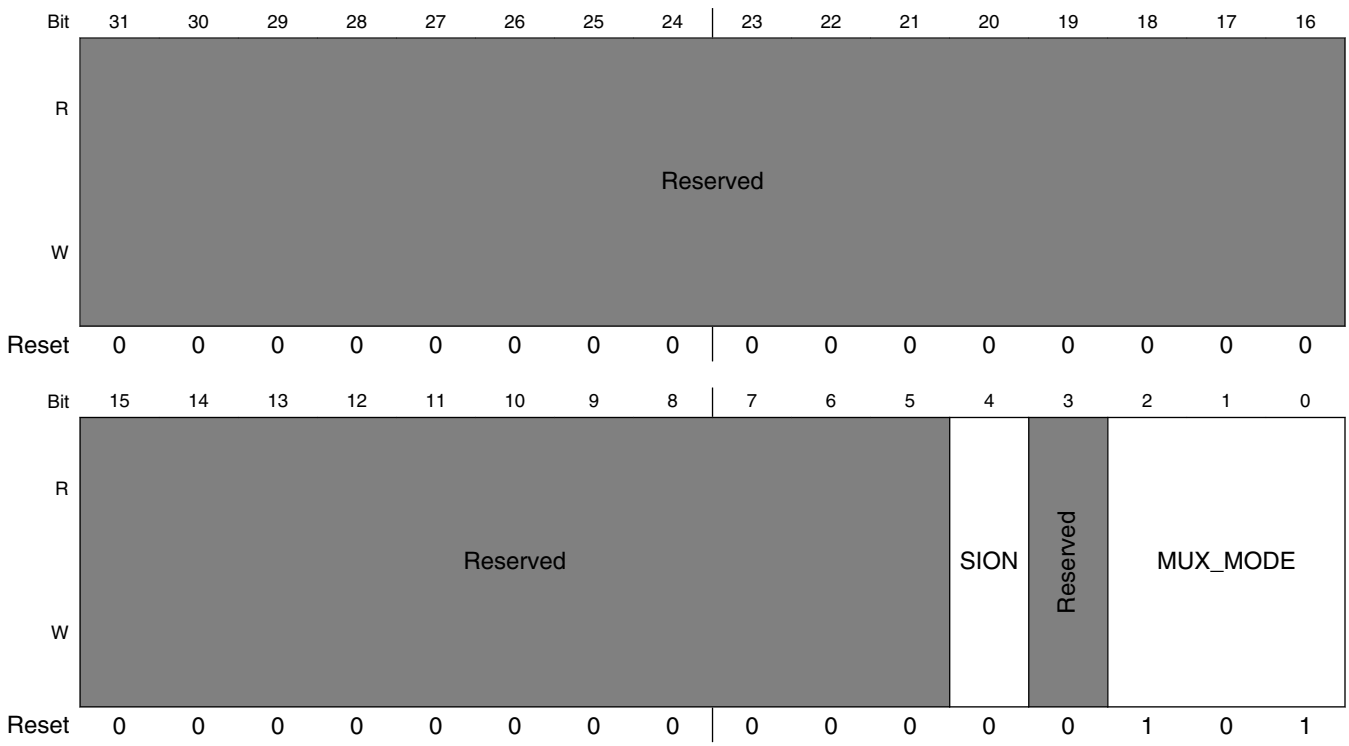
IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_17 field descriptions (continued)

Field	Description
	011 **ALT3** — Select mux mode: ALT3 mux port: FLEXCAN1_TX of instance: flexcan1 100 **ALT4** — Select mux mode: ALT4 mux port: QTIMER3_TIMER2 of instance: qtimer3 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO17 of instance: gpio4

11.6.19 SW_MUX_CTL_PAD_GPIO_EMC_18 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_18)

SW_MUX_CTL Register

Address: 401F_8000h base + 5Ch offset = 401F_805Ch



IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_18 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_EMC_18 0 **DISABLED** — Input Path is determined by functionality

Table continues on the next page...

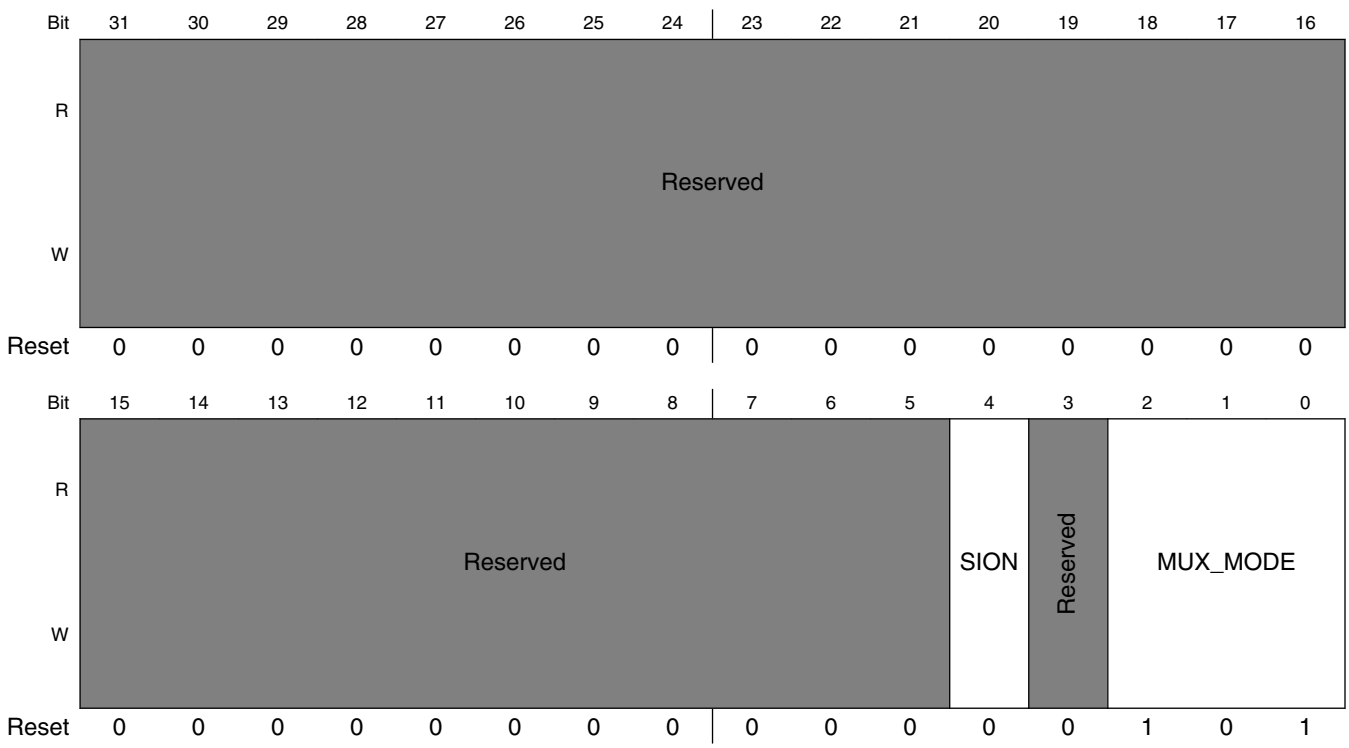
IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_18 field descriptions (continued)

Field	Description
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_EMC_18. 000 **ALT0** — Select mux mode: ALT0 mux port: SEMC_ADDR09 of instance: semc 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXPWM4_PWMB03 of instance: flexpwm4 010 **ALT2** — Select mux mode: ALT2 mux port: LPUART4_RTS_B of instance: lpuart4 011 **ALT3** — Select mux mode: ALT3 mux port: FLEXCAN1_RX of instance: flexcan1 100 **ALT4** — Select mux mode: ALT4 mux port: QTIMER3_TIMER3 of instance: qtimer3 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO18 of instance: gpio4 110 **ALT6** — Select mux mode: ALT6 mux port: SNVS_VIO_5_CTL of instance: snvs_hp

11.6.20 SW_MUX_CTL_PAD_GPIO_EMC_19 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_19)

SW_MUX_CTL Register

Address: 401F_8000h base + 60h offset = 401F_8060h



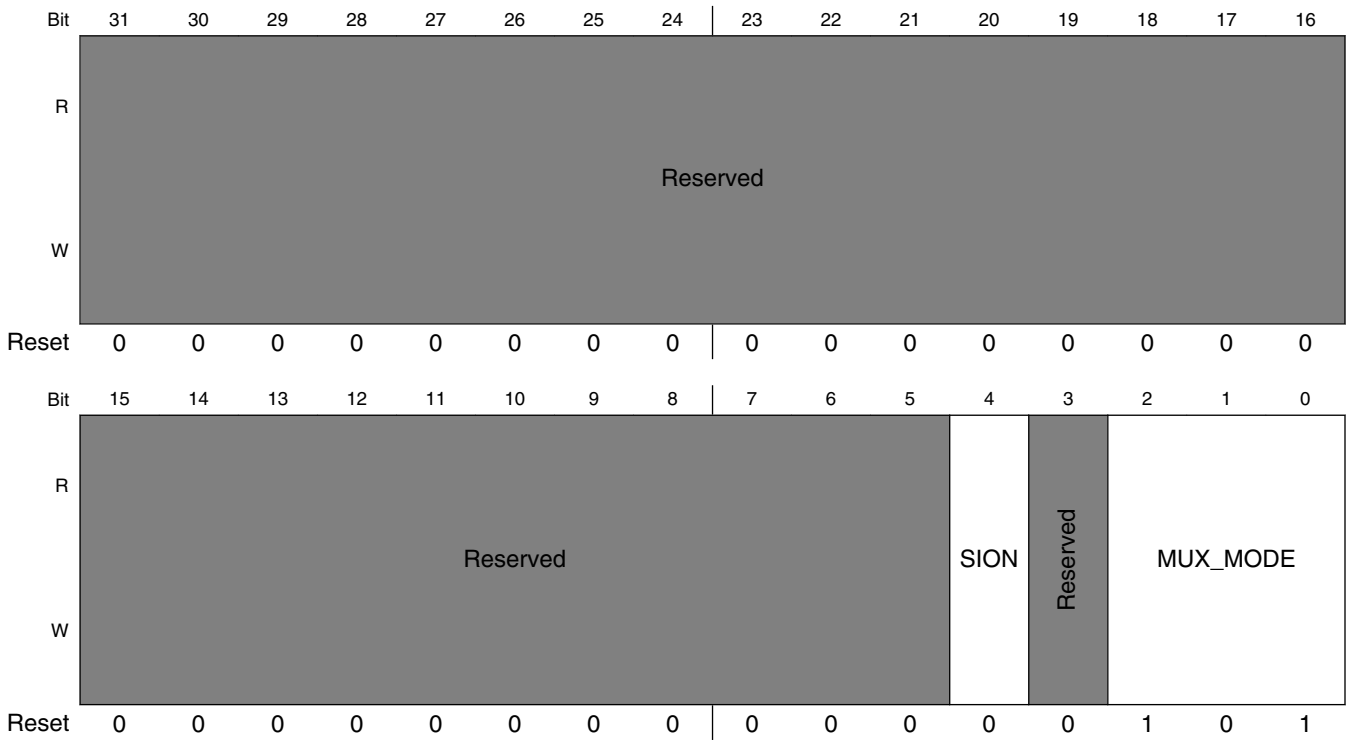
IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_19 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_EMC_19 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_EMC_19. 000 **ALT0** — Select mux mode: ALT0 mux port: SEMC_ADDR11 of instance: semc 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXPWM2_PWMA03 of instance: flexpwm2 010 **ALT2** — Select mux mode: ALT2 mux port: LPUART4_TX of instance: lpuart4 011 **ALT3** — Select mux mode: ALT3 mux port: ENET_RDATA01 of instance: enet 100 **ALT4** — Select mux mode: ALT4 mux port: QTIMER2_TIMER0 of instance: qtimer2 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO19 of instance: gpio4 110 **ALT6** — Select mux mode: ALT6 mux port: SNVS_VIO_5 of instance: snvs_hp

11.6.21 SW_MUX_CTL_PAD_GPIO_EMC_20 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_20)

SW_MUX_CTL Register

Address: 401F_8000h base + 64h offset = 401F_8064h



IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_20 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_EMC_20 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_EMC_20. 000 **ALT0** — Select mux mode: ALT0 mux port: SEMC_ADDR12 of instance: semc 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXPWM2_PWMB03 of instance: flexpwm2 010 **ALT2** — Select mux mode: ALT2 mux port: LPUART4_RX of instance: lpuart4

Table continues on the next page...

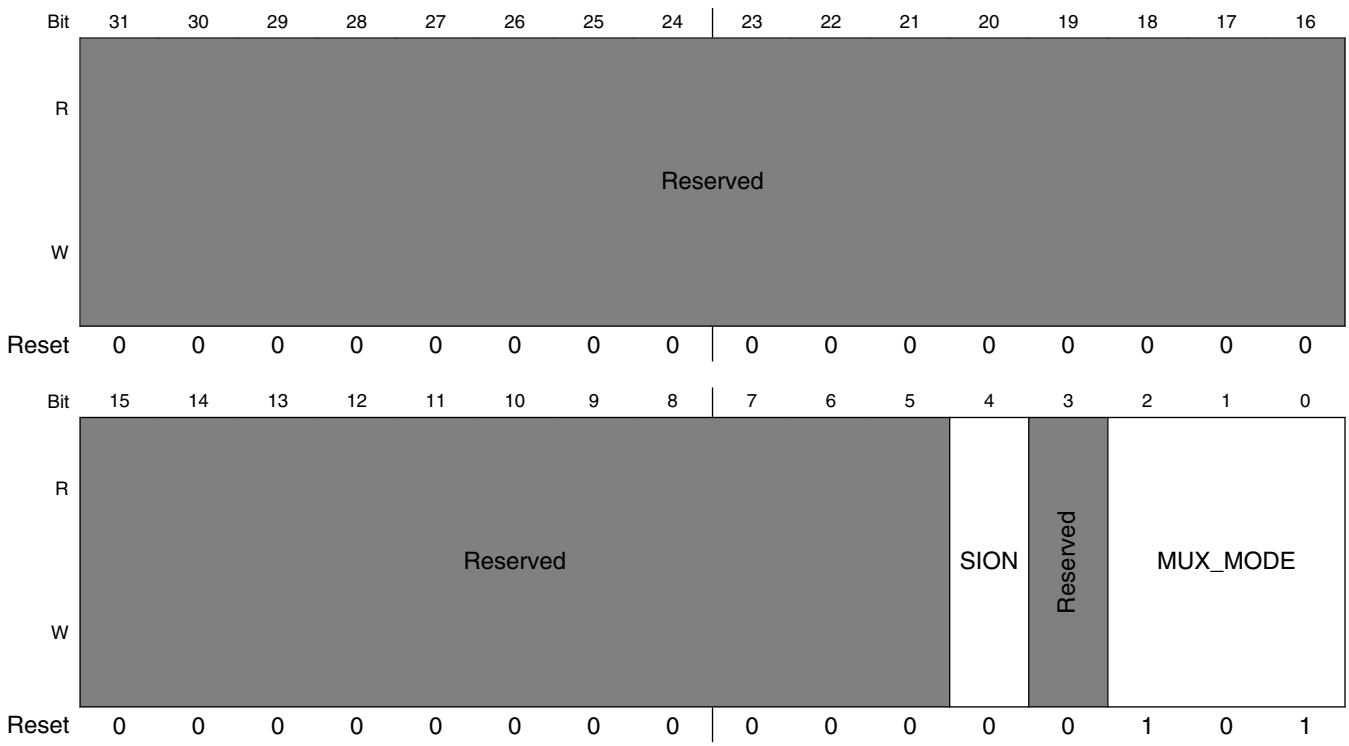
IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_20 field descriptions (continued)

Field	Description
	011 **ALT3** — Select mux mode: ALT3 mux port: ENET_RDATA00 of instance: enet 100 **ALT4** — Select mux mode: ALT4 mux port: QTIMER2_TIMER1 of instance: qtimer2 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO20 of instance: gpio4

11.6.22 SW_MUX_CTL_PAD_GPIO_EMC_21 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_21)

SW_MUX_CTL Register

Address: 401F_8000h base + 68h offset = 401F_8068h



IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_21 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_EMC_21 0 **DISABLED** — Input Path is determined by functionality

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_21 field descriptions (continued)

Field	Description
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_EMC_21. 000 **ALT0** — Select mux mode: ALT0 mux port: SEMC_BA0 of instance: semc 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXPWM3_PWMA03 of instance: flexpwm3 010 **ALT2** — Select mux mode: ALT2 mux port: LPI2C3_SDA of instance: lpi2c3 011 **ALT3** — Select mux mode: ALT3 mux port: ENET_TDATA01 of instance: enet 100 **ALT4** — Select mux mode: ALT4 mux port: QTIMER2_TIMER2 of instance: qtimer2 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO21 of instance: gpio4

11.6.23 SW_MUX_CTL_PAD_GPIO_EMC_22 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_22)

SW_MUX_CTL Register

Address: 401F_8000h base + 6Ch offset = 401F_806Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_22 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_EMC_22 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_EMC_22. 000 **ALT0** — Select mux mode: ALT0 mux port: SEMC_BA1 of instance: semc 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXPWM3_PWMB03 of instance: flexpwm3 010 **ALT2** — Select mux mode: ALT2 mux port: LPI2C3_SCL of instance: lpi2c3

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_22 field descriptions (continued)

Field	Description
	011 **ALT3** — Select mux mode: ALT3 mux port: ENET_TDATA00 of instance: enet 100 **ALT4** — Select mux mode: ALT4 mux port: QTIMER2_TIMER3 of instance: qtimer2 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO22 of instance: gpio4 1000 **ALT8** — Select mux mode: ALT8 mux port: FLEXSPI2_A_SS1_B of instance: flexspi2

11.6.24 SW_MUX_CTL_PAD_GPIO_EMC_23 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_23)

SW_MUX_CTL Register

Address: 401F_8000h base + 70h offset = 401F_8070h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_23 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_EMC_23 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_EMC_23. 000 **ALT0** — Select mux mode: ALT0 mux port: SEMC_ADDR10 of instance: semc 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXPWM1_PWMA00 of instance: flexpwm1 010 **ALT2** — Select mux mode: ALT2 mux port: LPUART5_TX of instance: lpuart5 011 **ALT3** — Select mux mode: ALT3 mux port: ENET_RX_EN of instance: enet 100 **ALT4** — Select mux mode: ALT4 mux port: GPT1_CAPTURE2 of instance: gpt1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO23 of instance: gpio4 1000 **ALT8** — Select mux mode: ALT8 mux port: FLEXSPI2_A_DQS of instance: flexspi2

11.6.25 SW_MUX_CTL_PAD_GPIO_EMC_24 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_24)

SW_MUX_CTL Register

Address: 401F_8000h base + 74h offset = 401F_8074h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_24 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_EMC_24 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_EMC_24. 000 **ALT0** — Select mux mode: ALT0 mux port: SEMC_CAS of instance: semc 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXPWM1_PWMB00 of instance: flexpwm1 010 **ALT2** — Select mux mode: ALT2 mux port: LPUART5_RX of instance: lpuart5 011 **ALT3** — Select mux mode: ALT3 mux port: ENET_TX_EN of instance: enet 100 **ALT4** — Select mux mode: ALT4 mux port: GPT1_CAPTURE1 of instance: gpt1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO24 of instance: gpio4 1000 **ALT8** — Select mux mode: ALT8 mux port: FLEXSPI2_A_SS0_B of instance: flexspi2

11.6.26 SW_MUX_CTL_PAD_GPIO_EMC_25 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_25)

SW_MUX_CTL Register

Address: 401F_8000h base + 78h offset = 401F_8078h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_25 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_EMC_25 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_EMC_25. 000 **ALT0** — Select mux mode: ALT0 mux port: SEMC_RAS of instance: semc 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXPWM1_PWMA01 of instance: flexpwm1 010 **ALT2** — Select mux mode: ALT2 mux port: LPUART6_TX of instance: lpuart6 011 **ALT3** — Select mux mode: ALT3 mux port: ENET_TX_CLK of instance: enet 100 **ALT4** — Select mux mode: ALT4 mux port: ENET_REF_CLK of instance: enet 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO25 of instance: gpio4 1000 **ALT8** — Select mux mode: ALT8 mux port: FLEXSPI2_A_SCLK of instance: flexspi2

11.6.27 SW_MUX_CTL_PAD_GPIO_EMC_26 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_26)

SW_MUX_CTL Register

Address: 401F_8000h base + 7Ch offset = 401F_807Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_26 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_EMC_26 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_EMC_26. 000 **ALT0** — Select mux mode: ALT0 mux port: SEMC_CLK of instance: semc 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXPWM1_PWMB01 of instance: flexpwm1 010 **ALT2** — Select mux mode: ALT2 mux port: LPUART6_RX of instance: lpuart6 011 **ALT3** — Select mux mode: ALT3 mux port: ENET_RX_ER of instance: enet 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO1_FLEXIO12 of instance: flexio1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO26 of instance: gpio4 1000 **ALT8** — Select mux mode: ALT8 mux port: FLEXSPI2_A_DATA00 of instance: flexspi2

11.6.28 SW_MUX_CTL_PAD_GPIO_EMC_27 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_27)

SW_MUX_CTL Register

Address: 401F_8000h base + 80h offset = 401F_8080h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_27 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_EMC_27 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_EMC_27. 000 **ALT0** — Select mux mode: ALT0 mux port: SEMC_CKE of instance: semc 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXPWM1_PWMA02 of instance: flexpwm1 010 **ALT2** — Select mux mode: ALT2 mux port: LPUART5_RTS_B of instance: lpuart5 011 **ALT3** — Select mux mode: ALT3 mux port: LPSPI1_SCK of instance: lpspi1 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO1_FLEXIO13 of instance: flexio1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO27 of instance: gpio4 1000 **ALT8** — Select mux mode: ALT8 mux port: FLEXSPI2_A_DATA01 of instance: flexspi2

11.6.29 SW_MUX_CTL_PAD_GPIO_EMC_28 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_28)

SW_MUX_CTL Register

Address: 401F_8000h base + 84h offset = 401F_8084h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_28 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_EMC_28 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_EMC_28. 000 **ALT0** — Select mux mode: ALT0 mux port: SEMC_WE of instance: semc 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXPWM1_PWMB02 of instance: flexpwm1 010 **ALT2** — Select mux mode: ALT2 mux port: LPUART5_CTS_B of instance: lpuart5 011 **ALT3** — Select mux mode: ALT3 mux port: LPSPI1_SDO of instance: lpspi1 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO1_FLEXIO14 of instance: flexio1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO28 of instance: gpio4 1000 **ALT8** — Select mux mode: ALT8 mux port: FLEXSPI2_A_DATA02 of instance: flexspi2

11.6.30 SW_MUX_CTL_PAD_GPIO_EMC_29 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_29)

SW_MUX_CTL Register

Address: 401F_8000h base + 88h offset = 401F_8088h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_29 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_EMC_29 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_EMC_29. 000 **ALT0** — Select mux mode: ALT0 mux port: SEMC_CS0 of instance: semc 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXPWM3_PWMA00 of instance: flexpwm3 010 **ALT2** — Select mux mode: ALT2 mux port: LPUART6_RTS_B of instance: lpuart6 011 **ALT3** — Select mux mode: ALT3 mux port: LPSPI1_SDI of instance: lpspi1 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO1_FLEXIO15 of instance: flexio1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO29 of instance: gpio4 1000 **ALT8** — Select mux mode: ALT8 mux port: FLEXSPI2_A_DATA03 of instance: flexspi2

11.6.31 SW_MUX_CTL_PAD_GPIO_EMC_30 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_30)

SW_MUX_CTL Register

Address: 401F_8000h base + 8Ch offset = 401F_808Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_30 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_EMC_30 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_EMC_30. 000 **ALT0** — Select mux mode: ALT0 mux port: SEMC_DATA08 of instance: semc 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXPWM3_PWMB00 of instance: flexpwm3 010 **ALT2** — Select mux mode: ALT2 mux port: LPUART6_CTS_B of instance: lpuart6 011 **ALT3** — Select mux mode: ALT3 mux port: LPSPI1_PCS0 of instance: lpspi1 100 **ALT4** — Select mux mode: ALT4 mux port: CSI_DATA23 of instance: csi 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO30 of instance: gpio4 1000 **ALT8** — Select mux mode: ALT8 mux port: ENET2_TDATA00 of instance: enet2

11.6.32 SW_MUX_CTL_PAD_GPIO_EMC_31 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_31) pin 29

SW_MUX_CTL Register

Address: 401F_8000h base + 90h offset = 401F_8090h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_31 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_EMC_31 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_EMC_31. pin 29 000 **ALT0** — Select mux mode: ALT0 mux port: SEMC_DATA09 of instance: semc 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXPWM3_PWMA01 of instance: flexpwm3 010 **ALT2** — Select mux mode: ALT2 mux port: LPUART7_TX of instance: lpuart7 011 **ALT3** — Select mux mode: ALT3 mux port: LPSPI1_PCS1 of instance: lpspi1 100 **ALT4** — Select mux mode: ALT4 mux port: CSI_DATA22 of instance: csi 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO4_IO31 of instance: gpio4 1000 **ALT8** — Select mux mode: ALT8 mux port: ENET2_TDATA01 of instance: enet2

11.6.33 SW_MUX_CTL_PAD_GPIO_EMC_32 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_32) pin 28

SW_MUX_CTL Register

Address: 401F_8000h base + 94h offset = 401F_8094h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_32 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_EMC_32 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_EMC_32. pin 28 000 **ALT0** — Select mux mode: ALT0 mux port: SEMC_DATA10 of instance: semc 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXPWM3_PWMB01 of instance: flexpwm3 010 **ALT2** — Select mux mode: ALT2 mux port: LPUART7_RX of instance: lpuart7 011 **ALT3** — Select mux mode: ALT3 mux port: CCM_PMIC_RDY of instance: ccm 100 **ALT4** — Select mux mode: ALT4 mux port: CSI_DATA21 of instance: csi 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO18 of instance: gpio3 1000 **ALT8** — Select mux mode: ALT8 mux port: ENET2_TX_EN of instance: enet2

11.6.34 SW_MUX_CTL_PAD_GPIO_EMC_33 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_33)

SW_MUX_CTL Register

Address: 401F_8000h base + 98h offset = 401F_8098h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_33 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_EMC_33 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_EMC_33. 000 **ALT0** — Select mux mode: ALT0 mux port: SEMC_DATA11 of instance: semc 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXPWM3_PWMA02 of instance: flexpwm3 010 **ALT2** — Select mux mode: ALT2 mux port: USDHC1_RESET_B of instance: usdhc1 011 **ALT3** — Select mux mode: ALT3 mux port: SAI3_RX_DATA of instance: sai3 100 **ALT4** — Select mux mode: ALT4 mux port: CSI_DATA20 of instance: csi 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO19 of instance: gpio3 1000 **ALT8** — Select mux mode: ALT8 mux port: ENET2_TX_CLK of instance: enet2 1001 **ALT9** — Select mux mode: ALT9 mux port: ENET2_REF_CLK2 of instance: enet2

11.6.35 SW_MUX_CTL_PAD_GPIO_EMC_34 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_34)

SW_MUX_CTL Register

Address: 401F_8000h base + 9Ch offset = 401F_809Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_34 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_EMC_34 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_EMC_34. 000 **ALT0** — Select mux mode: ALT0 mux port: SEMC_DATA12 of instance: semc 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXPWM3_PWMB02 of instance: flexpwm3 010 **ALT2** — Select mux mode: ALT2 mux port: USDHC1_VSELECT of instance: usdhc1 011 **ALT3** — Select mux mode: ALT3 mux port: SAI3_RX_SYNC of instance: sai3 100 **ALT4** — Select mux mode: ALT4 mux port: CSI_DATA19 of instance: csi 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO20 of instance: gpio3 1000 **ALT8** — Select mux mode: ALT8 mux port: ENET2_RX_ER of instance: enet2

11.6.36 SW_MUX_CTL_PAD_GPIO_EMC_35 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_35)

SW_MUX_CTL Register

Address: 401F_8000h base + A0h offset = 401F_80A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_35 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_EMC_35 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_EMC_35. 000 **ALT0** — Select mux mode: ALT0 mux port: SEMC_DATA13 of instance: semc 001 **ALT1** — Select mux mode: ALT1 mux port: XBAR1_INOUT18 of instance: xbar1 010 **ALT2** — Select mux mode: ALT2 mux port: GPT1_COMPARE1 of instance: gpt1 011 **ALT3** — Select mux mode: ALT3 mux port: SAI3_RX_BCLK of instance: sai3 100 **ALT4** — Select mux mode: ALT4 mux port: CSI_DATA18 of instance: csi 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO21 of instance: gpio3 110 **ALT6** — Select mux mode: ALT6 mux port: USDHC1_CD_B of instance: usdhc1 1000 **ALT8** — Select mux mode: ALT8 mux port: ENET2_RDATA00 of instance: enet2

11.6.37 SW_MUX_CTL_PAD_GPIO_EMC_36 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_36) pin 31

SW_MUX_CTL Register

Address: 401F_8000h base + A4h offset = 401F_80A4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_36 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_EMC_36 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_EMC_36. pin 31 000 **ALT0** — Select mux mode: ALT0 mux port: SEMC_DATA14 of instance: semc 001 **ALT1** — Select mux mode: ALT1 mux port: XBAR1_IN22 of instance: xbar1 010 **ALT2** — Select mux mode: ALT2 mux port: GPT1_COMPARE2 of instance: gpt1 011 **ALT3** — Select mux mode: ALT3 mux port: SAI3_TX_DATA of instance: sai3 100 **ALT4** — Select mux mode: ALT4 mux port: CSI_DATA17 of instance: csi 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO22 of instance: gpio3 110 **ALT6** — Select mux mode: ALT6 mux port: USDHC1_WP of instance: usdhc1 1000 **ALT8** — Select mux mode: ALT8 mux port: ENET2_RDATA01 of instance: enet2 1001 **ALT9** — Select mux mode: ALT9 mux port: FLEXCAN3_TX of instance: flexcan3/canfd

11.6.38 SW_MUX_CTL_PAD_GPIO_EMC_37 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_37) pin 30

SW_MUX_CTL Register

Address: 401F_8000h base + A8h offset = 401F_80A8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_37 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_EMC_37 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_EMC_37. pin 30 000 **ALT0** — Select mux mode: ALT0 mux port: SEMC_DATA15 of instance: semc 001 **ALT1** — Select mux mode: ALT1 mux port: XBAR1_IN23 of instance: xbar1 010 **ALT2** — Select mux mode: ALT2 mux port: GPT1_COMPARE3 of instance: gpt1 011 **ALT3** — Select mux mode: ALT3 mux port: SAI3_MCLK of instance: sai3 100 **ALT4** — Select mux mode: ALT4 mux port: CSI_DATA16 of instance: csi 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO23 of instance: gpio3 110 **ALT6** — Select mux mode: ALT6 mux port: USDHC2_WP of instance: usdhc2 1000 **ALT8** — Select mux mode: ALT8 mux port: ENET2_RX_EN of instance: enet2 1001 **ALT9** — Select mux mode: ALT9 mux port: FLEXCAN3_RX of instance: flexcan3/canfd

11.6.39 SW_MUX_CTL_PAD_GPIO_EMC_38 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_38)

SW_MUX_CTL Register

Address: 401F_8000h base + ACh offset = 401F_80ACh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_38 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_EMC_38 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_EMC_38. 000 **ALT0** — Select mux mode: ALT0 mux port: SEMC_DM01 of instance: semc 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXPWM1_PWMA03 of instance: flexpwm1 010 **ALT2** — Select mux mode: ALT2 mux port: LPUART8_TX of instance: lpuart8 011 **ALT3** — Select mux mode: ALT3 mux port: SAI3_TX_BCLK of instance: sai3 100 **ALT4** — Select mux mode: ALT4 mux port: CSI_FIELD of instance: csi 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO24 of instance: gpio3 110 **ALT6** — Select mux mode: ALT6 mux port: USDHC2_VSELECT of instance: usdhc2 1000 **ALT8** — Select mux mode: ALT8 mux port: ENET2_MDC of instance: enet2

11.6.40 SW_MUX_CTL_PAD_GPIO_EMC_39 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_39)

SW_MUX_CTL Register

Address: 401F_8000h base + B0h offset = 401F_80B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_39 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_EMC_39 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_EMC_39. 000 **ALT0** — Select mux mode: ALT0 mux port: SEMC_DQS of instance: semc 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXPWM1_PWMB03 of instance: flexpwm1 010 **ALT2** — Select mux mode: ALT2 mux port: LPUART8_RX of instance: lpuart8 011 **ALT3** — Select mux mode: ALT3 mux port: SAI3_TX_SYNC of instance: sai3 100 **ALT4** — Select mux mode: ALT4 mux port: WDOG1_WDOG_B of instance: wdog1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO25 of instance: gpio3 110 **ALT6** — Select mux mode: ALT6 mux port: USDHC2_CD_B of instance: usdhc2 1000 **ALT8** — Select mux mode: ALT8 mux port: ENET2_MDIO of instance: enet2 1001 **ALT9** — Select mux mode: ALT9 mux port: SEMC_DQS4 of instance: semc

11.6.41 SW_MUX_CTL_PAD_GPIO_EMC_40 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_40)

SW_MUX_CTL Register

Address: 401F_8000h base + B4h offset = 401F_80B4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

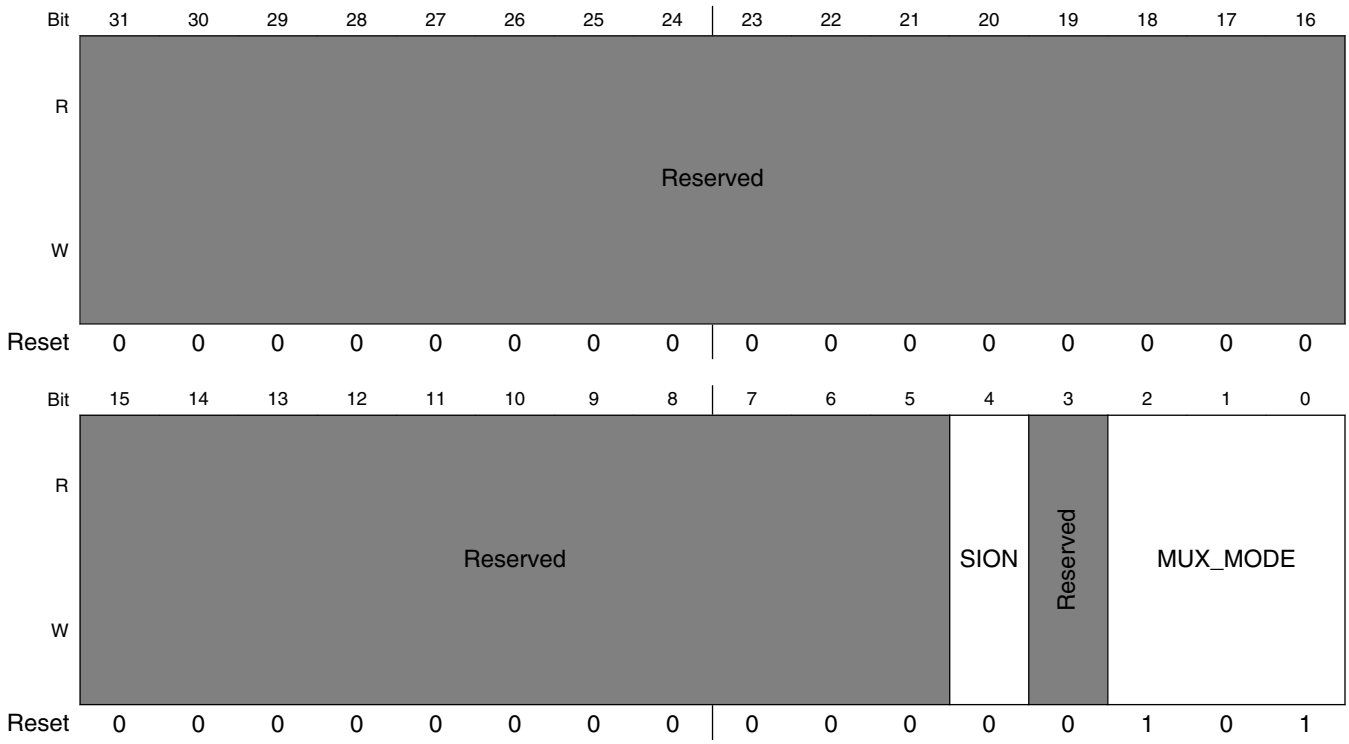
IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_40 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_EMC_40 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_EMC_40. 000 **ALT0** — Select mux mode: ALT0 mux port: SEMC_RDY of instance: semc 001 **ALT1** — Select mux mode: ALT1 mux port: GPT2_CAPTURE2 of instance: gpt2 010 **ALT2** — Select mux mode: ALT2 mux port: LPSPI1_PCS2 of instance: lpspi1 011 **ALT3** — Select mux mode: ALT3 mux port: USB_OTG2_OC of instance: usb 100 **ALT4** — Select mux mode: ALT4 mux port: ENET_MDC of instance: enet 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO26 of instance: gpio3 110 **ALT6** — Select mux mode: ALT6 mux port: USDHC2_RESET_B of instance: usdhc2 1001 **ALT9** — Select mux mode: ALT9 mux port: SEMC_CLK5 of instance: semc

11.6.42 SW_MUX_CTL_PAD_GPIO_EMC_41 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_41)

SW_MUX_CTL Register

Address: 401F_8000h base + B8h offset = 401F_80B8h



IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_41 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_EMC_41 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_EMC_41. 000 **ALT0** — Select mux mode: ALT0 mux port: SEMC_CSX00 of instance: semc 001 **ALT1** — Select mux mode: ALT1 mux port: GPT2_CAPTURE1 of instance: gpt2 010 **ALT2** — Select mux mode: ALT2 mux port: LPSPI1_PCS3 of instance: lpspi1

Table continues on the next page...

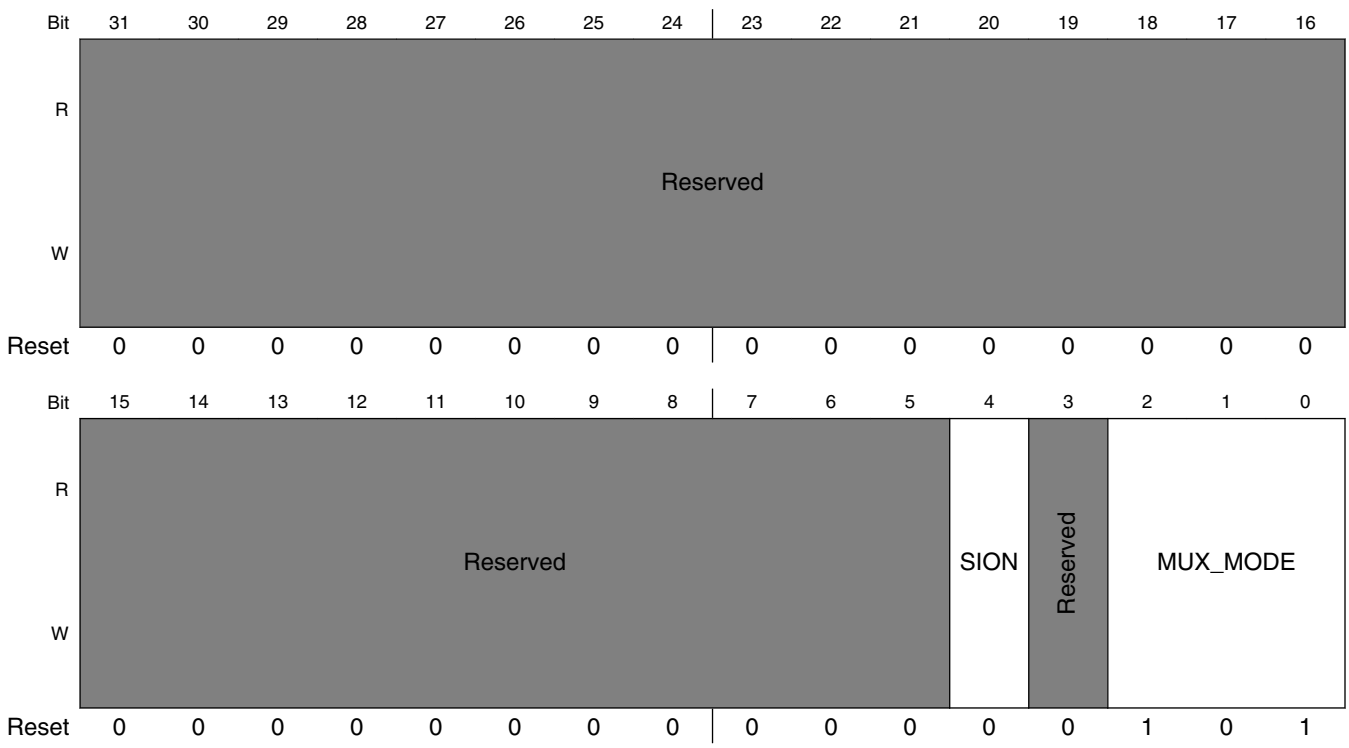
IOMUXC_SW_MUX_CTL_PAD_GPIO_EMC_41 field descriptions (continued)

Field	Description
	011 **ALT3** — Select mux mode: ALT3 mux port: USB_OTG2_PWR of instance: usb 100 **ALT4** — Select mux mode: ALT4 mux port: ENET_MDIO of instance: enet 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO27 of instance: gpio3 110 **ALT6** — Select mux mode: ALT6 mux port: USDHC1_VSELECT of instance: usdhc1

11.6.43 SW_MUX_CTL_PAD_GPIO_AD_B0_00 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_00)

SW_MUX_CTL Register

Address: 401F_8000h base + BCh offset = 401F_80BCh



IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_00 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality.

Table continues on the next page...

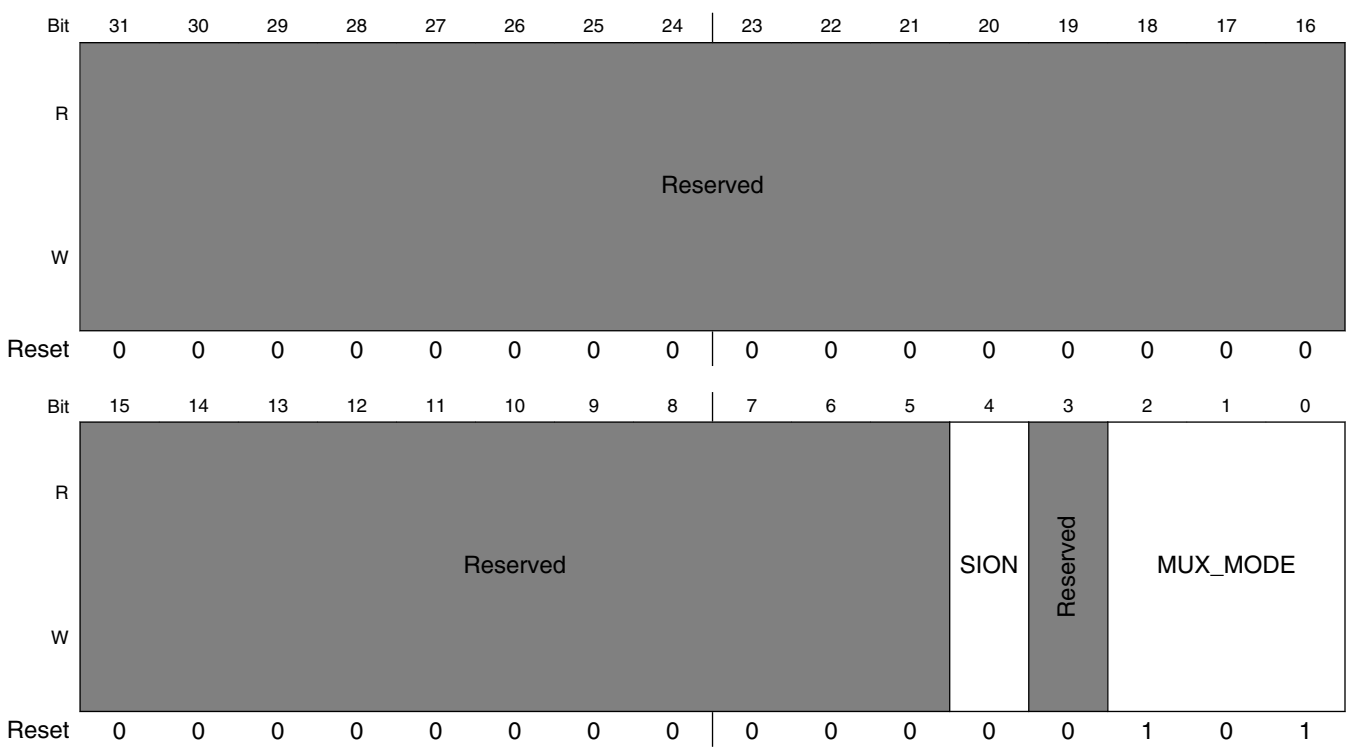
IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_00 field descriptions (continued)

Field	Description
	1 **ENABLED** — Force input path of pad GPIO_AD_B0_00 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_AD_B0_00. 000 **ALT0** — Select mux mode: ALT0 mux port: FLEXPWM2_PWMA03 of instance: flexpwm2 001 **ALT1** — Select mux mode: ALT1 mux port: XBAR1_INOUT14 of instance: xbar1 010 **ALT2** — Select mux mode: ALT2 mux port: REF_CLK_32K of instance: xtalosc 011 **ALT3** — Select mux mode: ALT3 mux port: USB_OTG2_ID of instance: usb 100 **ALT4** — Select mux mode: ALT4 mux port: LPI2C1_SCLS of instance: lpi2c1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO00 of instance: gpio1 110 **ALT6** — Select mux mode: ALT6 mux port: USDHC1_RESET_B of instance: usdhc1 111 **ALT7** — Select mux mode: ALT7 mux port: LPSPI3_SCK of instance: lpspi3

11.6.44 SW_MUX_CTL_PAD_GPIO_AD_B0_01 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_01)

SW_MUX_CTL Register

Address: 401F_8000h base + C0h offset = 401F_80C0h



IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_01 field descriptions

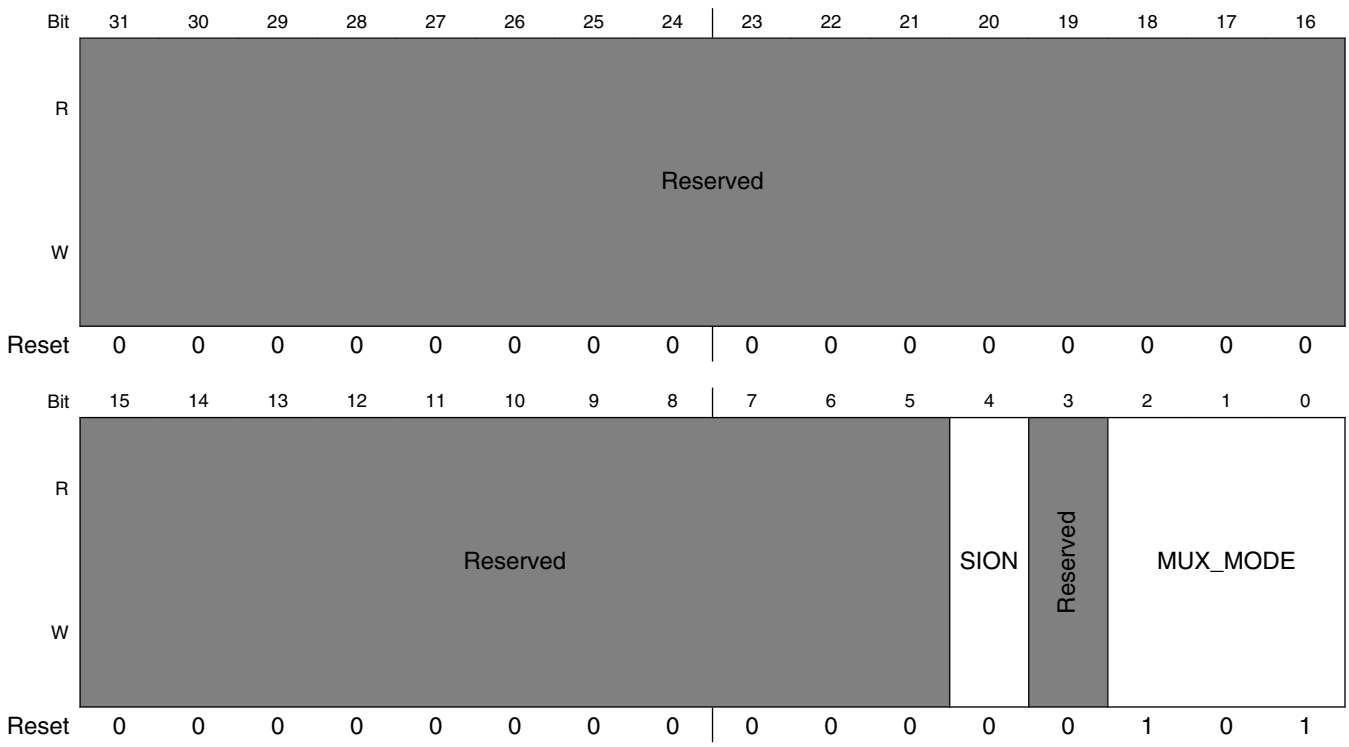
Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_AD_B0_01 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_AD_B0_01. 000 **ALT0** — Select mux mode: ALT0 mux port: FLEXPWM2_PWMB03 of instance: flexpwm2 001 **ALT1** — Select mux mode: ALT1 mux port: XBAR1_INOUT15 of instance: xbar1 010 **ALT2** — Select mux mode: ALT2 mux port: REF_CLK_24M of instance: anatop 011 **ALT3** — Select mux mode: ALT3 mux port: USB_OTG1_ID of instance: anatop 100 **ALT4** — Select mux mode: ALT4 mux port: LPI2C1_SDAS of instance: lpi2c1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO01 of instance: gpio1 110 **ALT6** — Select mux mode: ALT6 mux port: EWM_OUT_B of instance: ewm 111 **ALT7** — Select mux mode: ALT7 mux port: LPSPI3_SDO of instance: lpspi3

11.6.45 SW_MUX_CTL_PAD_GPIO_AD_B0_02 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_02)

pin 1

SW_MUX_CTL Register

Address: 401F_8000h base + C4h offset = 401F_80C4h



IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_02 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_AD_B0_02 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_AD_B0_02. pin 1 000 **ALT0** — Select mux mode: ALT0 mux port: FLEXCAN2_TX of instance: flexcan2 001 **ALT1** — Select mux mode: ALT1 mux port: XBAR1_INOUT16 of instance: xbar1

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_02 field descriptions (continued)

Field	Description
	010 **ALT2** — Select mux mode: ALT2 mux port: LPUART6_TX of instance: lpuart6 011 **ALT3** — Select mux mode: ALT3 mux port: USB_OTG1_PWR of instance: usb 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXPWM1_PWMX00 of instance: flexpwm1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO02 of instance: gpio1 110 **ALT6** — Select mux mode: ALT6 mux port: LPI2C1_HREQ of instance: lpi2c1 111 **ALT7** — Select mux mode: ALT7 mux port: LPSPI3_SDI of instance: lpspi3

11.6.46 SW_MUX_CTL_PAD_GPIO_AD_B0_03 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_03)

SW_MUX_CTL Register

Address: 401F_8000h base + C8h offset = 401F_80C8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	Reserved	MUX_MODE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_03 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field.

Table continues on the next page...

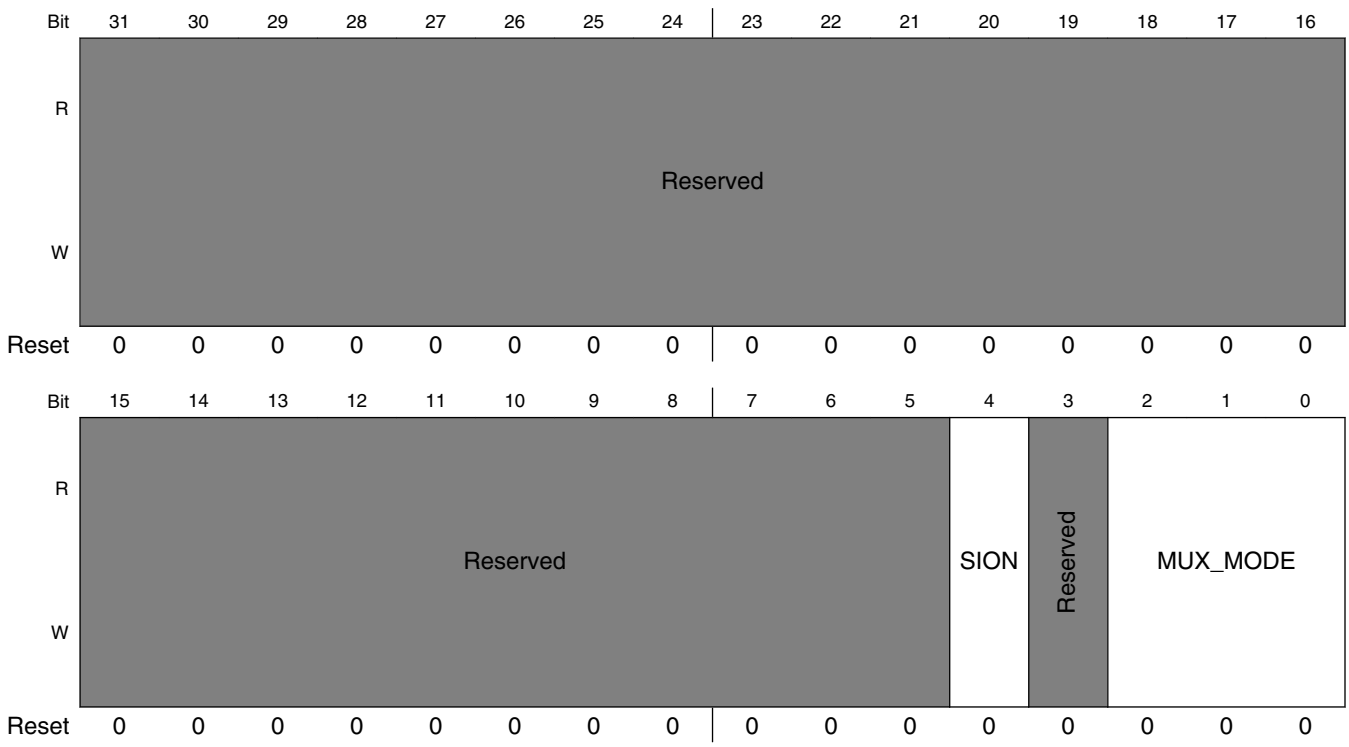
IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_03 field descriptions (continued)

Field	Description
	Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_AD_B0_03 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_AD_B0_03. pin 0 000 **ALT0** — Select mux mode: ALT0 mux port: FLEXCAN2_RX of instance: flexcan2 001 **ALT1** — Select mux mode: ALT1 mux port: XBAR1_INOUT17 of instance: xbar1 010 **ALT2** — Select mux mode: ALT2 mux port: LPUART6_RX of instance: lpuart6 011 **ALT3** — Select mux mode: ALT3 mux port: USB_OTG1_OC of instance: usb 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXPWM1_PWMX01 of instance: flexpwm1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO03 of instance: gpio1 110 **ALT6** — Select mux mode: ALT6 mux port: REF_CLK_24M of instance: anatop 111 **ALT7** — Select mux mode: ALT7 mux port: LPSPI3_PCS0 of instance: lpspi3

11.6.47 SW_MUX_CTL_PAD_GPIO_AD_B0_04 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_04)

SW_MUX_CTL Register

Address: 401F_8000h base + CCh offset = 401F_80CCh



IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_04 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_AD_B0_04 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_AD_B0_04. 000 **ALT0** — Select mux mode: ALT0 mux port: SRC_BOOT_MODE00 of instance: src 001 **ALT1** — Select mux mode: ALT1 mux port: MQS_RIGHT of instance: mqs

Table continues on the next page...

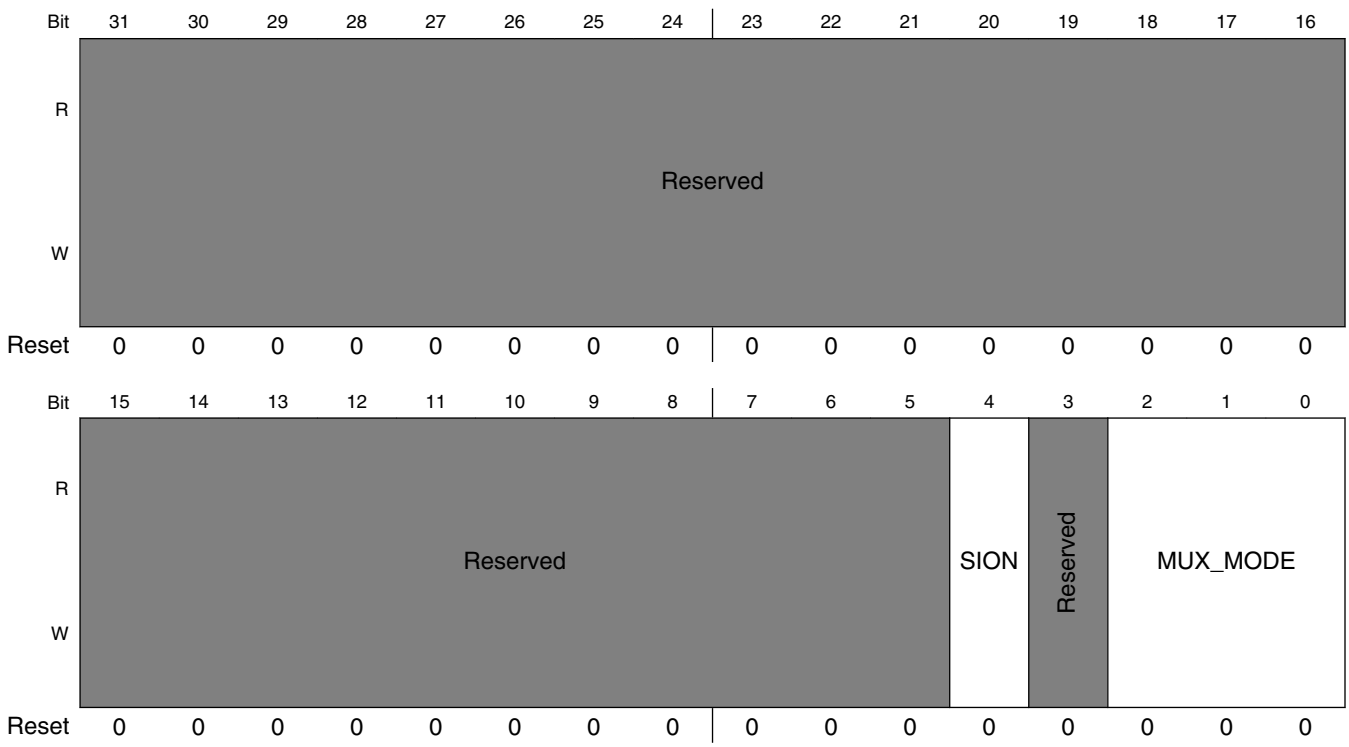
IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_04 field descriptions (continued)

Field	Description
	010 **ALT2** — Select mux mode: ALT2 mux port: ENET_TX_DATA03 of instance: enet 011 **ALT3** — Select mux mode: ALT3 mux port: SAI2_TX_SYNC of instance: sai2 100 **ALT4** — Select mux mode: ALT4 mux port: CSI_DATA09 of instance: csi 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO04 of instance: gpio1 110 **ALT6** — Select mux mode: ALT6 mux port: PIT_TRIGGER00 of instance: pit 111 **ALT7** — Select mux mode: ALT7 mux port: LPSPI3_PCS1 of instance: lpspi3

11.6.48 SW_MUX_CTL_PAD_GPIO_AD_B0_05 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_05)

SW_MUX_CTL Register

Address: 401F_8000h base + D0h offset = 401F_80D0h



IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_05 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field.

Table continues on the next page...

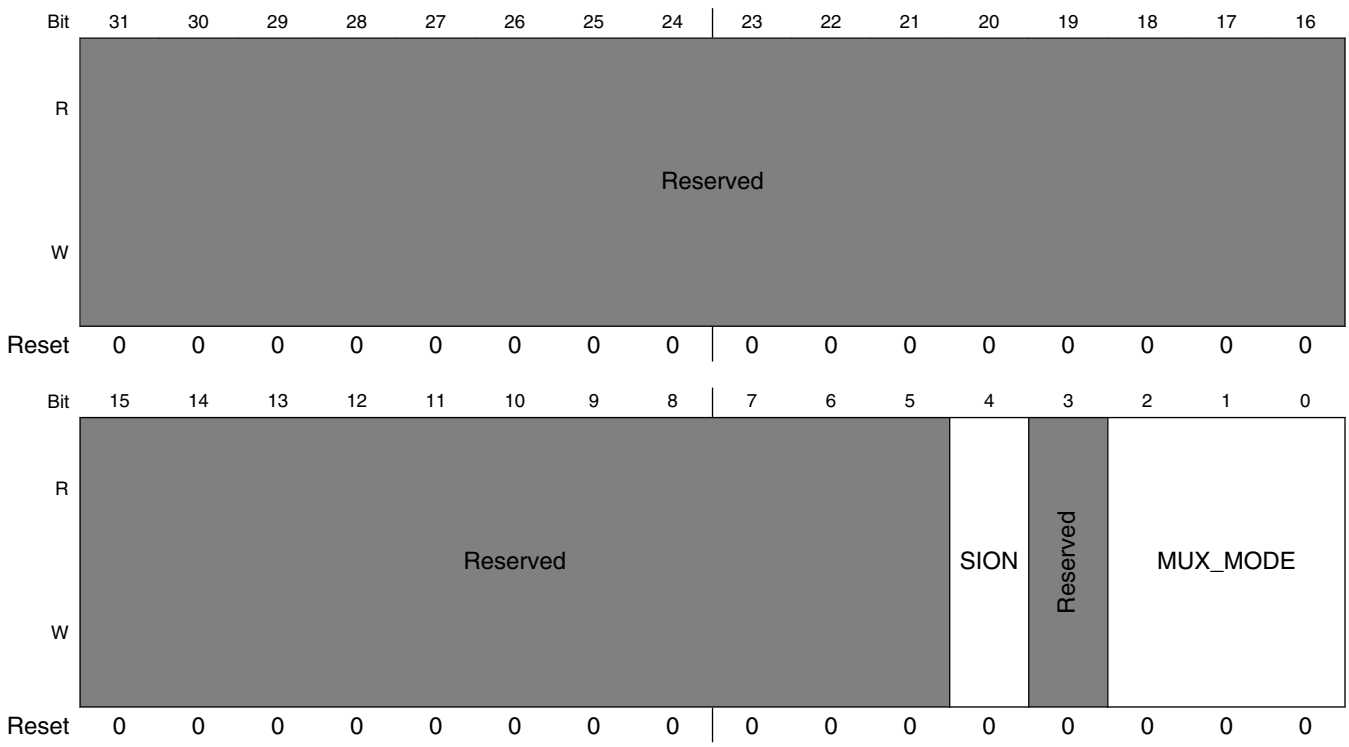
IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_05 field descriptions (continued)

Field	Description
	Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_AD_B0_05 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_AD_B0_05. 000 **ALT0** — Select mux mode: ALT0 mux port: SRC_BOOT_MODE01 of instance: src 001 **ALT1** — Select mux mode: ALT1 mux port: MQS_LEFT of instance: mqs 010 **ALT2** — Select mux mode: ALT2 mux port: ENET_TX_DATA02 of instance: enet 011 **ALT3** — Select mux mode: ALT3 mux port: SAI2_TX_BCLK of instance: sai2 100 **ALT4** — Select mux mode: ALT4 mux port: CSI_DATA08 of instance: csi 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO05 of instance: gpio1 110 **ALT6** — Select mux mode: ALT6 mux port: XBAR1_INOUT17 of instance: xbar1 111 **ALT7** — Select mux mode: ALT7 mux port: LPSPI3_PCS2 of instance: lpspi3

11.6.49 SW_MUX_CTL_PAD_GPIO_AD_B0_06 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_06)

SW_MUX_CTL Register

Address: 401F_8000h base + D4h offset = 401F_80D4h



IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_06 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_AD_B0_06 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_AD_B0_06. 000 **ALT0** — Select mux mode: ALT0 mux port: JTAG_TMS of instance: jtag_mux 001 **ALT1** — Select mux mode: ALT1 mux port: GPT2_COMPARE1 of instance: gpt2

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_06 field descriptions (continued)

Field	Description
	010 **ALT2** — Select mux mode: ALT2 mux port: ENET_RX_CLK of instance: enet 011 **ALT3** — Select mux mode: ALT3 mux port: SAI2_RX_BCLK of instance: sai2 100 **ALT4** — Select mux mode: ALT4 mux port: CSI_DATA07 of instance: csi 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO06 of instance: gpio1 110 **ALT6** — Select mux mode: ALT6 mux port: XBAR1_INOUT18 of instance: xbar1 111 **ALT7** — Select mux mode: ALT7 mux port: LPSPI3_PCS3 of instance: lpspi3

11.6.50 SW_MUX_CTL_PAD_GPIO_AD_B0_07 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_07)

SW_MUX_CTL Register

Address: 401F_8000h base + D8h offset = 401F_80D8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	Reserved	MUX_MODE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_07 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field.

Table continues on the next page...

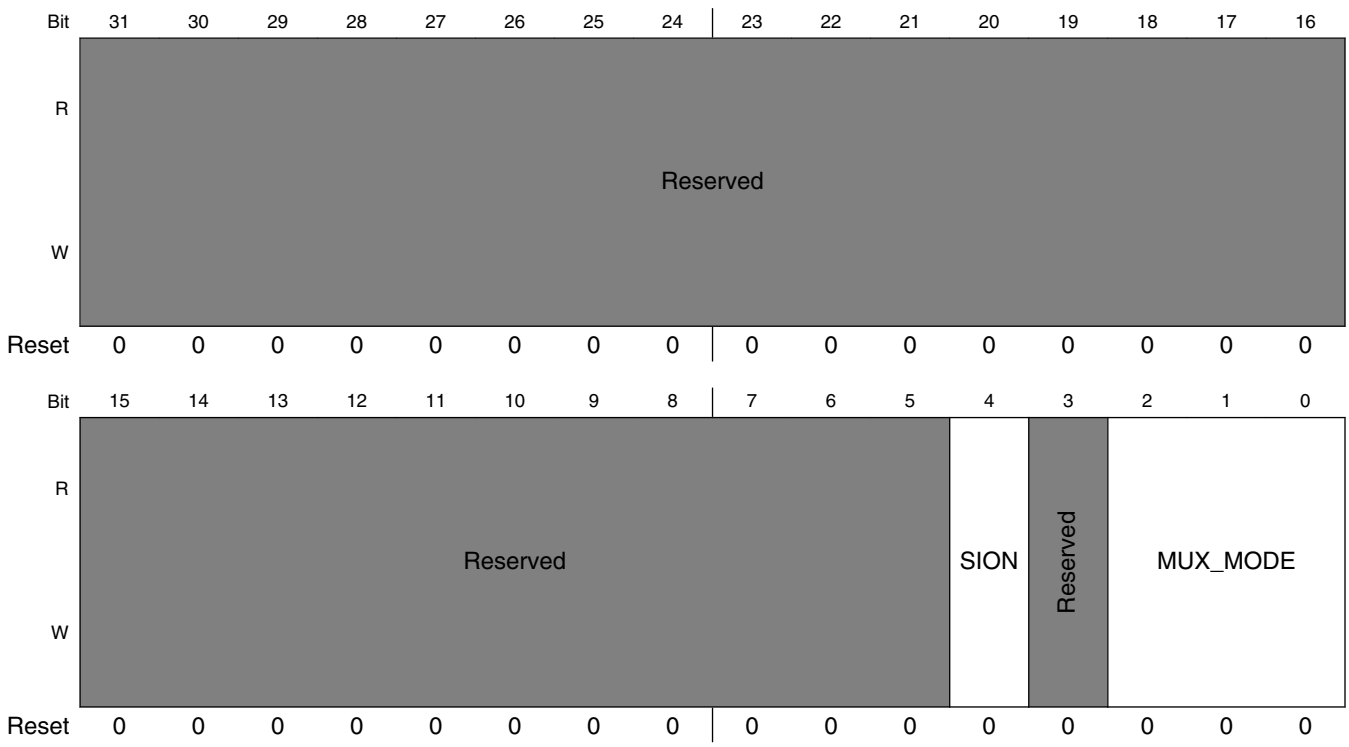
IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_07 field descriptions (continued)

Field	Description
	Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_AD_B0_07 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_AD_B0_07. 000 **ALT0** — Select mux mode: ALT0 mux port: JTAG_TCK of instance: jtag_mux 001 **ALT1** — Select mux mode: ALT1 mux port: GPT2_COMPARE2 of instance: gpt2 010 **ALT2** — Select mux mode: ALT2 mux port: ENET_TX_ER of instance: enet 011 **ALT3** — Select mux mode: ALT3 mux port: SAI2_RX_SYNC of instance: sai2 100 **ALT4** — Select mux mode: ALT4 mux port: CSI_DATA06 of instance: csi 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO07 of instance: gpio1 110 **ALT6** — Select mux mode: ALT6 mux port: XBAR1_INOUT19 of instance: xbar1 111 **ALT7** — Select mux mode: ALT7 mux port: ENET_1588_EVENT3_OUT of instance: enet

11.6.51 SW_MUX_CTL_PAD_GPIO_AD_B0_08 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_08)

SW_MUX_CTL Register

Address: 401F_8000h base + DCh offset = 401F_80DCh



IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_08 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_AD_B0_08 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_AD_B0_08. 000 **ALT0** — Select mux mode: ALT0 mux port: JTAG_MOD of instance: jtag_mux 001 **ALT1** — Select mux mode: ALT1 mux port: GPT2_COMPARE3 of instance: gpt2

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_08 field descriptions (continued)

Field	Description
	010 **ALT2** — Select mux mode: ALT2 mux port: ENET_RX_DATA03 of instance: enet 011 **ALT3** — Select mux mode: ALT3 mux port: SAI2_RX_DATA of instance: sai2 100 **ALT4** — Select mux mode: ALT4 mux port: CSI_DATA05 of instance: csi 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO08 of instance: gpio1 110 **ALT6** — Select mux mode: ALT6 mux port: XBAR1_IN20 of instance: xbar1 111 **ALT7** — Select mux mode: ALT7 mux port: ENET_1588_EVENT3_IN of instance: enet

11.6.52 SW_MUX_CTL_PAD_GPIO_AD_B0_09 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_09)

SW_MUX_CTL Register

Address: 401F_8000h base + E0h offset = 401F_80E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_09 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_AD_B0_09 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_AD_B0_09. 000 **ALT0** — Select mux mode: ALT0 mux port: JTAG_TDI of instance: jtag_mux 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXPWM2_PWMA03 of instance: flexpwm2 010 **ALT2** — Select mux mode: ALT2 mux port: ENET_RX_DATA02 of instance: enet 011 **ALT3** — Select mux mode: ALT3 mux port: SAI2_TX_DATA of instance: sai2 100 **ALT4** — Select mux mode: ALT4 mux port: CSI_DATA04 of instance: csi 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO09 of instance: gpio1 110 **ALT6** — Select mux mode: ALT6 mux port: XBAR1_IN21 of instance: xbar1

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_09 field descriptions (continued)

Field	Description
	111 **ALT7** — Select mux mode: ALT7 mux port: GPT2_CLK of instance: gpt2 1001 **ALT9** — Select mux mode: ALT9 mux port: SEMC_DQS4 of instance: semc

11.6.53 SW_MUX_CTL_PAD_GPIO_AD_B0_10 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_10)

SW_MUX_CTL Register

Address: 401F_8000h base + E4h offset = 401F_80E4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_10 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_AD_B0_10 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_AD_B0_10. 000 **ALT0** — Select mux mode: ALT0 mux port: JTAG_TDO of instance: jtag_mux 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXPWM1_PWMA03 of instance: flexpwm1 010 **ALT2** — Select mux mode: ALT2 mux port: ENET_CRS of instance: enet 011 **ALT3** — Select mux mode: ALT3 mux port: SAI2_MCLK of instance: sai2 100 **ALT4** — Select mux mode: ALT4 mux port: CSI_DATA03 of instance: csi 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO10 of instance: gpio1 110 **ALT6** — Select mux mode: ALT6 mux port: XBAR1_IN22 of instance: xbar1 111 **ALT7** — Select mux mode: ALT7 mux port: ENET_1588_EVENT0_OUT of instance: enet 1000 **ALT8** — Select mux mode: ALT8 mux port: FLEXCAN3_TX of instance: flexcan3/canfd 1001 **ALT9** — Select mux mode: ALT9 mux port: ARM_TRACE_SWO of instance: cm7_mx6rt

11.6.54 SW_MUX_CTL_PAD_GPIO_AD_B0_11 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_11)

SW_MUX_CTL Register

Address: 401F_8000h base + E8h offset = 401F_80E8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

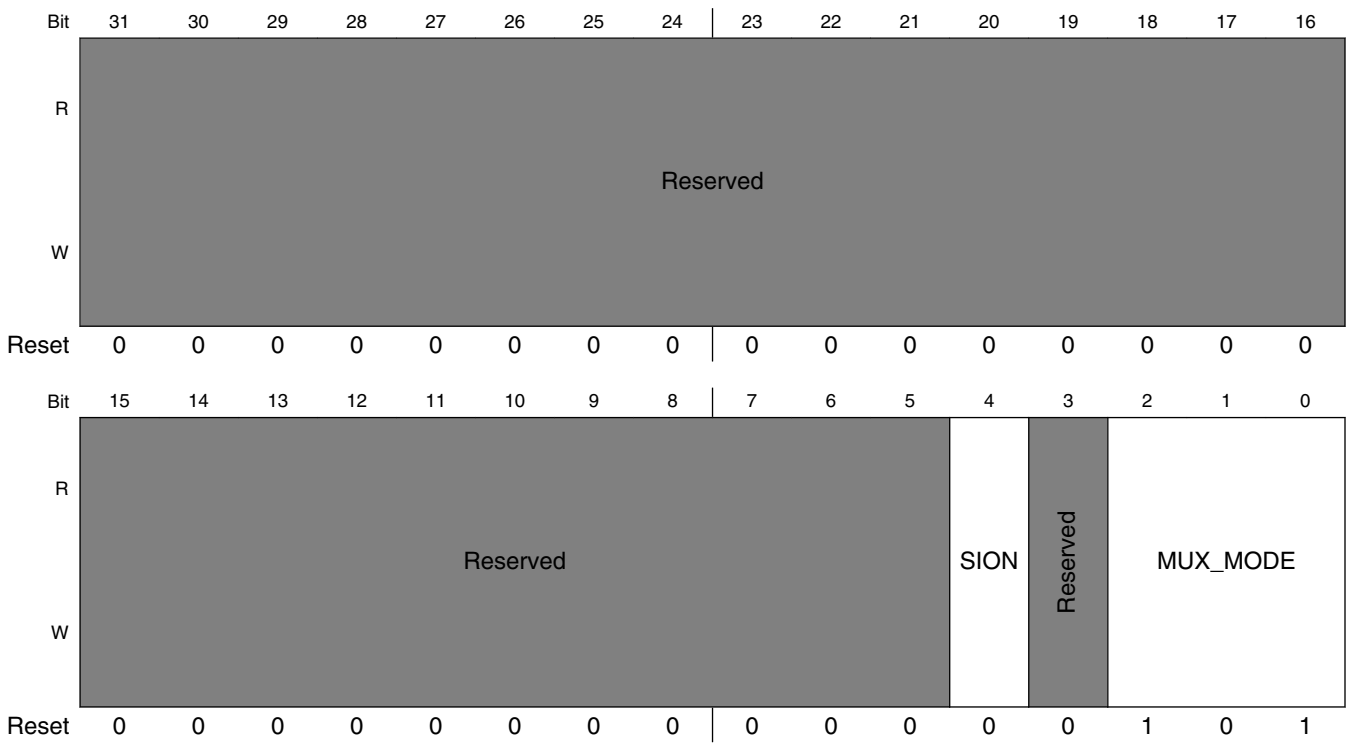
IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_11 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_AD_B0_11 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_AD_B0_11. 000 **ALT0** — Select mux mode: ALT0 mux port: JTAG_TRSTB of instance: jtag_mux 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXPWM1_PWMB03 of instance: flexpwm1 010 **ALT2** — Select mux mode: ALT2 mux port: ENET_COL of instance: enet 011 **ALT3** — Select mux mode: ALT3 mux port: WDOG1_WDOG_B of instance: wdog1 100 **ALT4** — Select mux mode: ALT4 mux port: CSI_DATA02 of instance: csi 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO11 of instance: gpio1 110 **ALT6** — Select mux mode: ALT6 mux port: XBAR1_IN23 of instance: xbar1 111 **ALT7** — Select mux mode: ALT7 mux port: ENET_1588_EVENT0_IN of instance: enet 1000 **ALT8** — Select mux mode: ALT8 mux port: FLEXCAN3_RX of instance: flexcan3/canfd 1001 **ALT9** — Select mux mode: ALT9 mux port: SEMC_CLK6 of instance: semc

11.6.55 SW_MUX_CTL_PAD_GPIO_AD_B0_12 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_12) pin 24

SW_MUX_CTL Register

Address: 401F_8000h base + ECh offset = 401F_80ECh



IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_12 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_AD_B0_12 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_AD_B0_12. pin 24 000 **ALT0** — Select mux mode: ALT0 mux port: LPI2C4_SCL of instance: lpi2c4 001 **ALT1** — Select mux mode: ALT1 mux port: CCM_PMIC_READY of instance: ccm

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_12 field descriptions (continued) pin 24

Field	Description
	010 **ALT2** — Select mux mode: ALT2 mux port: LPUART1_TX of instance: lpuart1 011 **ALT3** — Select mux mode: ALT3 mux port: WDOG2_WDOG_B of instance: wdog2 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXPWM1_PWMX02 of instance: flexpwm1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO12 of instance: gpio1 110 **ALT6** — Select mux mode: ALT6 mux port: ENET_1588_EVENT1_OUT of instance: enet 111 **ALT7** — Select mux mode: ALT7 mux port: NMI_GLUE_NMI of instance: nmi_glue

11.6.56 SW_MUX_CTL_PAD_GPIO_AD_B0_13 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_13) pin 25

SW_MUX_CTL Register

Address: 401F_8000h base + F0h offset = 401F_80F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	Reserved	MUX_MODE		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_13 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field.

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_13 field descriptions (continued) pin 25

Field	Description
	Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_AD_B0_13 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_AD_B0_13. pin 25 000 **ALT0** — Select mux mode: ALT0 mux port: LPI2C4_SDA of instance: lpi2c4 001 **ALT1** — Select mux mode: ALT1 mux port: GPT1_CLK of instance: gpt1 010 **ALT2** — Select mux mode: ALT2 mux port: LPUART1_RX of instance: lpuart1 011 **ALT3** — Select mux mode: ALT3 mux port: EWM_OUT_B of instance: ewm 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXPWM1_PWMX03 of instance: flexpwm1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO13 of instance: gpio1 110 **ALT6** — Select mux mode: ALT6 mux port: ENET_1588_EVENT1_IN of instance: enet 111 **ALT7** — Select mux mode: ALT7 mux port: REF_CLK_24M of instance: anatop

11.6.57 SW_MUX_CTL_PAD_GPIO_AD_B0_14 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_14)

SW_MUX_CTL Register

Address: 401F_8000h base + F4h offset = 401F_80F4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_14 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_AD_B0_14 0 **DISABLED** — Input Path is determined by functionality

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_14 field descriptions (continued)

Field	Description
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_AD_B0_14. 000 **ALT0** — Select mux mode: ALT0 mux port: USB_OTG2_OC of instance: usb 001 **ALT1** — Select mux mode: ALT1 mux port: XBAR1_IN24 of instance: xbar1 010 **ALT2** — Select mux mode: ALT2 mux port: LPUART1_CTS_B of instance: lpuart1 011 **ALT3** — Select mux mode: ALT3 mux port: ENET_1588_EVENT0_OUT of instance: enet 100 **ALT4** — Select mux mode: ALT4 mux port: CSI_VSYNC of instance: csi 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO14 of instance: gpio1 110 **ALT6** — Select mux mode: ALT6 mux port: FLEXCAN2_TX of instance: flexcan2 1000 **ALT8** — Select mux mode: ALT8 mux port: FLEXCAN3_TX of instance: flexcan3/canfd

11.6.58 SW_MUX_CTL_PAD_GPIO_AD_B0_15 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_15)

SW_MUX_CTL Register

Address: 401F_8000h base + F8h offset = 401F_80F8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_15 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_AD_B0_15 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_AD_B0_15. 000 **ALT0** — Select mux mode: ALT0 mux port: USB_OTG2_PWR of instance: usb 001 **ALT1** — Select mux mode: ALT1 mux port: XBAR1_IN25 of instance: xbar1

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B0_15 field descriptions (continued)

Field	Description
	010 **ALT2** — Select mux mode: ALT2 mux port: LPUART1_RTS_B of instance: lpuart1 011 **ALT3** — Select mux mode: ALT3 mux port: ENET_1588_EVENT0_IN of instance: enet 100 **ALT4** — Select mux mode: ALT4 mux port: CSI_HSYNC of instance: csi 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO15 of instance: gpio1 110 **ALT6** — Select mux mode: ALT6 mux port: FLEXCAN2_RX of instance: flexcan2 111 **ALT7** — Select mux mode: ALT7 mux port: WDOG1_WDOG_RST_B_DEB of instance: wdog1 1000 **ALT8** — Select mux mode: ALT8 mux port: FLEXCAN3_RX of instance: flexcan3/canfd

11.6.59 SW_MUX_CTL_PAD_GPIO_AD_B1_00 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B1_00)

pin 19

SW_MUX_CTL Register

Address: 401F_8000h base + FCh offset = 401F_80FCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B1_00 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_AD_B1_00 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_AD_B1_00. pin 19 000 **ALT0** — Select mux mode: ALT0 mux port: USB_OTG2_ID of instance: anatop 001 **ALT1** — Select mux mode: ALT1 mux port: QTIMER3_TIMER0 of instance: qtimer3 010 **ALT2** — Select mux mode: ALT2 mux port: LPUART2_CTS_B of instance: lpuart2 011 **ALT3** — Select mux mode: ALT3 mux port: LPI2C1_SCL of instance: lpi2c1 100 **ALT4** — Select mux mode: ALT4 mux port: WDOG1_B of instance: wdog1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO16 of instance: gpio1

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B1_00 field descriptions (continued) pin 19

Field	Description
	110 **ALT6** — Select mux mode: ALT6 mux port: USDHC1_WP of instance: usdhc1 111 **ALT7** — Select mux mode: ALT7 mux port: KPP_ROW07 of instance: kpp 1000 **ALT8** — Select mux mode: ALT8 mux port: ENET2_1588_EVENT0_OUT of instance: enet2 1001 **ALT9** — Select mux mode: ALT9 mux port: FLEXIO3_FLEXIO00 of instance: flexio3

11.6.60 SW_MUX_CTL_PAD_GPIO_AD_B1_01 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B1_01) pin 18

SW_MUX_CTL Register

Address: 401F_8000h base + 100h offset = 401F_8100h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B1_01 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_AD_B1_01 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_AD_B1_01. pin 18 000 **ALT0** — Select mux mode: ALT0 mux port: USB_OTG1_PWR of instance: usb 001 **ALT1** — Select mux mode: ALT1 mux port: QTIMER3_TIMER1 of instance: qtimer3 010 **ALT2** — Select mux mode: ALT2 mux port: LPUART2_RTS_B of instance: lpuart2 011 **ALT3** — Select mux mode: ALT3 mux port: LPI2C1_SDA of instance: lpi2c1 100 **ALT4** — Select mux mode: ALT4 mux port: CCM_PMIC_READY of instance: ccm 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO17 of instance: gpio1 110 **ALT6** — Select mux mode: ALT6 mux port: USDHC1_VSELECT of instance: usdhc1 111 **ALT7** — Select mux mode: ALT7 mux port: KPP_COL07 of instance: kpp 1000 **ALT8** — Select mux mode: ALT8 mux port: ENET2_1588_EVENT0_IN of instance: enet2 1001 **ALT9** — Select mux mode: ALT9 mux port: FLEXIO3_FLEXIO01 of instance: flexio3

11.6.61 SW_MUX_CTL_PAD_GPIO_AD_B1_02 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B1_02)

pin 14

SW_MUX_CTL Register

Address: 401F_8000h base + 104h offset = 401F_8104h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B1_02 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_AD_B1_02 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_AD_B1_02. pin 14 000 **ALT0** — Select mux mode: ALT0 mux port: USB_OTG1_ID of instance: anatop 001 **ALT1** — Select mux mode: ALT1 mux port: QTIMER3_TIMER2 of instance: qtimer3 010 **ALT2** — Select mux mode: ALT2 mux port: LPUART2_TX of instance: lpuart2 011 **ALT3** — Select mux mode: ALT3 mux port: SPDIF_OUT of instance: spdif 100 **ALT4** — Select mux mode: ALT4 mux port: ENET_1588_EVENT2_OUT of instance: enet 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO18 of instance: gpio1 110 **ALT6** — Select mux mode: ALT6 mux port: USDHC1_CD_B of instance: usdhc1 111 **ALT7** — Select mux mode: ALT7 mux port: KPP_ROW06 of instance: kpp 1000 **ALT8** — Select mux mode: ALT8 mux port: GPT2_CLK of instance: gpt2 1001 **ALT9** — Select mux mode: ALT9 mux port: FLEXIO3_FLEXIO02 of instance: flexio3

11.6.62 SW_MUX_CTL_PAD_GPIO_AD_B1_03 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B1_03)

pin 15

SW_MUX_CTL Register

Address: 401F_8000h base + 108h offset = 401F_8108h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B1_03 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_AD_B1_03 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_AD_B1_03. pin 15 000 **ALT0** — Select mux mode: ALT0 mux port: USB_OTG1_OC of instance: usb 001 **ALT1** — Select mux mode: ALT1 mux port: QTIMER3_TIMER3 of instance: qtimer3 010 **ALT2** — Select mux mode: ALT2 mux port: LPUART2_RX of instance: lpuart2 011 **ALT3** — Select mux mode: ALT3 mux port: SPDIF_IN of instance: spdif 100 **ALT4** — Select mux mode: ALT4 mux port: ENET_1588_EVENT2_IN of instance: enet 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO19 of instance: gpio1 110 **ALT6** — Select mux mode: ALT6 mux port: USDHC2_CD_B of instance: usdhc2 111 **ALT7** — Select mux mode: ALT7 mux port: KPP_COL06 of instance: kpp 1000 **ALT8** — Select mux mode: ALT8 mux port: GPT2_CAPTURE1 of instance: gpt2 1001 **ALT9** — Select mux mode: ALT9 mux port: FLEXIO3_FLEXIO03 of instance: flexio3

11.6.63 SW_MUX_CTL_PAD_GPIO_AD_B1_04 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B1_04)

pin 40 (T4.1)

SW_MUX_CTL Register

Address: 401F_8000h base + 10Ch offset = 401F_810Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B1_04 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_AD_B1_04 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_AD_B1_04. pin 40 (T4.1) 000 **ALT0** — Select mux mode: ALT0 mux port: FLEXSPIB_DATA03 of instance: flexspi 001 **ALT1** — Select mux mode: ALT1 mux port: ENET_MDC of instance: enet 010 **ALT2** — Select mux mode: ALT2 mux port: LPUART3_CTS_B of instance: lpuart3 011 **ALT3** — Select mux mode: ALT3 mux port: SPDIF_SR_CLK of instance: spdif 100 **ALT4** — Select mux mode: ALT4 mux port: CSI_PIXCLK of instance: csi 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO20 of instance: gpio1 110 **ALT6** — Select mux mode: ALT6 mux port: USDHC2_DATA0 of instance: usdhc2 111 **ALT7** — Select mux mode: ALT7 mux port: KPP_ROW05 of instance: kpp 1000 **ALT8** — Select mux mode: ALT8 mux port: GPT2_CAPTURE2 of instance: gpt2 1001 **ALT9** — Select mux mode: ALT9 mux port: FLEXIO3_FLEXIO04 of instance: flexio3

11.6.64 SW_MUX_CTL_PAD_GPIO_AD_B1_05 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B1_05)

pin 41 (T4.1)

SW_MUX_CTL Register

Address: 401F_8000h base + 110h offset = 401F_8110h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B1_05 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_AD_B1_05 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_AD_B1_05. pin 41 (T4.1) 000 **ALT0** — Select mux mode: ALT0 mux port: FLEXSPIB_DATA02 of instance: flexspi 001 **ALT1** — Select mux mode: ALT1 mux port: ENET_MDIO of instance: enet 010 **ALT2** — Select mux mode: ALT2 mux port: LPUART3_RTS_B of instance: lpuart3 011 **ALT3** — Select mux mode: ALT3 mux port: SPDIF_OUT of instance: spdif 100 **ALT4** — Select mux mode: ALT4 mux port: CSI_MCLK of instance: csi 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO21 of instance: gpio1 110 **ALT6** — Select mux mode: ALT6 mux port: USDHC2_DATA1 of instance: usdhc2 111 **ALT7** — Select mux mode: ALT7 mux port: KPP_COL05 of instance: kpp 1000 **ALT8** — Select mux mode: ALT8 mux port: GPT2_COMPARE1 of instance: gpt2 1001 **ALT9** — Select mux mode: ALT9 mux port: FLEXIO3_FLEXIO05 of instance: flexio3

11.6.65 SW_MUX_CTL_PAD_GPIO_AD_B1_06 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B1_06)

pin 17

SW_MUX_CTL Register

Address: 401F_8000h base + 114h offset = 401F_8114h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B1_06 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_AD_B1_06 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_AD_B1_06. pin 17 000 **ALT0** — Select mux mode: ALT0 mux port: FLEXSPIB_DATA01 of instance: flexspi 001 **ALT1** — Select mux mode: ALT1 mux port: LPI2C3_SDA of instance: lpi2c3 010 **ALT2** — Select mux mode: ALT2 mux port: LPUART3_TX of instance: lpuart3 011 **ALT3** — Select mux mode: ALT3 mux port: SPDIF_LOCK of instance: spdif 100 **ALT4** — Select mux mode: ALT4 mux port: CSI_VSYNC of instance: csi 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO22 of instance: gpio1 110 **ALT6** — Select mux mode: ALT6 mux port: USDHC2_DATA2 of instance: usdhc2 111 **ALT7** — Select mux mode: ALT7 mux port: KPP_ROW04 of instance: kpp 1000 **ALT8** — Select mux mode: ALT8 mux port: GPT2_COMPARE2 of instance: gpt2 1001 **ALT9** — Select mux mode: ALT9 mux port: FLEXIO3_FLEXIO06 of instance: flexio3

11.6.66 SW_MUX_CTL_PAD_GPIO_AD_B1_07 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B1_07)

pin 16

SW_MUX_CTL Register

Address: 401F_8000h base + 118h offset = 401F_8118h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B1_07 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_AD_B1_07 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_AD_B1_07. pin 16 000 **ALT0** — Select mux mode: ALT0 mux port: FLEXSPIB_DATA00 of instance: flexspi 001 **ALT1** — Select mux mode: ALT1 mux port: LPI2C3_SCL of instance: lpi2c3 010 **ALT2** — Select mux mode: ALT2 mux port: LPUART3_RX of instance: lpuart3 011 **ALT3** — Select mux mode: ALT3 mux port: SPDIF_EXT_CLK of instance: spdif 100 **ALT4** — Select mux mode: ALT4 mux port: CSI_HSYNC of instance: csi 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO23 of instance: gpio1 110 **ALT6** — Select mux mode: ALT6 mux port: USDHC2_DATA3 of instance: usdhc2 111 **ALT7** — Select mux mode: ALT7 mux port: KPP_COL04 of instance: kpp 1000 **ALT8** — Select mux mode: ALT8 mux port: GPT2_COMPARE3 of instance: gpt2 1001 **ALT9** — Select mux mode: ALT9 mux port: FLEXIO3_FLEXIO07 of instance: flexio3

11.6.67 SW_MUX_CTL_PAD_GPIO_AD_B1_08 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B1_08)

pin 22

SW_MUX_CTL Register

Address: 401F_8000h base + 11Ch offset = 401F_811Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B1_08 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_AD_B1_08 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_AD_B1_08. pin 22 000 **ALT0** — Select mux mode: ALT0 mux port: FLEXSPIA_SS1_B of instance: flexspi 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXPWM4_PWMA00 of instance: flexpwm4 010 **ALT2** — Select mux mode: ALT2 mux port: FLEXCAN1_TX of instance: flexcan1 011 **ALT3** — Select mux mode: ALT3 mux port: CCM_PMIC_READY of instance: ccm 100 **ALT4** — Select mux mode: ALT4 mux port: CSI_DATA09 of instance: csi 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO24 of instance: gpio1 110 **ALT6** — Select mux mode: ALT6 mux port: USDHC2_CMD of instance: usdhc2 111 **ALT7** — Select mux mode: ALT7 mux port: KPP_ROW03 of instance: kpp 1001 **ALT9** — Select mux mode: ALT9 mux port: FLEXIO3_FLEXIO08 of instance: flexio3

11.6.68 SW_MUX_CTL_PAD_GPIO_AD_B1_09 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B1_09)

pin 23

SW_MUX_CTL Register

Address: 401F_8000h base + 120h offset = 401F_8120h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B1_09 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_AD_B1_09 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_AD_B1_09. pin 23 000 **ALT0** — Select mux mode: ALT0 mux port: FLEXSPIA_DQS of instance: flexspi 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXPWM4_PWMA01 of instance: flexpwm4 010 **ALT2** — Select mux mode: ALT2 mux port: FLEXCAN1_RX of instance: flexcan1 011 **ALT3** — Select mux mode: ALT3 mux port: SAI1_MCLK of instance: sai1 100 **ALT4** — Select mux mode: ALT4 mux port: CSI_DATA08 of instance: csi 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO25 of instance: gpio1 110 **ALT6** — Select mux mode: ALT6 mux port: USDHC2_CLK of instance: usdhc2 111 **ALT7** — Select mux mode: ALT7 mux port: KPP_COL03 of instance: kpp 1001 **ALT9** — Select mux mode: ALT9 mux port: FLEXIO3_FLEXIO09 of instance: flexio3

11.6.69 SW_MUX_CTL_PAD_GPIO_AD_B1_10 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B1_10)

pin 20

SW_MUX_CTL Register

Address: 401F_8000h base + 124h offset = 401F_8124h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B1_10 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_AD_B1_10 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_AD_B1_10. pin 20 000 **ALT0** — Select mux mode: ALT0 mux port: FLEXSPIA_DATA03 of instance: flexspi 001 **ALT1** — Select mux mode: ALT1 mux port: WDOG1_B of instance: wdog1 010 **ALT2** — Select mux mode: ALT2 mux port: LPUART8_TX of instance: lpuart8 011 **ALT3** — Select mux mode: ALT3 mux port: SAI1_RX_SYNC of instance: sai1 100 **ALT4** — Select mux mode: ALT4 mux port: CSI_DATA07 of instance: csi 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO26 of instance: gpio1 110 **ALT6** — Select mux mode: ALT6 mux port: USDHC2_WP of instance: usdhc2 111 **ALT7** — Select mux mode: ALT7 mux port: KPP_ROW02 of instance: kpp 1000 **ALT8** — Select mux mode: ALT8 mux port: ENET2_1588_EVENT1_OUT of instance: enet2 1001 **ALT9** — Select mux mode: ALT9 mux port: FLEXIO3_FLEXIO10 of instance: flexio3

11.6.70 SW_MUX_CTL_PAD_GPIO_AD_B1_11 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B1_11)

pin 21

SW_MUX_CTL Register

Address: 401F_8000h base + 128h offset = 401F_8128h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B1_11 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_AD_B1_11 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_AD_B1_11. pin 21 000 **ALT0** — Select mux mode: ALT0 mux port: FLEXSPIA_DATA02 of instance: flexspi 001 **ALT1** — Select mux mode: ALT1 mux port: EWM_OUT_B of instance: ewm 010 **ALT2** — Select mux mode: ALT2 mux port: LPUART8_RX of instance: lpuart8 011 **ALT3** — Select mux mode: ALT3 mux port: SAI1_RX_BCLK of instance: sai1 100 **ALT4** — Select mux mode: ALT4 mux port: CSI_DATA06 of instance: csi 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO27 of instance: gpio1 110 **ALT6** — Select mux mode: ALT6 mux port: USDHC2_RESET_B of instance: usdhc2 111 **ALT7** — Select mux mode: ALT7 mux port: KPP_COL02 of instance: kpp 1000 **ALT8** — Select mux mode: ALT8 mux port: ENET2_1588_EVENT1_IN of instance: enet2 1001 **ALT9** — Select mux mode: ALT9 mux port: FLEXIO3_FLEXIO11 of instance: flexio3

11.6.71 SW_MUX_CTL_PAD_GPIO_AD_B1_12 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B1_12)

pin 38 (T4.1)

SW_MUX_CTL Register

Address: 401F_8000h base + 12Ch offset = 401F_812Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B1_12 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_AD_B1_12 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_AD_B1_12. pin 38 (T4.1) 000 **ALT0** — Select mux mode: ALT0 mux port: FLEXSPIA_DATA01 of instance: flexspi 001 **ALT1** — Select mux mode: ALT1 mux port: ACMP_OUT00 of instance: acmp 010 **ALT2** — Select mux mode: ALT2 mux port: LPSPI3_PCS0 of instance: lpspi3 011 **ALT3** — Select mux mode: ALT3 mux port: SAI1_RX_DATA00 of instance: sai1 100 **ALT4** — Select mux mode: ALT4 mux port: CSI_DATA05 of instance: csi 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO28 of instance: gpio1 110 **ALT6** — Select mux mode: ALT6 mux port: USDHC2_DATA4 of instance: usdhc2 111 **ALT7** — Select mux mode: ALT7 mux port: KPP_ROW01 of instance: kpp 1000 **ALT8** — Select mux mode: ALT8 mux port: ENET2_1588_EVENT2_OUT of instance: enet2 1001 **ALT9** — Select mux mode: ALT9 mux port: FLEXIO3_FLEXIO12 of instance: flexio3

11.6.72 SW_MUX_CTL_PAD_GPIO_AD_B1_13 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B1_13)

pin 39 (T4.1)

SW_MUX_CTL Register

Address: 401F_8000h base + 130h offset = 401F_8130h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B1_13 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_AD_B1_13 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_AD_B1_13. pin 39 (T4.1) 000 **ALT0** — Select mux mode: ALT0 mux port: FLEXSPIA_DATA00 of instance: flexspi 001 **ALT1** — Select mux mode: ALT1 mux port: ACMP_OUT01 of instance: acmp 010 **ALT2** — Select mux mode: ALT2 mux port: LPSPI3_SDI of instance: lpspi3 011 **ALT3** — Select mux mode: ALT3 mux port: SAI1_TX_DATA00 of instance: sai1 100 **ALT4** — Select mux mode: ALT4 mux port: CSI_DATA04 of instance: csi 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO29 of instance: gpio1 110 **ALT6** — Select mux mode: ALT6 mux port: USDHC2_DATA5 of instance: usdhc2 111 **ALT7** — Select mux mode: ALT7 mux port: KPP_COL01 of instance: kpp 1000 **ALT8** — Select mux mode: ALT8 mux port: ENET2_1588_EVENT2_IN of instance: enet2 1001 **ALT9** — Select mux mode: ALT9 mux port: FLEXIO3_FLEXIO13 of instance: flexio3

11.6.73 SW_MUX_CTL_PAD_GPIO_AD_B1_14 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B1_14)

pin 26

SW_MUX_CTL Register

Address: 401F_8000h base + 134h offset = 401F_8134h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B1_14 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_AD_B1_14 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_AD_B1_14. pin 26 000 **ALT0** — Select mux mode: ALT0 mux port: FLEXSPIA_SCLK of instance: flexspi 001 **ALT1** — Select mux mode: ALT1 mux port: ACMP_OUT02 of instance: acmp 010 **ALT2** — Select mux mode: ALT2 mux port: LPSPI3_SDO of instance: lpspi3 011 **ALT3** — Select mux mode: ALT3 mux port: SAI1_TX_BCLK of instance: sai1 100 **ALT4** — Select mux mode: ALT4 mux port: CSI_DATA03 of instance: csi 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO30 of instance: gpio1 110 **ALT6** — Select mux mode: ALT6 mux port: USDHC2_DATA6 of instance: usdhc2 111 **ALT7** — Select mux mode: ALT7 mux port: KPP_ROW00 of instance: kpp 1000 **ALT8** — Select mux mode: ALT8 mux port: ENET2_1588_EVENT3_OUT of instance: enet2 1001 **ALT9** — Select mux mode: ALT9 mux port: FLEXIO3_FLEXIO14 of instance: flexio3

11.6.74 SW_MUX_CTL_PAD_GPIO_AD_B1_15 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B1_15)

pin 27

SW_MUX_CTL Register

Address: 401F_8000h base + 138h offset = 401F_8138h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_B1_15 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_AD_B1_15 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_AD_B1_15. pin 27 000 **ALT0** — Select mux mode: ALT0 mux port: FLEXSPIA_SS0_B of instance: flexspi 001 **ALT1** — Select mux mode: ALT1 mux port: ACMP_OUT03 of instance: acmp 010 **ALT2** — Select mux mode: ALT2 mux port: LPSPI3_SCK of instance: lpspi3 011 **ALT3** — Select mux mode: ALT3 mux port: SAI1_TX_SYNC of instance: sai1 100 **ALT4** — Select mux mode: ALT4 mux port: CSI_DATA02 of instance: csi 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO1_IO31 of instance: gpio1 110 **ALT6** — Select mux mode: ALT6 mux port: USDHC2_DATA7 of instance: usdhc2 111 **ALT7** — Select mux mode: ALT7 mux port: KPP_COL00 of instance: kpp 1000 **ALT8** — Select mux mode: ALT8 mux port: ENET2_1588_EVENT3_IN of instance: enet2 1001 **ALT9** — Select mux mode: ALT9 mux port: FLEXIO3_FLEXIO15 of instance: flexio3

11.6.75 SW_MUX_CTL_PAD_GPIO_B0_00 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B0_00) pin 10

SW_MUX_CTL Register

Address: 401F_8000h base + 13Ch offset = 401F_813Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_B0_00 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_B0_00 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_B0_00. pin 10 000 **ALT0** — Select mux mode: ALT0 mux port: LCD_CLK of instance: lcdif 001 **ALT1** — Select mux mode: ALT1 mux port: QTIMER1_TIMER0 of instance: qtimer1 010 **ALT2** — Select mux mode: ALT2 mux port: MQS_RIGHT of instance: mqs 011 **ALT3** — Select mux mode: ALT3 mux port: LPSPI4_PCS0 of instance: lpspi4 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO2_FLEXIO00 of instance: flexio2 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO00 of instance: gpio2 110 **ALT6** — Select mux mode: ALT6 mux port: SEMC_CSX01 of instance: semc 1000 **ALT8** — Select mux mode: ALT8 mux port: ENET2_MDC of instance: enet2

11.6.76 SW_MUX_CTL_PAD_GPIO_B0_01 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B0_01) pin 12

SW_MUX_CTL Register

Address: 401F_8000h base + 140h offset = 401F_8140h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_B0_01 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_B0_01 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_B0_01. pin 12 000 **ALT0** — Select mux mode: ALT0 mux port: LCD_ENABLE of instance: lcdif 001 **ALT1** — Select mux mode: ALT1 mux port: QTIMER1_TIMER1 of instance: qtimer1 010 **ALT2** — Select mux mode: ALT2 mux port: MQS_LEFT of instance: mqs 011 **ALT3** — Select mux mode: ALT3 mux port: LPSPI4_SDI of instance: lpspi4 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO2_FLEXIO01 of instance: flexio2 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO01 of instance: gpio2 110 **ALT6** — Select mux mode: ALT6 mux port: SEMC_CSX02 of instance: semc 1000 **ALT8** — Select mux mode: ALT8 mux port: ENET2_MDIO of instance: enet2

11.6.77 SW_MUX_CTL_PAD_GPIO_B0_02 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B0_02) pin 11

SW_MUX_CTL Register

Address: 401F_8000h base + 144h offset = 401F_8144h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_B0_02 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_B0_02 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_B0_02. pin 11 000 **ALT0** — Select mux mode: ALT0 mux port: LCD_HSYNC of instance: lcdif 001 **ALT1** — Select mux mode: ALT1 mux port: QTIMER1_TIMER2 of instance: qtimer1 010 **ALT2** — Select mux mode: ALT2 mux port: FLEXCAN1_TX of instance: flexcan1 011 **ALT3** — Select mux mode: ALT3 mux port: LPSPI4_SDO of instance: lpspi4 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO2_FLEXIO02 of instance: flexio2 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO02 of instance: gpio2 110 **ALT6** — Select mux mode: ALT6 mux port: SEMC_CSX03 of instance: semc 1000 **ALT8** — Select mux mode: ALT8 mux port: ENET2_1588_EVENT0_OUT of instance: enet2

11.6.78 SW_MUX_CTL_PAD_GPIO_B0_03 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B0_03) pin 13

SW_MUX_CTL Register

Address: 401F_8000h base + 148h offset = 401F_8148h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_B0_03 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_B0_03 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_B0_03. pin 13 000 **ALT0** — Select mux mode: ALT0 mux port: LCD_VSYNC of instance: lcdif 001 **ALT1** — Select mux mode: ALT1 mux port: QTIMER2_TIMER0 of instance: qtimer2 010 **ALT2** — Select mux mode: ALT2 mux port: FLEXCAN1_RX of instance: flexcan1 011 **ALT3** — Select mux mode: ALT3 mux port: LPSPI4_SCK of instance: lpspi4 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO2_FLEXIO03 of instance: flexio2 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO03 of instance: gpio2 110 **ALT6** — Select mux mode: ALT6 mux port: WDOG2_RESET_B_DEB of instance: wdog2 1000 **ALT8** — Select mux mode: ALT8 mux port: ENET2_1588_EVENT0_IN of instance: enet2

11.6.79 SW_MUX_CTL_PAD_GPIO_B0_04 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B0_04)

pin 40 (MM)

SW_MUX_CTL Register

Address: 401F_8000h base + 14Ch offset = 401F_814Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_B0_04 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_B0_04 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_B0_04. pin 40 (MM) 000 **ALT0** — Select mux mode: ALT0 mux port: LCD_DATA00 of instance: lcdif 001 **ALT1** — Select mux mode: ALT1 mux port: QTIMER2_TIMER1 of instance: qtimer2 010 **ALT2** — Select mux mode: ALT2 mux port: LPI2C2_SCL of instance: lpi2c2 011 **ALT3** — Select mux mode: ALT3 mux port: ARM_TRACE0 of instance: cm7_mx6rt 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO2_FLEXIO04 of instance: flexio2 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO04 of instance: gpio2 110 **ALT6** — Select mux mode: ALT6 mux port: SRC_BOOT_CFG00 of instance: src 1000 **ALT8** — Select mux mode: ALT8 mux port: ENET2_TDATA03 of instance: enet2

11.6.80 SW_MUX_CTL_PAD_GPIO_B0_05 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B0_05)

pin 41 (MM)

SW_MUX_CTL Register

Address: 401F_8000h base + 150h offset = 401F_8150h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_B0_05 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_B0_05 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_B0_05. pin 41 (MM) 000 **ALT0** — Select mux mode: ALT0 mux port: LCD_DATA01 of instance: lcdif 001 **ALT1** — Select mux mode: ALT1 mux port: QTIMER2_TIMER2 of instance: qtimer2 010 **ALT2** — Select mux mode: ALT2 mux port: LPI2C2_SDA of instance: lpi2c2 011 **ALT3** — Select mux mode: ALT3 mux port: ARM_TRACE1 of instance: cm7_mx6rt 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO2_FLEXIO05 of instance: flexio2 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO05 of instance: gpio2 110 **ALT6** — Select mux mode: ALT6 mux port: SRC_BOOT_CFG01 of instance: src 1000 **ALT8** — Select mux mode: ALT8 mux port: ENET2_TDATA02 of instance: enet2

11.6.81 SW_MUX_CTL_PAD_GPIO_B0_06 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B0_06)

pin 42 (MM)

SW_MUX_CTL Register

Address: 401F_8000h base + 154h offset = 401F_8154h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_B0_06 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_B0_06 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_B0_06. pin 42 (MM) 000 **ALT0** — Select mux mode: ALT0 mux port: LCD_DATA02 of instance: lcdif 001 **ALT1** — Select mux mode: ALT1 mux port: QTIMER3_TIMER0 of instance: qtimer3 010 **ALT2** — Select mux mode: ALT2 mux port: FLEXPWM2_PWMA00 of instance: flexpwm2 011 **ALT3** — Select mux mode: ALT3 mux port: ARM_TRACE2 of instance: cm7_mx6rt 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO2_FLEXIO06 of instance: flexio2 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO06 of instance: gpio2 110 **ALT6** — Select mux mode: ALT6 mux port: SRC_BOOT_CFG02 of instance: src 1000 **ALT8** — Select mux mode: ALT8 mux port: ENET2_RX_CLK of instance: enet2

11.6.82 SW_MUX_CTL_PAD_GPIO_B0_07 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B0_07)

SW_MUX_CTL Register

pin 43 (MM)

Address: 401F_8000h base + 158h offset = 401F_8158h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_B0_07 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_B0_07 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_B0_07. pin 43 (MM) 000 **ALT0** — Select mux mode: ALT0 mux port: LCD_DATA03 of instance: lcdif 001 **ALT1** — Select mux mode: ALT1 mux port: QTIMER3_TIMER1 of instance: qtimer3 010 **ALT2** — Select mux mode: ALT2 mux port: FLEXPWM2_PWMB00 of instance: flexpwm2 011 **ALT3** — Select mux mode: ALT3 mux port: ARM_TRACE3 of instance: cm7_mx6rt 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO2_FLEXIO07 of instance: flexio2 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO07 of instance: gpio2 110 **ALT6** — Select mux mode: ALT6 mux port: SRC_BOOT_CFG03 of instance: src 1000 **ALT8** — Select mux mode: ALT8 mux port: ENET2_TX_ER of instance: enet2

11.6.83 SW_MUX_CTL_PAD_GPIO_B0_08 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B0_08)

SW_MUX_CTL Register

pin 44 (MM)

Address: 401F_8000h base + 15Ch offset = 401F_815Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_B0_08 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_B0_08 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_B0_08. pin 44 (MM) 000 **ALT0** — Select mux mode: ALT0 mux port: LCD_DATA04 of instance: lcdif 001 **ALT1** — Select mux mode: ALT1 mux port: QTIMER3_TIMER2 of instance: qtimer3 010 **ALT2** — Select mux mode: ALT2 mux port: FLEXPWM2_PWMA01 of instance: flexpwm2 011 **ALT3** — Select mux mode: ALT3 mux port: LPUART3_TX of instance: lpuart3 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO2_FLEXIO08 of instance: flexio2 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO08 of instance: gpio2 110 **ALT6** — Select mux mode: ALT6 mux port: SRC_BOOT_CFG04 of instance: src 1000 **ALT8** — Select mux mode: ALT8 mux port: ENET2_RDATA03 of instance: enet2

11.6.84 SW_MUX_CTL_PAD_GPIO_B0_09 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B0_09)

SW_MUX_CTL Register

pin 45 (MM)

Address: 401F_8000h base + 160h offset = 401F_8160h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_B0_09 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_B0_09 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_B0_09. pin 45 (MM) 000 **ALT0** — Select mux mode: ALT0 mux port: LCD_DATA05 of instance: lcdif 001 **ALT1** — Select mux mode: ALT1 mux port: QTIMER4_TIMER0 of instance: qtimer4 010 **ALT2** — Select mux mode: ALT2 mux port: FLEXPWM2_PWMB01 of instance: flexpwm2 011 **ALT3** — Select mux mode: ALT3 mux port: LPUART3_RX of instance: lpuart3 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO2_FLEXIO09 of instance: flexio2 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO09 of instance: gpio2 110 **ALT6** — Select mux mode: ALT6 mux port: SRC_BOOT_CFG05 of instance: src 1000 **ALT8** — Select mux mode: ALT8 mux port: ENET2_RDATA02 of instance: enet2

11.6.85 SW_MUX_CTL_PAD_GPIO_B0_10 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B0_10) pin 6

SW_MUX_CTL Register

Address: 401F_8000h base + 164h offset = 401F_8164h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_B0_10 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_B0_10 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_B0_10. pin 6 000 **ALT0** — Select mux mode: ALT0 mux port: LCD_DATA06 of instance: lcdif 001 **ALT1** — Select mux mode: ALT1 mux port: QTIMER4_TIMER1 of instance: qtimer4 010 **ALT2** — Select mux mode: ALT2 mux port: FLEXPWM2_PWMA02 of instance: flexpwm2 011 **ALT3** — Select mux mode: ALT3 mux port: SAI1_TX_DATA03 of instance: sai1 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO2_FLEXIO10 of instance: flexio2 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO10 of instance: gpio2 110 **ALT6** — Select mux mode: ALT6 mux port: SRC_BOOT_CFG06 of instance: src 1000 **ALT8** — Select mux mode: ALT8 mux port: ENET2_CRS of instance: enet2

11.6.86 SW_MUX_CTL_PAD_GPIO_B0_11 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B0_11) pin 9

SW_MUX_CTL Register

Address: 401F_8000h base + 168h offset = 401F_8168h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_B0_11 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_B0_11 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_B0_11. pin 9 000 **ALT0** — Select mux mode: ALT0 mux port: LCD_DATA07 of instance: lcdif 001 **ALT1** — Select mux mode: ALT1 mux port: QTIMER4_TIMER2 of instance: qtimer4 010 **ALT2** — Select mux mode: ALT2 mux port: FLEXPWM2_PWMB02 of instance: flexpwm2 011 **ALT3** — Select mux mode: ALT3 mux port: SAI1_TX_DATA02 of instance: sai1 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO2_FLEXIO11 of instance: flexio2 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO11 of instance: gpio2 110 **ALT6** — Select mux mode: ALT6 mux port: SRC_BOOT_CFG07 of instance: src 1000 **ALT8** — Select mux mode: ALT8 mux port: ENET2_COL of instance: enet2

11.6.87 SW_MUX_CTL_PAD_GPIO_B0_12 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B0_12) pin 32

SW_MUX_CTL Register

Address: 401F_8000h base + 16Ch offset = 401F_816Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_B0_12 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_B0_12 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_B0_12. pin 32 000 **ALT0** — Select mux mode: ALT0 mux port: LCD_DATA08 of instance: lcdif 001 **ALT1** — Select mux mode: ALT1 mux port: XBAR1_INOUT10 of instance: xbar1 010 **ALT2** — Select mux mode: ALT2 mux port: ARM_TRACE_CLK of instance: cm7_mx6rt 011 **ALT3** — Select mux mode: ALT3 mux port: SAI1_TX_DATA01 of instance: sai1 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO2_FLEXIO12 of instance: flexio2 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO12 of instance: gpio2 110 **ALT6** — Select mux mode: ALT6 mux port: SRC_BOOT_CFG08 of instance: src 1000 **ALT8** — Select mux mode: ALT8 mux port: ENET2_TDATA00 of instance: enet2

11.6.88 SW_MUX_CTL_PAD_GPIO_B0_13 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B0_13)

SW_MUX_CTL Register

Address: 401F_8000h base + 170h offset = 401F_8170h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_B0_13 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_B0_13 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_B0_13. 000 **ALT0** — Select mux mode: ALT0 mux port: LCD_DATA09 of instance: lcdif 001 **ALT1** — Select mux mode: ALT1 mux port: XBAR1_INOUT11 of instance: xbar1 010 **ALT2** — Select mux mode: ALT2 mux port: ARM_TRACE_SWO of instance: cm7_mx6rt 011 **ALT3** — Select mux mode: ALT3 mux port: SAI1_MCLK of instance: sai1 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO2_FLEXIO13 of instance: flexio2 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO13 of instance: gpio2 110 **ALT6** — Select mux mode: ALT6 mux port: SRC_BOOT_CFG09 of instance: src 1000 **ALT8** — Select mux mode: ALT8 mux port: ENET2_TDATA01 of instance: enet2

11.6.89 SW_MUX_CTL_PAD_GPIO_B0_14 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B0_14)

SW_MUX_CTL Register

Address: 401F_8000h base + 174h offset = 401F_8174h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_B0_14 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_B0_14 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_B0_14. 000 **ALT0** — Select mux mode: ALT0 mux port: LCD_DATA10 of instance: lcdif 001 **ALT1** — Select mux mode: ALT1 mux port: XBAR1_INOUT12 of instance: xbar1 010 **ALT2** — Select mux mode: ALT2 mux port: ARM_TXEV of instance: cm7_mx6rt 011 **ALT3** — Select mux mode: ALT3 mux port: SAI1_RX_SYNC of instance: sai1 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO2_FLEXIO14 of instance: flexio2 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO14 of instance: gpio2 110 **ALT6** — Select mux mode: ALT6 mux port: SRC_BOOT_CFG10 of instance: src 1000 **ALT8** — Select mux mode: ALT8 mux port: ENET2_TX_EN of instance: enet2

11.6.90 SW_MUX_CTL_PAD_GPIO_B0_15 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B0_15)

SW_MUX_CTL Register

Address: 401F_8000h base + 178h offset = 401F_8178h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_B0_15 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_B0_15 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_B0_15. 000 **ALT0** — Select mux mode: ALT0 mux port: LCD_DATA11 of instance: lcdif 001 **ALT1** — Select mux mode: ALT1 mux port: XBAR1_INOUT13 of instance: xbar1 010 **ALT2** — Select mux mode: ALT2 mux port: ARM_RXEV of instance: cm7_mx6rt 011 **ALT3** — Select mux mode: ALT3 mux port: SAI1_RX_BCLK of instance: sai1 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO2_FLEXIO15 of instance: flexio2 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO15 of instance: gpio2 110 **ALT6** — Select mux mode: ALT6 mux port: SRC_BOOT_CFG11 of instance: src 1000 **ALT8** — Select mux mode: ALT8 mux port: ENET2_TX_CLK of instance: enet2 1001 **ALT9** — Select mux mode: ALT9 mux port: ENET2_REF_CLK2 of instance: enet2

11.6.91 SW_MUX_CTL_PAD_GPIO_B1_00 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B1_00) pin 8

SW_MUX_CTL Register

Address: 401F_8000h base + 17Ch offset = 401F_817Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_B1_00 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_B1_00 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_B1_00. pin 8 000 **ALT0** — Select mux mode: ALT0 mux port: LCD_DATA12 of instance: lcdif 001 **ALT1** — Select mux mode: ALT1 mux port: XBAR1_INOUT14 of instance: xbar1 010 **ALT2** — Select mux mode: ALT2 mux port: LPUART4_TX of instance: lpuart4 011 **ALT3** — Select mux mode: ALT3 mux port: SAI1_RX_DATA00 of instance: sai1 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO2_FLEXIO16 of instance: flexio2 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO16 of instance: gpio2 110 **ALT6** — Select mux mode: ALT6 mux port: FLEXPWM1_PWMA03 of instance: flexpwm1 1000 **ALT8** — Select mux mode: ALT8 mux port: ENET2_RX_ER of instance: enet2 1001 **ALT9** — Select mux mode: ALT9 mux port: FLEXIO3_FLEXIO16 of instance: flexio3

11.6.92 SW_MUX_CTL_PAD_GPIO_B1_01 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B1_01) pin 7

SW_MUX_CTL Register

Address: 401F_8000h base + 180h offset = 401F_8180h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_B1_01 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_B1_01 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_B1_01. pin 7 000 **ALT0** — Select mux mode: ALT0 mux port: LCD_DATA13 of instance: lcdif 001 **ALT1** — Select mux mode: ALT1 mux port: XBAR1_INOUT15 of instance: xbar1 010 **ALT2** — Select mux mode: ALT2 mux port: LPUART4_RX of instance: lpuart4 011 **ALT3** — Select mux mode: ALT3 mux port: SAI1_TX_DATA00 of instance: sai1 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO2_FLEXIO17 of instance: flexio2 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO17 of instance: gpio2 110 **ALT6** — Select mux mode: ALT6 mux port: FLEXPWM1_PWMB03 of instance: flexpwm1 1000 **ALT8** — Select mux mode: ALT8 mux port: ENET2_RDATA00 of instance: enet2 1001 **ALT9** — Select mux mode: ALT9 mux port: FLEXIO3_FLEXIO17 of instance: flexio3

11.6.93 SW_MUX_CTL_PAD_GPIO_B1_02 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B1_02)

pin 36 (T4.1)

SW_MUX_CTL Register

Address: 401F_8000h base + 184h offset = 401F_8184h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_B1_02 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_B1_02 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_B1_02. pin 36 (T4.1) 000 **ALT0** — Select mux mode: ALT0 mux port: LCD_DATA14 of instance: lcdif 001 **ALT1** — Select mux mode: ALT1 mux port: XBAR1_INOUT16 of instance: xbar1 010 **ALT2** — Select mux mode: ALT2 mux port: LPSPI4_PCS2 of instance: lpspi4 011 **ALT3** — Select mux mode: ALT3 mux port: SAI1_TX_BCLK of instance: sai1 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO2_FLEXIO18 of instance: flexio2 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO18 of instance: gpio2 110 **ALT6** — Select mux mode: ALT6 mux port: FLEXPWM2_PWMA03 of instance: flexpwm2 1000 **ALT8** — Select mux mode: ALT8 mux port: ENET2_RDATA01 of instance: enet2 1001 **ALT9** — Select mux mode: ALT9 mux port: FLEXIO3_FLEXIO18 of instance: flexio3

11.6.94 SW_MUX_CTL_PAD_GPIO_B1_03 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B1_03)

pin 37 (T4.1)

SW_MUX_CTL Register

Address: 401F_8000h base + 188h offset = 401F_8188h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_B1_03 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_B1_03 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_B1_03. pin 37 (T4.1) 000 **ALT0** — Select mux mode: ALT0 mux port: LCD_DATA15 of instance: lcdif 001 **ALT1** — Select mux mode: ALT1 mux port: XBAR1_INOUT17 of instance: xbar1 010 **ALT2** — Select mux mode: ALT2 mux port: LPSPI4_PCS1 of instance: lpspi4 011 **ALT3** — Select mux mode: ALT3 mux port: SAI1_TX_SYNC of instance: sai1 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO2_FLEXIO19 of instance: flexio2 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO19 of instance: gpio2 110 **ALT6** — Select mux mode: ALT6 mux port: FLEXPWM2_PWMB03 of instance: flexpwm2 1000 **ALT8** — Select mux mode: ALT8 mux port: ENET2_RX_EN of instance: enet2 1001 **ALT9** — Select mux mode: ALT9 mux port: FLEXIO3_FLEXIO19 of instance: flexio3

11.6.95 SW_MUX_CTL_PAD_GPIO_B1_04 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B1_04)

SW_MUX_CTL Register

Address: 401F_8000h base + 18Ch offset = 401F_818Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_B1_04 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_B1_04 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_B1_04. 000 **ALT0** — Select mux mode: ALT0 mux port: LCD_DATA16 of instance: lcdif 001 **ALT1** — Select mux mode: ALT1 mux port: LPSPI4_PCS0 of instance: lpspi4 010 **ALT2** — Select mux mode: ALT2 mux port: CSI_DATA15 of instance: csi 011 **ALT3** — Select mux mode: ALT3 mux port: ENET_RX_DATA00 of instance: enet 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO2_FLEXIO20 of instance: flexio2 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO20 of instance: gpio2 1000 **ALT8** — Select mux mode: ALT8 mux port: GPT1_CLK of instance: gpt1 1001 **ALT9** — Select mux mode: ALT9 mux port: FLEXIO3_FLEXIO20 of instance: flexio3

11.6.96 SW_MUX_CTL_PAD_GPIO_B1_05 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B1_05)

SW_MUX_CTL Register

Address: 401F_8000h base + 190h offset = 401F_8190h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_B1_05 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_B1_05 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_B1_05. 000 **ALT0** — Select mux mode: ALT0 mux port: LCD_DATA17 of instance: lcdif 001 **ALT1** — Select mux mode: ALT1 mux port: LPSPI4_SDI of instance: lpspi4 010 **ALT2** — Select mux mode: ALT2 mux port: CSI_DATA14 of instance: csi 011 **ALT3** — Select mux mode: ALT3 mux port: ENET_RX_DATA01 of instance: enet 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO2_FLEXIO21 of instance: flexio2 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO21 of instance: gpio2 1000 **ALT8** — Select mux mode: ALT8 mux port: GPT1_CAPTURE1 of instance: gpt1 1001 **ALT9** — Select mux mode: ALT9 mux port: FLEXIO3_FLEXIO21 of instance: flexio3

11.6.97 SW_MUX_CTL_PAD_GPIO_B1_06 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B1_06)

SW_MUX_CTL Register

Address: 401F_8000h base + 194h offset = 401F_8194h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_B1_06 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_B1_06 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_B1_06. 000 **ALT0** — Select mux mode: ALT0 mux port: LCD_DATA18 of instance: lcdif 001 **ALT1** — Select mux mode: ALT1 mux port: LPSPI4_SDO of instance: lpspi4 010 **ALT2** — Select mux mode: ALT2 mux port: CSI_DATA13 of instance: csi 011 **ALT3** — Select mux mode: ALT3 mux port: ENET_RX_EN of instance: enet 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO2_FLEXIO22 of instance: flexio2 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO22 of instance: gpio2 1000 **ALT8** — Select mux mode: ALT8 mux port: GPT1_CAPTURE2 of instance: gpt1 1001 **ALT9** — Select mux mode: ALT9 mux port: FLEXIO3_FLEXIO22 of instance: flexio3

11.6.98 SW_MUX_CTL_PAD_GPIO_B1_07 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B1_07)

SW_MUX_CTL Register

Address: 401F_8000h base + 198h offset = 401F_8198h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_B1_07 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_B1_07 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_B1_07. 000 **ALT0** — Select mux mode: ALT0 mux port: LCD_DATA19 of instance: lcdif 001 **ALT1** — Select mux mode: ALT1 mux port: LPSPI4_SCK of instance: lpspi4 010 **ALT2** — Select mux mode: ALT2 mux port: CSI_DATA12 of instance: csi 011 **ALT3** — Select mux mode: ALT3 mux port: ENET_TX_DATA00 of instance: enet 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO2_FLEXIO23 of instance: flexio2 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO23 of instance: gpio2 1000 **ALT8** — Select mux mode: ALT8 mux port: GPT1_COMPARE1 of instance: gpt1 1001 **ALT9** — Select mux mode: ALT9 mux port: FLEXIO3_FLEXIO23 of instance: flexio3

11.6.99 SW_MUX_CTL_PAD_GPIO_B1_08 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B1_08)

SW_MUX_CTL Register

Address: 401F_8000h base + 19Ch offset = 401F_819Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_B1_08 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_B1_08 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_B1_08. 000 **ALT0** — Select mux mode: ALT0 mux port: LCD_DATA20 of instance: lcdif 001 **ALT1** — Select mux mode: ALT1 mux port: QTIMER1_TIMER3 of instance: qtimer1 010 **ALT2** — Select mux mode: ALT2 mux port: CSI_DATA11 of instance: csi 011 **ALT3** — Select mux mode: ALT3 mux port: ENET_TX_DATA01 of instance: enet 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO2_FLEXIO24 of instance: flexio2 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO24 of instance: gpio2 110 **ALT6** — Select mux mode: ALT6 mux port: FLEXCAN2_TX of instance: flexcan2 1000 **ALT8** — Select mux mode: ALT8 mux port: GPT1_COMPARE2 of instance: gpt1 1001 **ALT9** — Select mux mode: ALT9 mux port: FLEXIO3_FLEXIO24 of instance: flexio3

11.6.100 SW_MUX_CTL_PAD_GPIO_B1_09 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B1_09)

SW_MUX_CTL Register

Address: 401F_8000h base + 1A0h offset = 401F_81A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_B1_09 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_B1_09 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_B1_09. 000 **ALT0** — Select mux mode: ALT0 mux port: LCD_DATA21 of instance: lcdif 001 **ALT1** — Select mux mode: ALT1 mux port: QTIMER2_TIMER3 of instance: qtimer2 010 **ALT2** — Select mux mode: ALT2 mux port: CSI_DATA10 of instance: csi 011 **ALT3** — Select mux mode: ALT3 mux port: ENET_TX_EN of instance: enet 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO2_FLEXIO25 of instance: flexio2 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO25 of instance: gpio2 110 **ALT6** — Select mux mode: ALT6 mux port: FLEXCAN2_RX of instance: flexcan2 1000 **ALT8** — Select mux mode: ALT8 mux port: GPT1_COMPARE3 of instance: gpt1 1001 **ALT9** — Select mux mode: ALT9 mux port: FLEXIO3_FLEXIO25 of instance: flexio3

11.6.101 SW_MUX_CTL_PAD_GPIO_B1_10 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B1_10)

SW_MUX_CTL Register

Address: 401F_8000h base + 1A4h offset = 401F_81A4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_B1_10 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_B1_10 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_B1_10. 000 **ALT0** — Select mux mode: ALT0 mux port: LCD_DATA22 of instance: lcdif 001 **ALT1** — Select mux mode: ALT1 mux port: QTIMER3_TIMER3 of instance: qtimer3 010 **ALT2** — Select mux mode: ALT2 mux port: CSI_DATA00 of instance: csi 011 **ALT3** — Select mux mode: ALT3 mux port: ENET_TX_CLK of instance: enet 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO2_FLEXIO26 of instance: flexio2 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO26 of instance: gpio2 110 **ALT6** — Select mux mode: ALT6 mux port: ENET_REF_CLK of instance: enet 1001 **ALT9** — Select mux mode: ALT9 mux port: FLEXIO3_FLEXIO26 of instance: flexio3

11.6.102 SW_MUX_CTL_PAD_GPIO_B1_11 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B1_11)

SW_MUX_CTL Register

Address: 401F_8000h base + 1A8h offset = 401F_81A8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_B1_11 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_B1_11 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_B1_11. 000 **ALT0** — Select mux mode: ALT0 mux port: LCD_DATA23 of instance: lcdif 001 **ALT1** — Select mux mode: ALT1 mux port: QTIMER4_TIMER3 of instance: qtimer4 010 **ALT2** — Select mux mode: ALT2 mux port: CSI_DATA01 of instance: csi 011 **ALT3** — Select mux mode: ALT3 mux port: ENET_RX_ER of instance: enet 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO2_FLEXIO27 of instance: flexio2 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO27 of instance: gpio2 110 **ALT6** — Select mux mode: ALT6 mux port: LPSPI4_PCS3 of instance: lpspi4 1001 **ALT9** — Select mux mode: ALT9 mux port: FLEXIO3_FLEXIO27 of instance: flexio3

11.6.103 SW_MUX_CTL_PAD_GPIO_B1_12 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B1_12)

pin 35 (T4.1)

SW_MUX_CTL Register

Address: 401F_8000h base + 1ACh offset = 401F_81ACh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_B1_12 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_B1_12 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_B1_12. pin 35 (T4.1) 001 **ALT1** — Select mux mode: ALT1 mux port: LPUART5_TX of instance: lpuart5 010 **ALT2** — Select mux mode: ALT2 mux port: CSI_PIXCLK of instance: csi 011 **ALT3** — Select mux mode: ALT3 mux port: ENET_1588_EVENT0_IN of instance: enet 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO2_FLEXIO28 of instance: flexio2 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO28 of instance: gpio2 110 **ALT6** — Select mux mode: ALT6 mux port: USDHC1_CD_B of instance: usdhc1 1001 **ALT9** — Select mux mode: ALT9 mux port: FLEXIO3_FLEXIO28 of instance: flexio3

11.6.104 SW_MUX_CTL_PAD_GPIO_B1_13 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B1_13)

pin 34 (T4.1)

SW_MUX_CTL Register

Address: 401F_8000h base + 1B0h offset = 401F_81B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_B1_13 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_B1_13 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_B1_13. pin 34 (T4.1) 000 **ALT0** — Select mux mode: ALT0 mux port: WDOG1_B of instance: wdog1 001 **ALT1** — Select mux mode: ALT1 mux port: LPUART5_RX of instance: lpuart5 010 **ALT2** — Select mux mode: ALT2 mux port: CSI_VSYNC of instance: csi 011 **ALT3** — Select mux mode: ALT3 mux port: ENET_1588_EVENT0_OUT of instance: enet 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO2_FLEXIO29 of instance: flexio2 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO29 of instance: gpio2 110 **ALT6** — Select mux mode: ALT6 mux port: USDHC1_WP of instance: usdhc1 1000 **ALT8** — Select mux mode: ALT8 mux port: SEMC_DQS4 of instance: semc 1001 **ALT9** — Select mux mode: ALT9 mux port: FLEXIO3_FLEXIO29 of instance: flexio3

11.6.105 SW_MUX_CTL_PAD_GPIO_B1_14 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B1_14)

SW_MUX_CTL Register

Address: 401F_8000h base + 1B4h offset = 401F_81B4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_B1_14 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_B1_14 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_B1_14. 000 **ALT0** — Select mux mode: ALT0 mux port: ENET_MDC of instance: enet 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXPWM4_PWMA02 of instance: flexpwm4 010 **ALT2** — Select mux mode: ALT2 mux port: CSI_HSYNC of instance: csi 011 **ALT3** — Select mux mode: ALT3 mux port: XBAR1_IN02 of instance: xbar1 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO2_FLEXIO30 of instance: flexio2 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO30 of instance: gpio2 110 **ALT6** — Select mux mode: ALT6 mux port: USDHC1_VSELECT of instance: usdhc1 1000 **ALT8** — Select mux mode: ALT8 mux port: ENET2_TDATA00 of instance: enet2 1001 **ALT9** — Select mux mode: ALT9 mux port: FLEXIO3_FLEXIO30 of instance: flexio3

11.6.106 SW_MUX_CTL_PAD_GPIO_B1_15 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_B1_15)

SW_MUX_CTL Register

Address: 401F_8000h base + 1B8h offset = 401F_81B8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_B1_15 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_B1_15 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_B1_15. 000 **ALT0** — Select mux mode: ALT0 mux port: ENET_MDIO of instance: enet 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXPWM4_PWMA03 of instance: flexpwm4 010 **ALT2** — Select mux mode: ALT2 mux port: CSI_MCLK of instance: csi 011 **ALT3** — Select mux mode: ALT3 mux port: XBAR1_IN03 of instance: xbar1 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO2_FLEXIO31 of instance: flexio2 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO31 of instance: gpio2 110 **ALT6** — Select mux mode: ALT6 mux port: USDHC1_RESET_B of instance: usdhc1 1000 **ALT8** — Select mux mode: ALT8 mux port: ENET2_TDATA01 of instance: enet2 1001 **ALT9** — Select mux mode: ALT9 mux port: FLEXIO3_FLEXIO31 of instance: flexio3

11.6.107 SW_MUX_CTL_PAD_GPIO_SD_B0_00 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B0_00)

SW_MUX_CTL Register

Address: 401F_8000h base + 1BCh offset = 401F_81BCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B0_00 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_SD_B0_00 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_SD_B0_00. 000 **ALT0** — Select mux mode: ALT0 mux port: USDHC1_CMD of instance: usdhc1 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXPWM1_PWMA00 of instance: flexpwm1 010 **ALT2** — Select mux mode: ALT2 mux port: LPI2C3_SCL of instance: lpi2c3 011 **ALT3** — Select mux mode: ALT3 mux port: XBAR1_INOUT04 of instance: xbar1 100 **ALT4** — Select mux mode: ALT4 mux port: LPSPI1_SCK of instance: lpspi1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO12 of instance: gpio3 110 **ALT6** — Select mux mode: ALT6 mux port: FLEXSPIA_SS1_B of instance: flexspi 1000 **ALT8** — Select mux mode: ALT8 mux port: ENET2_TX_EN of instance: enet2 1001 **ALT9** — Select mux mode: ALT9 mux port: SEMC_DQS4 of instance: semc

11.6.108 SW_MUX_CTL_PAD_GPIO_SD_B0_01 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B0_01)

SW_MUX_CTL Register

Address: 401F_8000h base + 1C0h offset = 401F_81C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B0_01 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_SD_B0_01 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_SD_B0_01. 000 **ALT0** — Select mux mode: ALT0 mux port: USDHC1_CLK of instance: usdhc1 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXPWM1_PWMB00 of instance: flexpwm1 010 **ALT2** — Select mux mode: ALT2 mux port: LPI2C3_SDA of instance: lpi2c3 011 **ALT3** — Select mux mode: ALT3 mux port: XBAR1_INOUT05 of instance: xbar1 100 **ALT4** — Select mux mode: ALT4 mux port: LPSPI1_PCS0 of instance: lpspi1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO13 of instance: gpio3 110 **ALT6** — Select mux mode: ALT6 mux port: FLEXSPIB_SS1_B of instance: flexspi 1000 **ALT8** — Select mux mode: ALT8 mux port: ENET2_TX_CLK of instance: enet2 1001 **ALT9** — Select mux mode: ALT9 mux port: ENET2_REF_CLK2 of instance: enet2

11.6.109 SW_MUX_CTL_PAD_GPIO_SD_B0_02 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B0_02)

SW_MUX_CTL Register

Address: 401F_8000h base + 1C4h offset = 401F_81C4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B0_02 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_SD_B0_02 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_SD_B0_02. 000 **ALT0** — Select mux mode: ALT0 mux port: USDHC1_DATA0 of instance: usdhc1 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXPWM1_PWMA01 of instance: flexpwm1 010 **ALT2** — Select mux mode: ALT2 mux port: LPUART8_CTS_B of instance: lpuart8 011 **ALT3** — Select mux mode: ALT3 mux port: XBAR1_INOUT06 of instance: xbar1 100 **ALT4** — Select mux mode: ALT4 mux port: LPSPI1_SDO of instance: lpspi1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO14 of instance: gpio3 1000 **ALT8** — Select mux mode: ALT8 mux port: ENET2_RX_ER of instance: enet2 1001 **ALT9** — Select mux mode: ALT9 mux port: SEMC_CLK5 of instance: semc

11.6.110 SW_MUX_CTL_PAD_GPIO_SD_B0_03 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B0_03)

SW_MUX_CTL Register

Address: 401F_8000h base + 1C8h offset = 401F_81C8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B0_03 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_SD_B0_03 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_SD_B0_03. 000 **ALT0** — Select mux mode: ALT0 mux port: USDHC1_DATA1 of instance: usdhc1 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXPWM1_PWMB01 of instance: flexpwm1 010 **ALT2** — Select mux mode: ALT2 mux port: LPUART8_RTS_B of instance: lpuart8 011 **ALT3** — Select mux mode: ALT3 mux port: XBAR1_INOUT07 of instance: xbar1 100 **ALT4** — Select mux mode: ALT4 mux port: LPSPI1_SDI of instance: lpspi1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO15 of instance: gpio3 1000 **ALT8** — Select mux mode: ALT8 mux port: ENET2_RDATA00 of instance: enet2 1001 **ALT9** — Select mux mode: ALT9 mux port: SEMC_CLK6 of instance: semc

11.6.111 SW_MUX_CTL_PAD_GPIO_SD_B0_04 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B0_04)

SW_MUX_CTL Register

Address: 401F_8000h base + 1CCh offset = 401F_81CCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B0_04 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_SD_B0_04 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_SD_B0_04. 000 **ALT0** — Select mux mode: ALT0 mux port: USDHC1_DATA2 of instance: usdhc1 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXPWM1_PWMA02 of instance: flexpwm1 010 **ALT2** — Select mux mode: ALT2 mux port: LPUART8_TX of instance: lpuart8 011 **ALT3** — Select mux mode: ALT3 mux port: XBAR1_INOUT08 of instance: xbar1 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXSPIB_SS0_B of instance: flexspi 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO16 of instance: gpio3 110 **ALT6** — Select mux mode: ALT6 mux port: CCM_CLKO1 of instance: ccm 1000 **ALT8** — Select mux mode: ALT8 mux port: ENET2_RDATA01 of instance: enet2

11.6.112 SW_MUX_CTL_PAD_GPIO_SD_B0_05 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B0_05)

SW_MUX_CTL Register

Address: 401F_8000h base + 1D0h offset = 401F_81D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B0_05 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_SD_B0_05 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_SD_B0_05. 000 **ALT0** — Select mux mode: ALT0 mux port: USDHC1_DATA3 of instance: usdhc1 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXPWM1_PWMB02 of instance: flexpwm1 010 **ALT2** — Select mux mode: ALT2 mux port: LPUART8_RX of instance: lpuart8 011 **ALT3** — Select mux mode: ALT3 mux port: XBAR1_INOUT09 of instance: xbar1 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXSPIB_DQS of instance: flexspi 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO17 of instance: gpio3 110 **ALT6** — Select mux mode: ALT6 mux port: CCM_CLKO2 of instance: ccm 1000 **ALT8** — Select mux mode: ALT8 mux port: ENET2_RX_EN of instance: enet2

11.6.113 SW_MUX_CTL_PAD_GPIO_SD_B1_00 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B1_00)

SW_MUX_CTL Register

Address: 401F_8000h base + 1D4h offset = 401F_81D4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B1_00 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_SD_B1_00 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_SD_B1_00. 000 **ALT0** — Select mux mode: ALT0 mux port: USDHC2_DATA3 of instance: usdhc2 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXSPIB_DATA03 of instance: flexspi 010 **ALT2** — Select mux mode: ALT2 mux port: FLEXPWM1_PWMA03 of instance: flexpwm1 011 **ALT3** — Select mux mode: ALT3 mux port: SAI1_TX_DATA03 of instance: sai1 100 **ALT4** — Select mux mode: ALT4 mux port: LPUART4_TX of instance: lpuart4 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO00 of instance: gpio3 1000 **ALT8** — Select mux mode: ALT8 mux port: SAI3_RX_DATA of instance: sai3

11.6.114 SW_MUX_CTL_PAD_GPIO_SD_B1_01 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B1_01)

SW_MUX_CTL Register

Address: 401F_8000h base + 1D8h offset = 401F_81D8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B1_01 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_SD_B1_01 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_SD_B1_01. 000 **ALT0** — Select mux mode: ALT0 mux port: USDHC2_DATA2 of instance: usdhc2 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXSPIB_DATA02 of instance: flexspi 010 **ALT2** — Select mux mode: ALT2 mux port: FLEXPWM1_PWMB03 of instance: flexpwm1 011 **ALT3** — Select mux mode: ALT3 mux port: SAI1_TX_DATA02 of instance: sai1 100 **ALT4** — Select mux mode: ALT4 mux port: LPUART4_RX of instance: lpuart4 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO01 of instance: gpio3 1000 **ALT8** — Select mux mode: ALT8 mux port: SAI3_TX_DATA of instance: sai3

11.6.115 SW_MUX_CTL_PAD_GPIO_SD_B1_02 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B1_02)

SW_MUX_CTL Register

Address: 401F_8000h base + 1DCh offset = 401F_81DCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B1_02 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_SD_B1_02 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_SD_B1_02. 000 **ALT0** — Select mux mode: ALT0 mux port: USDHC2_DATA1 of instance: usdhc2 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXSPIB_DATA01 of instance: flexspi 010 **ALT2** — Select mux mode: ALT2 mux port: FLEXPWM2_PWMA03 of instance: flexpwm2 011 **ALT3** — Select mux mode: ALT3 mux port: SAI1_TX_DATA01 of instance: sai1 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXCAN1_TX of instance: flexcan1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO02 of instance: gpio3 110 **ALT6** — Select mux mode: ALT6 mux port: CCM_WAIT of instance: ccm 1000 **ALT8** — Select mux mode: ALT8 mux port: SAI3_TX_SYNC of instance: sai3

11.6.116 SW_MUX_CTL_PAD_GPIO_SD_B1_03 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B1_03)

SW_MUX_CTL Register

Address: 401F_8000h base + 1E0h offset = 401F_81E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B1_03 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_SD_B1_03 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_SD_B1_03. 000 **ALT0** — Select mux mode: ALT0 mux port: USDHC2_DATA0 of instance: usdhc2 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXSPIB_DATA00 of instance: flexspi 010 **ALT2** — Select mux mode: ALT2 mux port: FLEXPWM2_PWMB03 of instance: flexpwm2 011 **ALT3** — Select mux mode: ALT3 mux port: SAI1_MCLK of instance: sai1 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXCAN1_RX of instance: flexcan1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO03 of instance: gpio3 110 **ALT6** — Select mux mode: ALT6 mux port: CCM_PMIC_READY of instance: ccm 1000 **ALT8** — Select mux mode: ALT8 mux port: SAI3_TX_BCLK of instance: sai3

11.6.117 SW_MUX_CTL_PAD_GPIO_SD_B1_04 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B1_04)

SW_MUX_CTL Register

Address: 401F_8000h base + 1E4h offset = 401F_81E4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B1_04 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_SD_B1_04 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_SD_B1_04. 000 **ALT0** — Select mux mode: ALT0 mux port: USDHC2_CLK of instance: usdhc2 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXSPIB_SCLK of instance: flexspi 010 **ALT2** — Select mux mode: ALT2 mux port: LPI2C1_SCL of instance: lpi2c1 011 **ALT3** — Select mux mode: ALT3 mux port: SAI1_RX_SYNC of instance: sai1 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXSPIA_SS1_B of instance: flexspi 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO04 of instance: gpio3 110 **ALT6** — Select mux mode: ALT6 mux port: CCM_STOP of instance: ccm 1000 **ALT8** — Select mux mode: ALT8 mux port: SAI3_MCLK of instance: sai3

11.6.118 SW_MUX_CTL_PAD_GPIO_SD_B1_05 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B1_05)

SW_MUX_CTL Register

Address: 401F_8000h base + 1E8h offset = 401F_81E8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B1_05 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_SD_B1_05 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_SD_B1_05. 000 **ALT0** — Select mux mode: ALT0 mux port: USDHC2_CMD of instance: usdhc2 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXSPIA_DQS of instance: flexspi 010 **ALT2** — Select mux mode: ALT2 mux port: LPI2C1_SDA of instance: lpi2c1 011 **ALT3** — Select mux mode: ALT3 mux port: SAI1_RX_BCLK of instance: sai1 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXSPIB_SS0_B of instance: flexspi 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO05 of instance: gpio3 1000 **ALT8** — Select mux mode: ALT8 mux port: SAI3_RX_SYNC of instance: sai3

11.6.119 SW_MUX_CTL_PAD_GPIO_SD_B1_06 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B1_06)

SW_MUX_CTL Register

Address: 401F_8000h base + 1ECh offset = 401F_81ECh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved											SION	MUX_MODE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1

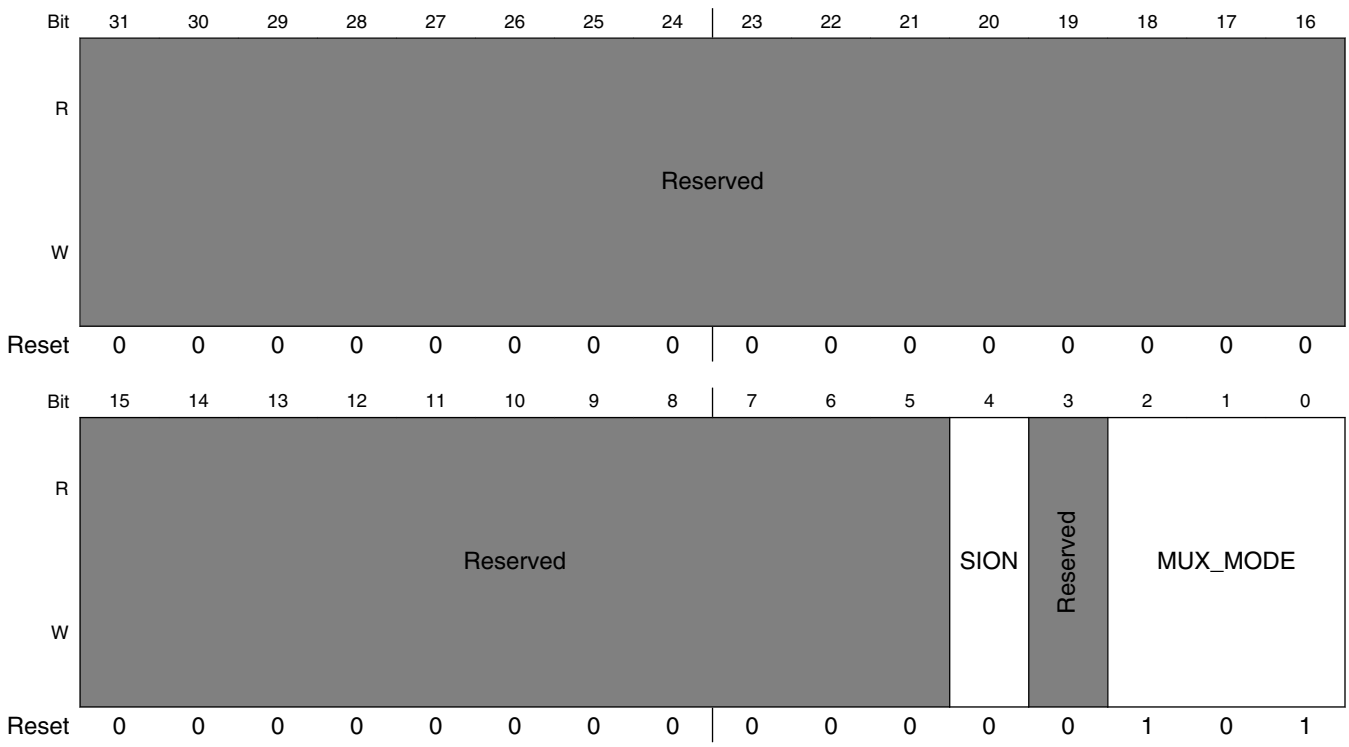
IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B1_06 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_SD_B1_06 0 **DISABLED** — Input Path is determined by functionality
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_SD_B1_06. 000 **ALT0** — Select mux mode: ALT0 mux port: USDHC2_RESET_B of instance: usdhc2 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXSPIA_SS0_B of instance: flexspi 010 **ALT2** — Select mux mode: ALT2 mux port: LPUART7_CTS_B of instance: lpuart7 011 **ALT3** — Select mux mode: ALT3 mux port: SAI1_RX_DATA00 of instance: sai1 100 **ALT4** — Select mux mode: ALT4 mux port: LPSPI2_PCS0 of instance: lpspi2 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO06 of instance: gpio3 1000 **ALT8** — Select mux mode: ALT8 mux port: SAI3_RX_BCLK of instance: sai3

11.6.120 SW_MUX_CTL_PAD_GPIO_SD_B1_07 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B1_07)

SW_MUX_CTL Register

Address: 401F_8000h base + 1F0h offset = 401F_81F0h



IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B1_07 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_SD_B1_07 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_SD_B1_07. 000 **ALT0** — Select mux mode: ALT0 mux port: SEMC_CSX01 of instance: semc 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXSPIA_SCLK of instance: flexspi

Table continues on the next page...

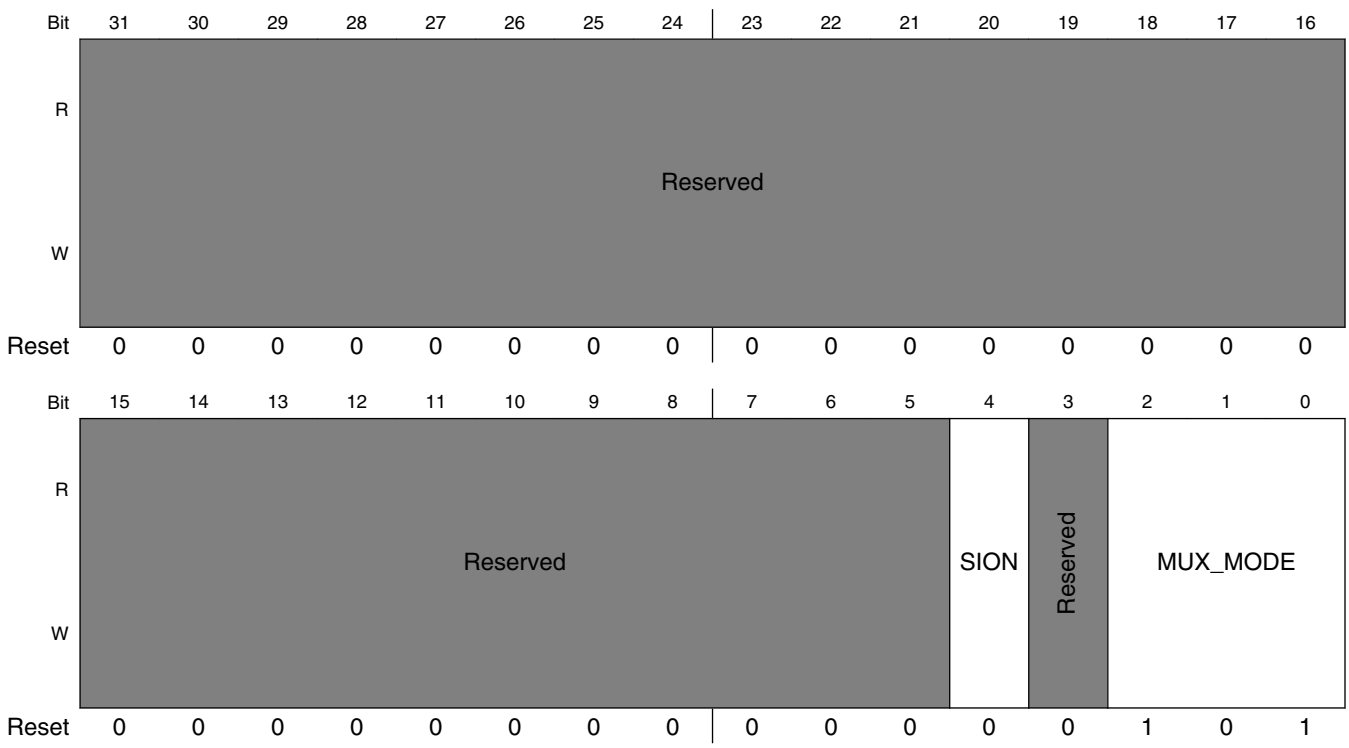
IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B1_07 field descriptions (continued)

Field	Description
	010 **ALT2** — Select mux mode: ALT2 mux port: LPUART7_RTS_B of instance: lpuart7 011 **ALT3** — Select mux mode: ALT3 mux port: SAI1_TX_DATA00 of instance: sai1 100 **ALT4** — Select mux mode: ALT4 mux port: LPSPI2_SCK of instance: lpspi2 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO07 of instance: gpio3

11.6.121 SW_MUX_CTL_PAD_GPIO_SD_B1_08 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B1_08)

SW_MUX_CTL Register

Address: 401F_8000h base + 1F4h offset = 401F_81F4h



IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B1_08 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality.

Table continues on the next page...

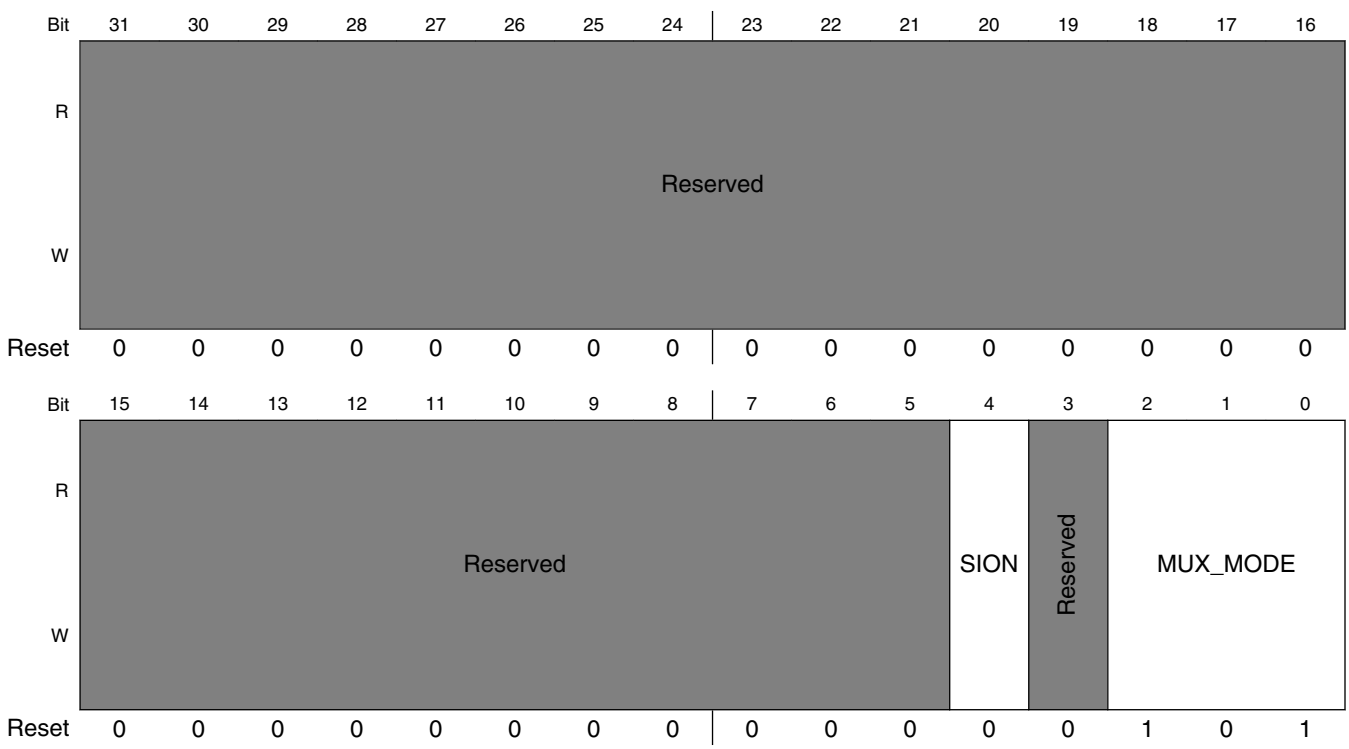
IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B1_08 field descriptions (continued)

Field	Description
	1 **ENABLED** — Force input path of pad GPIO_SD_B1_08 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_SD_B1_08. 000 **ALT0** — Select mux mode: ALT0 mux port: USDHC2_DATA4 of instance: usdhc2 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXSPIA_DATA00 of instance: flexspi 010 **ALT2** — Select mux mode: ALT2 mux port: LPUART7_TX of instance: lpuart7 011 **ALT3** — Select mux mode: ALT3 mux port: SAI1_TX_BCLK of instance: sai1 100 **ALT4** — Select mux mode: ALT4 mux port: LPSPI2_SD0 of instance: lpspi2 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO08 of instance: gpio3 110 **ALT6** — Select mux mode: ALT6 mux port: SEMC_CSX02 of instance: semc

11.6.122 SW_MUX_CTL_PAD_GPIO_SD_B1_09 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B1_09)

SW_MUX_CTL Register

Address: 401F_8000h base + 1F8h offset = 401F_81F8h



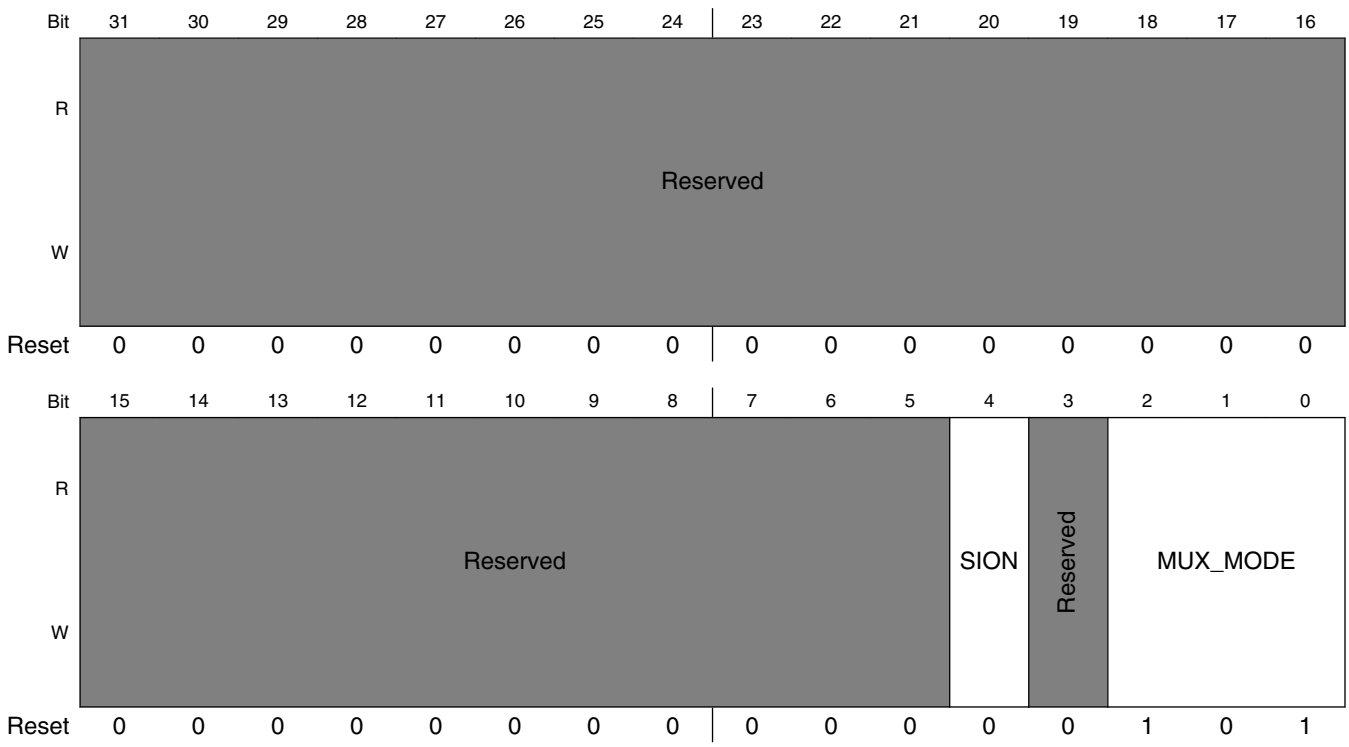
IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B1_09 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_SD_B1_09 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_SD_B1_09. 000 **ALT0** — Select mux mode: ALT0 mux port: USDHC2_DATA5 of instance: usdhc2 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXSPIA_DATA01 of instance: flexspi 010 **ALT2** — Select mux mode: ALT2 mux port: LPUART7_RX of instance: lpuart7 011 **ALT3** — Select mux mode: ALT3 mux port: SAI1_TX_SYNC of instance: sai1 100 **ALT4** — Select mux mode: ALT4 mux port: LPSPI2_SDI of instance: lpspi2 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO09 of instance: gpio3

11.6.123 SW_MUX_CTL_PAD_GPIO_SD_B1_10 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B1_10)

SW_MUX_CTL Register

Address: 401F_8000h base + 1FCh offset = 401F_81FCh



IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B1_10 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 1 **ENABLED** — Force input path of pad GPIO_SD_B1_10 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_SD_B1_10. 000 **ALT0** — Select mux mode: ALT0 mux port: USDHC2_DATA6 of instance: usdhc2 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXSPIA_DATA02 of instance: flexspi

Table continues on the next page...

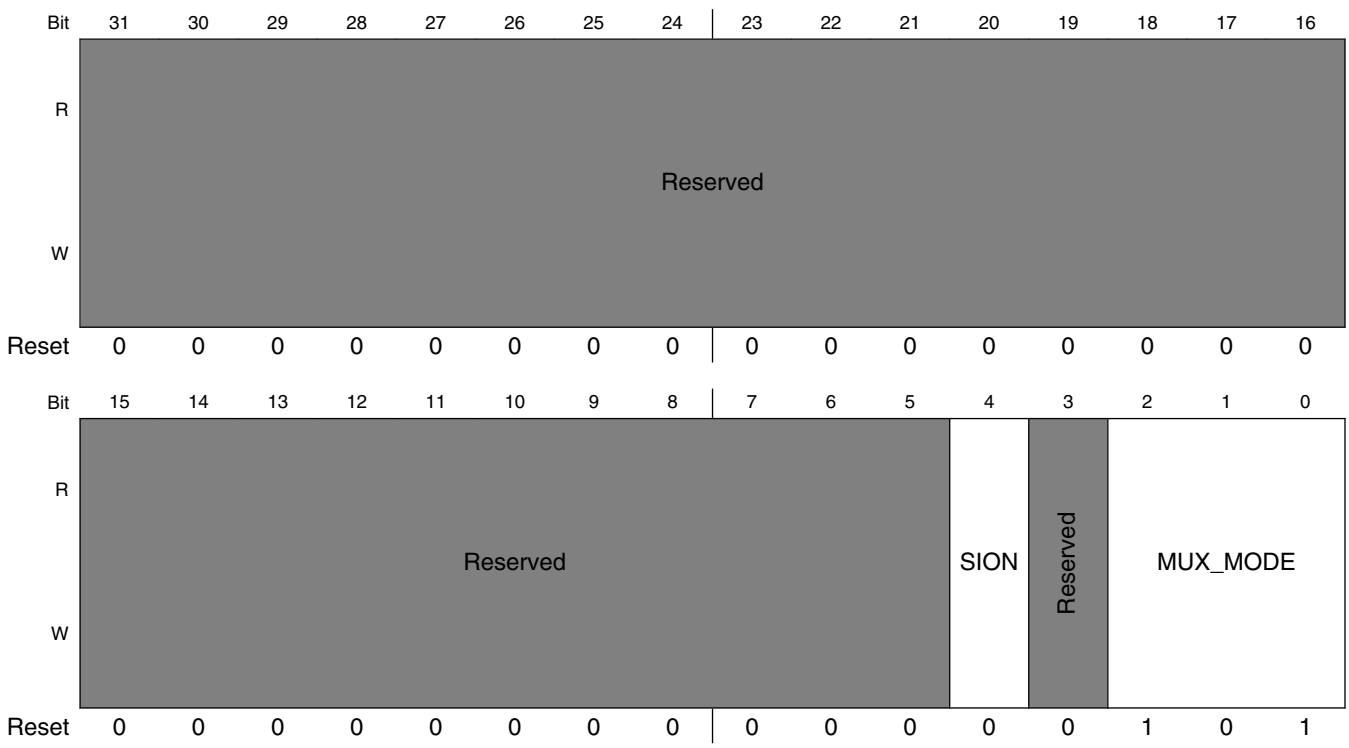
IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B1_10 field descriptions (continued)

Field	Description
	010 **ALT2** — Select mux mode: ALT2 mux port: LPUART2_RX of instance: lpuart2 011 **ALT3** — Select mux mode: ALT3 mux port: LPI2C2_SDA of instance: lpi2c2 100 **ALT4** — Select mux mode: ALT4 mux port: LPSPI2_PCS2 of instance: lpspi2 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO10 of instance: gpio3

11.6.124 SW_MUX_CTL_PAD_GPIO_SD_B1_11 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B1_11)

SW_MUX_CTL Register

Address: 401F_8000h base + 200h offset = 401F_8200h



IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B1_11 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality.

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_B1_11 field descriptions (continued)

Field	Description
	1 **ENABLED** — Force input path of pad GPIO_SD_B1_11 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select one of iomux modes to be used for pad: GPIO_SD_B1_11. 000 **ALT0** — Select mux mode: ALT0 mux port: USDHC2_DATA7 of instance: usdhc2 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXSPIA_DATA03 of instance: flexspi 010 **ALT2** — Select mux mode: ALT2 mux port: LPUART2_TX of instance: lpuart2 011 **ALT3** — Select mux mode: ALT3 mux port: LPI2C2_SCL of instance: lpi2c2 100 **ALT4** — Select mux mode: ALT4 mux port: LPSPI2_PCS3 of instance: lpspi2 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO3_IO11 of instance: gpio3

11.6.125 SW_PAD_CTL_PAD_GPIO_EMC_00 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_00)

SW_PAD_CTL Register

Address: 401F_8000h base + 204h offset = 401F_8204h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_00 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_EMC_00 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_EMC_00 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_00 field descriptions (continued)

Field	Description
	01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_EMC_00 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_EMC_00 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_EMC_00 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_EMC_00 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_EMC_00 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_EMC_00

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_00 field descriptions (continued)

Field	Description
	0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.126 SW_PAD_CTL_PAD_GPIO_EMC_01 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_01)

SW_PAD_CTL Register

Address: 401F_8000h base + 208h offset = 401F_8208h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_01 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_EMC_01 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_EMC_01 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_EMC_01 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_EMC_01

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_01 field descriptions (continued)

Field	Description
	0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_EMC_01 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_EMC_01 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_EMC_01 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_EMC_01 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.127 SW_PAD_CTL_PAD_GPIO_EMC_02 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_02)

SW_PAD_CTL Register

Address: 401F_8000h base + 20Ch offset = 401F_820Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_02 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_EMC_02 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_EMC_02 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_EMC_02 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_EMC_02 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_EMC_02

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_02 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_EMC_02 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_EMC_02 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_EMC_02 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.128 SW_PAD_CTL_PAD_GPIO_EMC_03 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_03)

SW_PAD_CTL Register

Address: 401F_8000h base + 210h offset = 401F_8210h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_03 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_EMC_03 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_EMC_03 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_EMC_03 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_EMC_03 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_EMC_03 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_EMC_03 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_EMC_03 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_03 field descriptions (continued)

Field	Description
	011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_EMC_03 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.129 SW_PAD_CTL_PAD_GPIO_EMC_04 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_04)

pin 2

SW_PAD_CTL Register

Address: 401F_8000h base + 214h offset = 401F_8214h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_04 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_EMC_04 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_EMC_04 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_04 field descriptions (continued) pin 2

Field	Description
	10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_EMC_04 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_EMC_04 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_EMC_04 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_EMC_04 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_EMC_04 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_EMC_04 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.130 SW_PAD_CTL_PAD_GPIO_EMC_05 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_05)

pin 3

SW_PAD_CTL Register

Address: 401F_8000h base + 218h offset = 401F_8218h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_05 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_EMC_05 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_EMC_05 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_EMC_05 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_EMC_05 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_EMC_05

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_05 field descriptions (continued) pin 3

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_EMC_05 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_EMC_05 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_EMC_05 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.131 SW_PAD_CTL_PAD_GPIO_EMC_06 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_06) pin 4

SW_PAD_CTL Register

Address: 401F_8000h base + 21Ch offset = 401F_821Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_06 field descriptions pin 4

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_EMC_06 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_EMC_06 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_EMC_06 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_EMC_06 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_EMC_06 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_EMC_06 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_EMC_06 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_06 field descriptions (continued) pin 4

Field	Description
	011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_EMC_06 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.132 SW_PAD_CTL_PAD_GPIO_EMC_07 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_07) pin 33

SW_PAD_CTL Register

Address: 401F_8000h base + 220h offset = 401F_8220h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_07 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_EMC_07 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_EMC_07 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_07 field descriptions (continued) pin 33

Field	Description
	10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_EMC_07 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_EMC_07 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_EMC_07 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_EMC_07 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_EMC_07 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_EMC_07 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.133 SW_PAD_CTL_PAD_GPIO_EMC_08 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_08) pin 5

SW_PAD_CTL Register

Address: 401F_8000h base + 224h offset = 401F_8224h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_08 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_EMC_08 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_EMC_08 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_EMC_08 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_EMC_08 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_EMC_08

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_08 field descriptions (continued) pin 5

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_EMC_08 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_EMC_08 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_EMC_08 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.134 SW_PAD_CTL_PAD_GPIO_EMC_09 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_09)

SW_PAD_CTL Register

Address: 401F_8000h base + 228h offset = 401F_8228h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_09 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_EMC_09 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_EMC_09 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_EMC_09 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_EMC_09 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_EMC_09 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_EMC_09 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_EMC_09 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_09 field descriptions (continued)

Field	Description
	011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_EMC_09 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.135 SW_PAD_CTL_PAD_GPIO_EMC_10 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_10)

SW_PAD_CTL Register

Address: 401F_8000h base + 22Ch offset = 401F_822Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_10 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_EMC_10 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_EMC_10 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_10 field descriptions (continued)

Field	Description
	10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_EMC_10 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_EMC_10 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_EMC_10 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_EMC_10 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_EMC_10 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_EMC_10 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.136 SW_PAD_CTL_PAD_GPIO_EMC_11 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_11)

SW_PAD_CTL Register

Address: 401F_8000h base + 230h offset = 401F_8230h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_11 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_EMC_11 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_EMC_11 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_EMC_11 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_EMC_11 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_EMC_11

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_11 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_EMC_11 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_EMC_11 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_EMC_11 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.137 SW_PAD_CTL_PAD_GPIO_EMC_12 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_12)

SW_PAD_CTL Register

Address: 401F_8000h base + 234h offset = 401F_8234h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_12 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_EMC_12 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_EMC_12 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_EMC_12 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_EMC_12 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_EMC_12 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_EMC_12 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_EMC_12 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_12 field descriptions (continued)

Field	Description
	011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_EMC_12 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.138 SW_PAD_CTL_PAD_GPIO_EMC_13 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_13)

SW_PAD_CTL Register

Address: 401F_8000h base + 238h offset = 401F_8238h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_13 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_EMC_13 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_EMC_13 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_13 field descriptions (continued)

Field	Description
	10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_EMC_13 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_EMC_13 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_EMC_13 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_EMC_13 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_EMC_13 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_EMC_13 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.139 SW_PAD_CTL_PAD_GPIO_EMC_14 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_14)

SW_PAD_CTL Register

Address: 401F_8000h base + 23Ch offset = 401F_823Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_14 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_EMC_14 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_EMC_14 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_EMC_14 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_EMC_14 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_EMC_14

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_14 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_EMC_14 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_EMC_14 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_EMC_14 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.140 SW_PAD_CTL_PAD_GPIO_EMC_15 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_15)

SW_PAD_CTL Register

Address: 401F_8000h base + 240h offset = 401F_8240h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_15 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_EMC_15 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_EMC_15 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_EMC_15 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_EMC_15 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_EMC_15 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_EMC_15 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_EMC_15 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_15 field descriptions (continued)

Field	Description
	011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_EMC_15 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.141 SW_PAD_CTL_PAD_GPIO_EMC_16 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_16)

SW_PAD_CTL Register

Address: 401F_8000h base + 244h offset = 401F_8244h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_16 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_EMC_16 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_EMC_16 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_16 field descriptions (continued)

Field	Description
	10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_EMC_16 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_EMC_16 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_EMC_16 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_EMC_16 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_EMC_16 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_EMC_16 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.142 SW_PAD_CTL_PAD_GPIO_EMC_17 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_17)

SW_PAD_CTL Register

Address: 401F_8000h base + 248h offset = 401F_8248h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_17 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_EMC_17 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_EMC_17 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_EMC_17 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_EMC_17 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_EMC_17

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_17 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_EMC_17 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_EMC_17 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_EMC_17 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.143 SW_PAD_CTL_PAD_GPIO_EMC_18 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_18)

SW_PAD_CTL Register

Address: 401F_8000h base + 24Ch offset = 401F_824Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_18 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_EMC_18 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_EMC_18 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_EMC_18 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_EMC_18 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_EMC_18 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_EMC_18 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_EMC_18 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_18 field descriptions (continued)

Field	Description
	011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_EMC_18 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.144 SW_PAD_CTL_PAD_GPIO_EMC_19 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_19)

SW_PAD_CTL Register

Address: 401F_8000h base + 250h offset = 401F_8250h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_19 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_EMC_19 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_EMC_19 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_19 field descriptions (continued)

Field	Description
	10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_EMC_19 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_EMC_19 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_EMC_19 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_EMC_19 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_EMC_19 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_EMC_19 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.145 SW_PAD_CTL_PAD_GPIO_EMC_20 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_20)

SW_PAD_CTL Register

Address: 401F_8000h base + 254h offset = 401F_8254h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_20 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_EMC_20 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_EMC_20 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_EMC_20 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_EMC_20 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_EMC_20

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_20 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_EMC_20 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_EMC_20 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_EMC_20 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.146 SW_PAD_CTL_PAD_GPIO_EMC_21 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_21)

SW_PAD_CTL Register

Address: 401F_8000h base + 258h offset = 401F_8258h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_21 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_EMC_21 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_EMC_21 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_EMC_21 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_EMC_21 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_EMC_21 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_EMC_21 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_EMC_21 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_21 field descriptions (continued)

Field	Description
	011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_EMC_21 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.147 SW_PAD_CTL_PAD_GPIO_EMC_22 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_22)

SW_PAD_CTL Register

Address: 401F_8000h base + 25Ch offset = 401F_825Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_22 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_EMC_22 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_EMC_22 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_22 field descriptions (continued)

Field	Description
	10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_EMC_22 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_EMC_22 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_EMC_22 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_EMC_22 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_EMC_22 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_EMC_22 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.148 SW_PAD_CTL_PAD_GPIO_EMC_23 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_23)

SW_PAD_CTL Register

Address: 401F_8000h base + 260h offset = 401F_8260h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_23 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_EMC_23 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_EMC_23 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_EMC_23 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_EMC_23 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_EMC_23

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_23 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_EMC_23 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_EMC_23 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_EMC_23 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.149 SW_PAD_CTL_PAD_GPIO_EMC_24 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_24)

SW_PAD_CTL Register

Address: 401F_8000h base + 264h offset = 401F_8264h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_24 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_EMC_24 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_EMC_24 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_EMC_24 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_EMC_24 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_EMC_24 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_EMC_24 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_EMC_24 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_24 field descriptions (continued)

Field	Description
	011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_EMC_24 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.150 SW_PAD_CTL_PAD_GPIO_EMC_25 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_25)

SW_PAD_CTL Register

Address: 401F_8000h base + 268h offset = 401F_8268h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_25 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_EMC_25 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_EMC_25 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_25 field descriptions (continued)

Field	Description
	10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_EMC_25 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_EMC_25 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_EMC_25 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_EMC_25 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_EMC_25 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_EMC_25 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.151 SW_PAD_CTL_PAD_GPIO_EMC_26 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_26)

SW_PAD_CTL Register

Address: 401F_8000h base + 26Ch offset = 401F_826Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_26 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_EMC_26 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_EMC_26 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_EMC_26 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_EMC_26 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_EMC_26

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_26 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_EMC_26 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_EMC_26 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_EMC_26 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.152 SW_PAD_CTL_PAD_GPIO_EMC_27 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_27)

SW_PAD_CTL Register

Address: 401F_8000h base + 270h offset = 401F_8270h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	1	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_27 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_EMC_27 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_EMC_27 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_EMC_27 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_EMC_27 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_EMC_27 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_EMC_27 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_EMC_27 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_27 field descriptions (continued)

Field	Description
	011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_EMC_27 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.153 SW_PAD_CTL_PAD_GPIO_EMC_28 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_28)

SW_PAD_CTL Register

Address: 401F_8000h base + 274h offset = 401F_8274h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_28 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_EMC_28 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_EMC_28 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_28 field descriptions (continued)

Field	Description
	10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_EMC_28 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_EMC_28 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_EMC_28 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_EMC_28 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_EMC_28 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_EMC_28 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.154 SW_PAD_CTL_PAD_GPIO_EMC_29 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_29)

SW_PAD_CTL Register

Address: 401F_8000h base + 278h offset = 401F_8278h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_29 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_EMC_29 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_EMC_29 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_EMC_29 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_EMC_29 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_EMC_29

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_29 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_EMC_29 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_EMC_29 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_EMC_29 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.155 SW_PAD_CTL_PAD_GPIO_EMC_30 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_30)

SW_PAD_CTL Register

Address: 401F_8000h base + 27Ch offset = 401F_827Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_30 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_EMC_30 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_EMC_30 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_EMC_30 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_EMC_30 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_EMC_30 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_EMC_30 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_EMC_30 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_30 field descriptions (continued)

Field	Description
	011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_EMC_30 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.156 SW_PAD_CTL_PAD_GPIO_EMC_31 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_31)

pin 29

SW_PAD_CTL Register

Address: 401F_8000h base + 280h offset = 401F_8280h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_31 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_EMC_31 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_EMC_31 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_31 field descriptions (continued) pin 29

Field	Description
	10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_EMC_31 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_EMC_31 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_EMC_31 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_EMC_31 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_EMC_31 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_EMC_31 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.157 SW_PAD_CTL_PAD_GPIO_EMC_32 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_32) pin 28

SW_PAD_CTL Register

Address: 401F_8000h base + 284h offset = 401F_8284h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_32 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_EMC_32 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_EMC_32 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_EMC_32 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_EMC_32 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_EMC_32

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_32 field descriptions (continued) pin 28

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_EMC_32 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_EMC_32 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_EMC_32 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.158 SW_PAD_CTL_PAD_GPIO_EMC_33 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_33)

SW_PAD_CTL Register

Address: 401F_8000h base + 288h offset = 401F_8288h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_33 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_EMC_33 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_EMC_33 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_EMC_33 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_EMC_33 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_EMC_33 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_EMC_33 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_EMC_33 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_33 field descriptions (continued)

Field	Description
	011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_EMC_33 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.159 SW_PAD_CTL_PAD_GPIO_EMC_34 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_34)

SW_PAD_CTL Register

Address: 401F_8000h base + 28Ch offset = 401F_828Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_34 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_EMC_34 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_EMC_34 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_34 field descriptions (continued)

Field	Description
	10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_EMC_34 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_EMC_34 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_EMC_34 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_EMC_34 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_EMC_34 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_EMC_34 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.160 SW_PAD_CTL_PAD_GPIO_EMC_35 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_35)

SW_PAD_CTL Register

Address: 401F_8000h base + 290h offset = 401F_8290h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_35 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_EMC_35 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_EMC_35 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_EMC_35 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_EMC_35 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_EMC_35

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_35 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_EMC_35 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_EMC_35 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_EMC_35 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.161 SW_PAD_CTL_PAD_GPIO_EMC_36 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_36)

pin 31

SW_PAD_CTL Register

Address: 401F_8000h base + 294h offset = 401F_8294h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_36 field descriptions pin 31

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_EMC_36 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_EMC_36 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_EMC_36 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_EMC_36 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_EMC_36 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_EMC_36 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_EMC_36 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_36 field descriptions (continued) pin 31

Field	Description
	011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_EMC_36 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.162 SW_PAD_CTL_PAD_GPIO_EMC_37 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_37) pin 30

SW_PAD_CTL Register

Address: 401F_8000h base + 298h offset = 401F_8298h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_37 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_EMC_37 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_EMC_37 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_37 field descriptions (continued) pin 30

Field	Description
	10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_EMC_37 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_EMC_37 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_EMC_37 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_EMC_37 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_EMC_37 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_EMC_37 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.163 SW_PAD_CTL_PAD_GPIO_EMC_38 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_38)

SW_PAD_CTL Register

Address: 401F_8000h base + 29Ch offset = 401F_829Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_38 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_EMC_38 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_EMC_38 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_EMC_38 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_EMC_38 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_EMC_38

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_38 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_EMC_38 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_EMC_38 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_EMC_38 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.164 SW_PAD_CTL_PAD_GPIO_EMC_39 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_39)

SW_PAD_CTL Register

Address: 401F_8000h base + 2A0h offset = 401F_82A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_39 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_EMC_39 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_EMC_39 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_EMC_39 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_EMC_39 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_EMC_39 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_EMC_39 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_EMC_39 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_39 field descriptions (continued)

Field	Description
	011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_EMC_39 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.165 SW_PAD_CTL_PAD_GPIO_EMC_40 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_40)

SW_PAD_CTL Register

Address: 401F_8000h base + 2A4h offset = 401F_82A4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_40 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_EMC_40 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_EMC_40 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_40 field descriptions (continued)

Field	Description
	10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_EMC_40 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_EMC_40 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_EMC_40 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_EMC_40 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_EMC_40 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_EMC_40 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.166 SW_PAD_CTL_PAD_GPIO_EMC_41 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_41)

SW_PAD_CTL Register

Address: 401F_8000h base + 2A8h offset = 401F_82A8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_41 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_EMC_41 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_EMC_41 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_EMC_41 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_EMC_41 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_EMC_41

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_EMC_41 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_EMC_41 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_EMC_41 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_EMC_41 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.167 SW_PAD_CTL_PAD_GPIO_AD_B0_00 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_00)

SW_PAD_CTL Register

Address: 401F_8000h base + 2ACh offset = 401F_82ACh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_00 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_AD_B0_00 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_AD_B0_00 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_AD_B0_00 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_AD_B0_00 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_AD_B0_00

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_00 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_AD_B0_00 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_AD_B0_00 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_AD_B0_00 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.168 SW_PAD_CTL_PAD_GPIO_AD_B0_01 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_01)

SW_PAD_CTL Register

Address: 401F_8000h base + 2B0h offset = 401F_82B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_01 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_AD_B0_01 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_AD_B0_01 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_AD_B0_01 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_AD_B0_01 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_AD_B0_01

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_01 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_AD_B0_01 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_AD_B0_01 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_AD_B0_01 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.169 SW_PAD_CTL_PAD_GPIO_AD_B0_02 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_02)

pin 1

SW_PAD_CTL Register

Address: 401F_8000h base + 2B4h offset = 401F_82B4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_02 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_AD_B0_02 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_AD_B0_02 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_AD_B0_02 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_AD_B0_02 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_AD_B0_02

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_02 field descriptions (continued) pin 1

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_AD_B0_02 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_AD_B0_02 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_AD_B0_02 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.170 SW_PAD_CTL_PAD_GPIO_AD_B0_03 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_03)

pin

SW_PAD_CTL Register

Address: 401F_8000h base + 2B8h offset = 401F_82B8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_03 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_AD_B0_03 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_AD_B0_03 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_AD_B0_03 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_AD_B0_03 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_AD_B0_03

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_03 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_AD_B0_03 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_AD_B0_03 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_AD_B0_03 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.171 SW_PAD_CTL_PAD_GPIO_AD_B0_04 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_04)

SW_PAD_CTL Register

Address: 401F_8000h base + 2BCh offset = 401F_82BCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	1	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_04 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_AD_B0_04 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_AD_B0_04 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_AD_B0_04 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_AD_B0_04 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_AD_B0_04

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_04 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_AD_B0_04 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_AD_B0_04 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_AD_B0_04 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.172 SW_PAD_CTL_PAD_GPIO_AD_B0_05 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_05)

SW_PAD_CTL Register

Address: 401F_8000h base + 2C0h offset = 401F_82C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	1	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_05 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_AD_B0_05 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_AD_B0_05 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_AD_B0_05 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_AD_B0_05 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_AD_B0_05

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_05 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_AD_B0_05 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_AD_B0_05 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_AD_B0_05 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.173 SW_PAD_CTL_PAD_GPIO_AD_B0_06 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_06)

SW_PAD_CTL Register

Address: 401F_8000h base + 2C4h offset = 401F_82C4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	1	1	1	0	0	0	0	1	0	1	0	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_06 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_AD_B0_06 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_AD_B0_06 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_AD_B0_06 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_AD_B0_06 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_AD_B0_06

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_06 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_AD_B0_06 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_AD_B0_06 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_AD_B0_06 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.174 SW_PAD_CTL_PAD_GPIO_AD_B0_07 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_07)

SW_PAD_CTL Register

Address: 401F_8000h base + 2C8h offset = 401F_82C8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	1	1	0	0	0	0	1	0	1	0	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_07 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_AD_B0_07 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_AD_B0_07 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_AD_B0_07 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_AD_B0_07 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_AD_B0_07

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_07 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_AD_B0_07 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_AD_B0_07 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_AD_B0_07 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.175 SW_PAD_CTL_PAD_GPIO_AD_B0_08 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_08)

SW_PAD_CTL Register

Address: 401F_8000h base + 2CCh offset = 401F_82CCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	1	1	0	0	0	0	1	0	1	0	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_08 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_AD_B0_08 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_AD_B0_08 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_AD_B0_08 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_AD_B0_08 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_AD_B0_08

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_08 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_AD_B0_08 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_AD_B0_08 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_AD_B0_08 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.176 SW_PAD_CTL_PAD_GPIO_AD_B0_09 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_09)

SW_PAD_CTL Register

Address: 401F_8000h base + 2D0h offset = 401F_82D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	1	1	1	0	0	0	0	1	0	1	0	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_09 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_AD_B0_09 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_AD_B0_09 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_AD_B0_09 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_AD_B0_09 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_AD_B0_09

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_09 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_AD_B0_09 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_AD_B0_09 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_AD_B0_09 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.177 SW_PAD_CTL_PAD_GPIO_AD_B0_10 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_10)

SW_PAD_CTL Register

Address: 401F_8000h base + 2D4h offset = 401F_82D4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	1	0	0	1	0	0	0	0	1	0	1	1	0	0	0	1

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_10 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_AD_B0_10 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_AD_B0_10 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_AD_B0_10 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_AD_B0_10 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_AD_B0_10

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_10 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_AD_B0_10 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_AD_B0_10 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_AD_B0_10 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.178 SW_PAD_CTL_PAD_GPIO_AD_B0_11 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_11)

SW_PAD_CTL Register

Address: 401F_8000h base + 2D8h offset = 401F_82D8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	1	1	1	0	0	0	0	1	0	1	0	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_11 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_AD_B0_11 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_AD_B0_11 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_AD_B0_11 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_AD_B0_11 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_AD_B0_11

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_11 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_AD_B0_11 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_AD_B0_11 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_AD_B0_11 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.179 SW_PAD_CTL_PAD_GPIO_AD_B0_12 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_12)

pin 24

SW_PAD_CTL Register

Address: 401F_8000h base + 2DCh offset = 401F_82DCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_12 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_AD_B0_12 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_AD_B0_12 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_AD_B0_12 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_AD_B0_12 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_AD_B0_12

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_12 field descriptions (continued) pin 24

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_AD_B0_12 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_AD_B0_12 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_AD_B0_12 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.180 SW_PAD_CTL_PAD_GPIO_AD_B0_13 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_13)

pin 25

SW_PAD_CTL Register

Address: 401F_8000h base + 2E0h offset = 401F_82E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_13 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_AD_B0_13 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_AD_B0_13 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_AD_B0_13 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_AD_B0_13 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_AD_B0_13

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_13 field descriptions (continued) pin 25

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_AD_B0_13 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_AD_B0_13 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_AD_B0_13 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.181 SW_PAD_CTL_PAD_GPIO_AD_B0_14 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_14)

SW_PAD_CTL Register

Address: 401F_8000h base + 2E4h offset = 401F_82E4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_14 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_AD_B0_14 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_AD_B0_14 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_AD_B0_14 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_AD_B0_14 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_AD_B0_14

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_14 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_AD_B0_14 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_AD_B0_14 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_AD_B0_14 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.182 SW_PAD_CTL_PAD_GPIO_AD_B0_15 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_15)

SW_PAD_CTL Register

Address: 401F_8000h base + 2E8h offset = 401F_82E8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_15 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_AD_B0_15 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_AD_B0_15 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_AD_B0_15 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_AD_B0_15 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_AD_B0_15

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B0_15 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_AD_B0_15 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_AD_B0_15 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_AD_B0_15 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.183 SW_PAD_CTL_PAD_GPIO_AD_B1_00 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_00)

pin 19

SW_PAD_CTL Register

Address: 401F_8000h base + 2ECh offset = 401F_82ECh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_00 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_AD_B1_00 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_AD_B1_00 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_AD_B1_00 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_AD_B1_00 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_AD_B1_00

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_00 field descriptions (continued) pin 19

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_AD_B1_00 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_AD_B1_00 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_AD_B1_00 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.184 SW_PAD_CTL_PAD_GPIO_AD_B1_01 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_01)

pin 18

SW_PAD_CTL Register

Address: 401F_8000h base + 2F0h offset = 401F_82F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_01 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_AD_B1_01 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_AD_B1_01 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_AD_B1_01 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_AD_B1_01 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_AD_B1_01

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_01 field descriptions (continued) pin 18

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_AD_B1_01 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_AD_B1_01 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_AD_B1_01 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.185 SW_PAD_CTL_PAD_GPIO_AD_B1_02 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_02)

pin 14

SW_PAD_CTL Register

Address: 401F_8000h base + 2F4h offset = 401F_82F4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_02 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_AD_B1_02 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_AD_B1_02 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_AD_B1_02 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_AD_B1_02 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_AD_B1_02

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_02 field descriptions (continued) pin 14

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_AD_B1_02 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_AD_B1_02 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_AD_B1_02 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.186 SW_PAD_CTL_PAD_GPIO_AD_B1_03 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_03)

pin 15

SW_PAD_CTL Register

Address: 401F_8000h base + 2F8h offset = 401F_82F8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_03 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_AD_B1_03 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_AD_B1_03 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_AD_B1_03 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_AD_B1_03 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_AD_B1_03

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_03 field descriptions (continued) pin 15

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_AD_B1_03 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_AD_B1_03 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_AD_B1_03 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.187 SW_PAD_CTL_PAD_GPIO_AD_B1_04 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_04)

pin 40 (T4.1)

SW_PAD_CTL Register

Address: 401F_8000h base + 2FCh offset = 401F_82FCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_04 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_AD_B1_04 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_AD_B1_04 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_AD_B1_04 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_AD_B1_04 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_AD_B1_04

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_04 field descriptions (continued) pin 40 (T4.1)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_AD_B1_04 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_AD_B1_04 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_AD_B1_04 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.188 SW_PAD_CTL_PAD_GPIO_AD_B1_05 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_05)

pin 41 (T4.1)

SW_PAD_CTL Register

Address: 401F_8000h base + 300h offset = 401F_8300h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_05 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_AD_B1_05 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_AD_B1_05 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_AD_B1_05 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_AD_B1_05 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_AD_B1_05

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_05 field descriptions (continued) pin 41 (T4.1)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_AD_B1_05 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_AD_B1_05 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_AD_B1_05 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.189 SW_PAD_CTL_PAD_GPIO_AD_B1_06 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_06)

pin 17

SW_PAD_CTL Register

Address: 401F_8000h base + 304h offset = 401F_8304h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_06 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_AD_B1_06 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_AD_B1_06 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_AD_B1_06 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_AD_B1_06 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_AD_B1_06

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_06 field descriptions (continued) pin 17

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_AD_B1_06 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_AD_B1_06 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_AD_B1_06 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.190 SW_PAD_CTL_PAD_GPIO_AD_B1_07 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_07)

pin 16

SW_PAD_CTL Register

Address: 401F_8000h base + 308h offset = 401F_8308h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_07 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_AD_B1_07 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_AD_B1_07 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_AD_B1_07 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_AD_B1_07 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_AD_B1_07

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_07 field descriptions (continued) pin 16

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_AD_B1_07 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_AD_B1_07 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_AD_B1_07 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.191 SW_PAD_CTL_PAD_GPIO_AD_B1_08 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_08)

pin 22

SW_PAD_CTL Register

Address: 401F_8000h base + 30Ch offset = 401F_830Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_08 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_AD_B1_08 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_AD_B1_08 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_AD_B1_08 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_AD_B1_08 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_AD_B1_08

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_08 field descriptions (continued) pin 22

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_AD_B1_08 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_AD_B1_08 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_AD_B1_08 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.192 SW_PAD_CTL_PAD_GPIO_AD_B1_09 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_09)

pin 23

SW_PAD_CTL Register

Address: 401F_8000h base + 310h offset = 401F_8310h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_09 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_AD_B1_09 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_AD_B1_09 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_AD_B1_09 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_AD_B1_09 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_AD_B1_09

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_09 field descriptions (continued) pin 23

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_AD_B1_09 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_AD_B1_09 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_AD_B1_09 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.193 SW_PAD_CTL_PAD_GPIO_AD_B1_10 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_10)

pin 20

SW_PAD_CTL Register

Address: 401F_8000h base + 314h offset = 401F_8314h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_10 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_AD_B1_10 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_AD_B1_10 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_AD_B1_10 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_AD_B1_10 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_AD_B1_10

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_10 field descriptions (continued) pin 20

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_AD_B1_10 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_AD_B1_10 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_AD_B1_10 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.194 SW_PAD_CTL_PAD_GPIO_AD_B1_11 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_11)

pin 21

SW_PAD_CTL Register

Address: 401F_8000h base + 318h offset = 401F_8318h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_11 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_AD_B1_11 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_AD_B1_11 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_AD_B1_11 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_AD_B1_11 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_AD_B1_11

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_11 field descriptions (continued) pin 21

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_AD_B1_11 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_AD_B1_11 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_AD_B1_11 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.195 SW_PAD_CTL_PAD_GPIO_AD_B1_12 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_12)

pin 38 (T4.1)

SW_PAD_CTL Register

Address: 401F_8000h base + 31Ch offset = 401F_831Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_12 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_AD_B1_12 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_AD_B1_12 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_AD_B1_12 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_AD_B1_12 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_AD_B1_12

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_12 field descriptions (continued) pin 38 (T4.1)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_AD_B1_12 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_AD_B1_12 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_AD_B1_12 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.196 SW_PAD_CTL_PAD_GPIO_AD_B1_13 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_13)

pin 39 (T4.1)

SW_PAD_CTL Register

Address: 401F_8000h base + 320h offset = 401F_8320h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_13 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_AD_B1_13 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_AD_B1_13 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_AD_B1_13 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_AD_B1_13 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_AD_B1_13

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_13 field descriptions (continued) pin 39 (T4.1)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_AD_B1_13 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_AD_B1_13 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_AD_B1_13 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.197 SW_PAD_CTL_PAD_GPIO_AD_B1_14 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_14)

pin 26

SW_PAD_CTL Register

Address: 401F_8000h base + 324h offset = 401F_8324h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_14 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_AD_B1_14 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_AD_B1_14 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_AD_B1_14 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_AD_B1_14 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_AD_B1_14

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_14 field descriptions (continued) pin 26

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_AD_B1_14 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_AD_B1_14 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_AD_B1_14 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.198 SW_PAD_CTL_PAD_GPIO_AD_B1_15 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_15)

pin 27

SW_PAD_CTL Register

Address: 401F_8000h base + 328h offset = 401F_8328h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_15 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_AD_B1_15 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_AD_B1_15 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_AD_B1_15 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_AD_B1_15 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_AD_B1_15

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_B1_15 field descriptions (continued) pin 27

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_AD_B1_15 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_AD_B1_15 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_AD_B1_15 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.199 SW_PAD_CTL_PAD_GPIO_B0_00 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_00) pin 10

SW_PAD_CTL Register

Address: 401F_8000h base + 32Ch offset = 401F_832Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_00 field descriptions pin 10

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_B0_00 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_B0_00 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_B0_00 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_B0_00 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_B0_00 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_B0_00 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_B0_00 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_00 field descriptions (continued) pin 10

Field	Description
	011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_B0_00 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.200 SW_PAD_CTL_PAD_GPIO_B0_01 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_01) pin 12

SW_PAD_CTL Register

Address: 401F_8000h base + 330h offset = 401F_8330h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_01 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_B0_01 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_B0_01 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_01 field descriptions (continued) pin 12

Field	Description
	10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_B0_01 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_B0_01 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_B0_01 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_B0_01 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_B0_01 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_B0_01 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.201 SW_PAD_CTL_PAD_GPIO_B0_02 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_02) pin 11

SW_PAD_CTL Register

Address: 401F_8000h base + 334h offset = 401F_8334h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_02 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_B0_02 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_B0_02 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_B0_02 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_B0_02 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_B0_02

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_02 field descriptions (continued) pin 11

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_B0_02 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_B0_02 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_B0_02 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.202 SW_PAD_CTL_PAD_GPIO_B0_03 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_03) pin 13

SW_PAD_CTL Register

Address: 401F_8000h base + 338h offset = 401F_8338h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_03 field descriptions pin 13

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_B0_03 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_B0_03 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_B0_03 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_B0_03 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_B0_03 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_B0_03 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_B0_03 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_03 field descriptions (continued) pin 13

Field	Description
	011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_B0_03 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.203 SW_PAD_CTL_PAD_GPIO_B0_04 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_04)

pin 40 (MM)

SW_PAD_CTL Register

Address: 401F_8000h base + 33Ch offset = 401F_833Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_04 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_B0_04 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_B0_04 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_04 field descriptions (continued) pin 40 (MM)

Field	Description
	10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_B0_04 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_B0_04 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_B0_04 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_B0_04 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_B0_04 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_B0_04 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.204 SW_PAD_CTL_PAD_GPIO_B0_05 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_05)

pin 41 (MM)

SW_PAD_CTL Register

Address: 401F_8000h base + 340h offset = 401F_8340h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_05 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_B0_05 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_B0_05 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_B0_05 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_B0_05 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_B0_05

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_05 field descriptions (continued) pin 41 (MM)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_B0_05 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_B0_05 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_B0_05 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.205 SW_PAD_CTL_PAD_GPIO_B0_06 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_06)

pin 42 (MM)

SW_PAD_CTL Register

Address: 401F_8000h base + 344h offset = 401F_8344h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_06 field descriptions pin 42 (MM)

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_B0_06 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_B0_06 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_B0_06 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_B0_06 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_B0_06 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_B0_06 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_B0_06 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_06 field descriptions (continued) pin 42 (MM)

Field	Description
	011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_B0_06 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.206 SW_PAD_CTL_PAD_GPIO_B0_07 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_07)

pin 43 (MM)

SW_PAD_CTL Register

Address: 401F_8000h base + 348h offset = 401F_8348h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_07 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_B0_07 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_B0_07 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_07 field descriptions (continued) pin 43 (MM)

Field	Description
	10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_B0_07 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_B0_07 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_B0_07 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_B0_07 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_B0_07 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_B0_07 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.207 SW_PAD_CTL_PAD_GPIO_B0_08 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_08)

pin 44 (MM)

SW_PAD_CTL Register

Address: 401F_8000h base + 34Ch offset = 401F_834Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_08 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_B0_08 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_B0_08 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_B0_08 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_B0_08 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_B0_08

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_08 field descriptions (continued) pin 44 (MM)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_B0_08 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_B0_08 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_B0_08 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.208 SW_PAD_CTL_PAD_GPIO_B0_09 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_09)

pin 45 (MM)

SW_PAD_CTL Register

Address: 401F_8000h base + 350h offset = 401F_8350h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_09 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_B0_09 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_B0_09 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_B0_09 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_B0_09 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_B0_09 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_B0_09 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_B0_09 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_09 field descriptions (continued) pin 45 (MM)

Field	Description
	011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_B0_09 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.209 SW_PAD_CTL_PAD_GPIO_B0_10 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_10) pin 6

SW_PAD_CTL Register

Address: 401F_8000h base + 354h offset = 401F_8354h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_10 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_B0_10 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_B0_10 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_10 field descriptions (continued) pin 6

Field	Description
	10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_B0_10 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_B0_10 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_B0_10 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_B0_10 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_B0_10 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_B0_10 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.210 SW_PAD_CTL_PAD_GPIO_B0_11 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_11) pin 9

SW_PAD_CTL Register

Address: 401F_8000h base + 358h offset = 401F_8358h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_11 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_B0_11 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_B0_11 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_B0_11 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_B0_11 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_B0_11

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_11 field descriptions (continued) pin 9

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_B0_11 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_B0_11 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_B0_11 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.211 SW_PAD_CTL_PAD_GPIO_B0_12 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_12) pin 32

SW_PAD_CTL Register

Address: 401F_8000h base + 35Ch offset = 401F_835Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_12 field descriptions pin 32

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_B0_12 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_B0_12 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_B0_12 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_B0_12 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_B0_12 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_B0_12 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_B0_12 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_12 field descriptions (continued) pin 32

Field	Description
	011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_B0_12 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.212 SW_PAD_CTL_PAD_GPIO_B0_13 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_13)

SW_PAD_CTL Register

Address: 401F_8000h base + 360h offset = 401F_8360h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_13 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_B0_13 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_B0_13 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_13 field descriptions (continued)

Field	Description
	10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_B0_13 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_B0_13 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_B0_13 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_B0_13 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_B0_13 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_B0_13 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.213 SW_PAD_CTL_PAD_GPIO_B0_14 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_14)

SW_PAD_CTL Register

Address: 401F_8000h base + 364h offset = 401F_8364h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_14 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_B0_14 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_B0_14 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_B0_14 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_B0_14 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_B0_14

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_14 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_B0_14 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_B0_14 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_B0_14 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.214 SW_PAD_CTL_PAD_GPIO_B0_15 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_15)

SW_PAD_CTL Register

Address: 401F_8000h base + 368h offset = 401F_8368h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_15 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_B0_15 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_B0_15 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_B0_15 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_B0_15 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_B0_15 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_B0_15 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_B0_15 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_B0_15 field descriptions (continued)

Field	Description
	011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_B0_15 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.215 SW_PAD_CTL_PAD_GPIO_B1_00 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_00)

pin 8

SW_PAD_CTL Register

Address: 401F_8000h base + 36Ch offset = 401F_836Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_00 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_B1_00 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_B1_00 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_00 field descriptions (continued) pin 8

Field	Description
	10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_B1_00 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_B1_00 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_B1_00 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_B1_00 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_B1_00 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_B1_00 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.216 SW_PAD_CTL_PAD_GPIO_B1_01 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_01) pin 7

SW_PAD_CTL Register

Address: 401F_8000h base + 370h offset = 401F_8370h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_01 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_B1_01 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_B1_01 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_B1_01 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_B1_01 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_B1_01

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_01 field descriptions (continued) pin 7

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_B1_01 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_B1_01 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_B1_01 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.217 SW_PAD_CTL_PAD_GPIO_B1_02 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_02)

pin 36 (T4.1)

SW_PAD_CTL Register

Address: 401F_8000h base + 374h offset = 401F_8374h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_02 field descriptions pin 36 (T4.1)

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_B1_02 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_B1_02 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_B1_02 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_B1_02 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_B1_02 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_B1_02 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_B1_02 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_02 field descriptions (continued) pin 36 (T4.1)

Field	Description
	011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_B1_02 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.218 SW_PAD_CTL_PAD_GPIO_B1_03 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_03)

pin 37 (T4.1)

SW_PAD_CTL Register

Address: 401F_8000h base + 378h offset = 401F_8378h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_03 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_B1_03 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_B1_03 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_03 field descriptions (continued) pin 37 (T4.1)

Field	Description
	10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_B1_03 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_B1_03 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_B1_03 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_B1_03 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_B1_03 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_B1_03 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.219 SW_PAD_CTL_PAD_GPIO_B1_04 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_04)

SW_PAD_CTL Register

Address: 401F_8000h base + 37Ch offset = 401F_837Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_04 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_B1_04 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_B1_04 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_B1_04 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_B1_04 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_B1_04

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_04 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_B1_04 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_B1_04 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_B1_04 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.220 SW_PAD_CTL_PAD_GPIO_B1_05 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_05)

SW_PAD_CTL Register

Address: 401F_8000h base + 380h offset = 401F_8380h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_05 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_B1_05 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_B1_05 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_B1_05 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_B1_05 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_B1_05 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_B1_05 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_B1_05 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_05 field descriptions (continued)

Field	Description
	011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_B1_05 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.221 SW_PAD_CTL_PAD_GPIO_B1_06 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_06)

SW_PAD_CTL Register

Address: 401F_8000h base + 384h offset = 401F_8384h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_06 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_B1_06 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_B1_06 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_06 field descriptions (continued)

Field	Description
	10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_B1_06 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_B1_06 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_B1_06 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_B1_06 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_B1_06 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_B1_06 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.222 SW_PAD_CTL_PAD_GPIO_B1_07 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_07)

SW_PAD_CTL Register

Address: 401F_8000h base + 388h offset = 401F_8388h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_07 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_B1_07 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_B1_07 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_B1_07 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_B1_07 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_B1_07

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_07 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_B1_07 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_B1_07 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_B1_07 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.223 SW_PAD_CTL_PAD_GPIO_B1_08 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_08)

SW_PAD_CTL Register

Address: 401F_8000h base + 38Ch offset = 401F_838Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_08 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_B1_08 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_B1_08 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_B1_08 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_B1_08 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_B1_08 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_B1_08 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_B1_08 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_08 field descriptions (continued)

Field	Description
	011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_B1_08 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.224 SW_PAD_CTL_PAD_GPIO_B1_09 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_09)

SW_PAD_CTL Register

Address: 401F_8000h base + 390h offset = 401F_8390h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_09 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_B1_09 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_B1_09 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_09 field descriptions (continued)

Field	Description
	10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_B1_09 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_B1_09 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_B1_09 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_B1_09 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_B1_09 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_B1_09 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.225 SW_PAD_CTL_PAD_GPIO_B1_10 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_10)

SW_PAD_CTL Register

Address: 401F_8000h base + 394h offset = 401F_8394h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_10 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_B1_10 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_B1_10 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_B1_10 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_B1_10 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_B1_10

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_10 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_B1_10 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_B1_10 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_B1_10 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.226 SW_PAD_CTL_PAD_GPIO_B1_11 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_11)

SW_PAD_CTL Register

Address: 401F_8000h base + 398h offset = 401F_8398h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_11 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_B1_11 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_B1_11 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_B1_11 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_B1_11 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_B1_11 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_B1_11 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_B1_11 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_11 field descriptions (continued)

Field	Description
	011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_B1_11 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.227 SW_PAD_CTL_PAD_GPIO_B1_12 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_12)

SW_PAD_CTL Register

pin 35 (T4.1)

Address: 401F_8000h base + 39Ch offset = 401F_839Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_12 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_B1_12 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_B1_12 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_12 field descriptions (continued) pin 35 (T4.1)

Field	Description
	10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_B1_12 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_B1_12 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_B1_12 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_B1_12 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_B1_12 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_B1_12 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.228 SW_PAD_CTL_PAD_GPIO_B1_13 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_13)

pin 34 (T4.1)

SW_PAD_CTL Register

Address: 401F_8000h base + 3A0h offset = 401F_83A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_13 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_B1_13 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_B1_13 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_B1_13 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_B1_13 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_B1_13

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_13 field descriptions (continued) pin 34 (T4.1)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_B1_13 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_B1_13 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_B1_13 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.229 SW_PAD_CTL_PAD_GPIO_B1_14 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_14)

SW_PAD_CTL Register

Address: 401F_8000h base + 3A4h offset = 401F_83A4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_14 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_B1_14 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_B1_14 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_B1_14 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_B1_14 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_B1_14 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_B1_14 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_B1_14 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_14 field descriptions (continued)

Field	Description
	011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_B1_14 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.230 SW_PAD_CTL_PAD_GPIO_B1_15 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_15)

SW_PAD_CTL Register

Address: 401F_8000h base + 3A8h offset = 401F_83A8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_15 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_B1_15 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_B1_15 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_B1_15 field descriptions (continued)

Field	Description
	10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_B1_15 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_B1_15 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_B1_15 0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_B1_15 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_B1_15 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_B1_15 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.231 SW_PAD_CTL_PAD_GPIO_SD_B0_00 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B0_00)

SW_PAD_CTL Register

Address: 401F_8000h base + 3ACh offset = 401F_83ACh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B0_00 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_SD_B0_00 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_SD_B0_00 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_SD_B0_00 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_SD_B0_00 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_SD_B0_00

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B0_00 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_SD_B0_00 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_SD_B0_00 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_SD_B0_00 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.232 SW_PAD_CTL_PAD_GPIO_SD_B0_01 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B0_01)

SW_PAD_CTL Register

Address: 401F_8000h base + 3B0h offset = 401F_83B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B0_01 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_SD_B0_01 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_SD_B0_01 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_SD_B0_01 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_SD_B0_01 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_SD_B0_01

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B0_01 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_SD_B0_01 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_SD_B0_01 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_SD_B0_01 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.233 SW_PAD_CTL_PAD_GPIO_SD_B0_02 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B0_02)

SW_PAD_CTL Register

Address: 401F_8000h base + 3B4h offset = 401F_83B4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B0_02 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_SD_B0_02 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_SD_B0_02 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_SD_B0_02 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_SD_B0_02 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_SD_B0_02

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B0_02 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_SD_B0_02 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_SD_B0_02 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_SD_B0_02 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.234 SW_PAD_CTL_PAD_GPIO_SD_B0_03 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B0_03)

SW_PAD_CTL Register

Address: 401F_8000h base + 3B8h offset = 401F_83B8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B0_03 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_SD_B0_03 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_SD_B0_03 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_SD_B0_03 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_SD_B0_03 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_SD_B0_03

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B0_03 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_SD_B0_03 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_SD_B0_03 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_SD_B0_03 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.235 SW_PAD_CTL_PAD_GPIO_SD_B0_04 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B0_04)

SW_PAD_CTL Register

Address: 401F_8000h base + 3BCh offset = 401F_83BCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B0_04 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_SD_B0_04 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_SD_B0_04 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_SD_B0_04 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_SD_B0_04 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_SD_B0_04

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B0_04 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_SD_B0_04 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_SD_B0_04 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_SD_B0_04 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.236 SW_PAD_CTL_PAD_GPIO_SD_B0_05 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B0_05)

SW_PAD_CTL Register

Address: 401F_8000h base + 3C0h offset = 401F_83C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B0_05 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_SD_B0_05 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_SD_B0_05 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_SD_B0_05 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_SD_B0_05 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_SD_B0_05

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B0_05 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_SD_B0_05 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_SD_B0_05 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_SD_B0_05 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.237 SW_PAD_CTL_PAD_GPIO_SD_B1_00 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B1_00)

SW_PAD_CTL Register

Address: 401F_8000h base + 3C4h offset = 401F_83C4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B1_00 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_SD_B1_00 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_SD_B1_00 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_SD_B1_00 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_SD_B1_00 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_SD_B1_00

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B1_00 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_SD_B1_00 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_SD_B1_00 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_SD_B1_00 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.238 SW_PAD_CTL_PAD_GPIO_SD_B1_01 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B1_01)

SW_PAD_CTL Register

Address: 401F_8000h base + 3C8h offset = 401F_83C8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B1_01 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_SD_B1_01 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_SD_B1_01 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_SD_B1_01 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_SD_B1_01 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_SD_B1_01

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B1_01 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_SD_B1_01 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_SD_B1_01 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_SD_B1_01 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.239 SW_PAD_CTL_PAD_GPIO_SD_B1_02 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B1_02)

SW_PAD_CTL Register

Address: 401F_8000h base + 3CCh offset = 401F_83CCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B1_02 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_SD_B1_02 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_SD_B1_02 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_SD_B1_02 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_SD_B1_02 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_SD_B1_02

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B1_02 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_SD_B1_02 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_SD_B1_02 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_SD_B1_02 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.240 SW_PAD_CTL_PAD_GPIO_SD_B1_03 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B1_03)

SW_PAD_CTL Register

Address: 401F_8000h base + 3D0h offset = 401F_83D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B1_03 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_SD_B1_03 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_SD_B1_03 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_SD_B1_03 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_SD_B1_03 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_SD_B1_03

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B1_03 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_SD_B1_03 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_SD_B1_03 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_SD_B1_03 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.241 SW_PAD_CTL_PAD_GPIO_SD_B1_04 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B1_04)

SW_PAD_CTL Register

Address: 401F_8000h base + 3D4h offset = 401F_83D4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B1_04 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_SD_B1_04 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_SD_B1_04 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_SD_B1_04 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_SD_B1_04 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_SD_B1_04

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B1_04 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_SD_B1_04 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_SD_B1_04 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_SD_B1_04 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.242 SW_PAD_CTL_PAD_GPIO_SD_B1_05 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B1_05)

SW_PAD_CTL Register

Address: 401F_8000h base + 3D8h offset = 401F_83D8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B1_05 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_SD_B1_05 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_SD_B1_05 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_SD_B1_05 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_SD_B1_05 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_SD_B1_05

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B1_05 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_SD_B1_05 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_SD_B1_05 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_SD_B1_05 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.243 SW_PAD_CTL_PAD_GPIO_SD_B1_06 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B1_06)

SW_PAD_CTL Register

Address: 401F_8000h base + 3DCh offset = 401F_83DCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B1_06 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_SD_B1_06 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_SD_B1_06 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_SD_B1_06 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_SD_B1_06 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_SD_B1_06

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B1_06 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_SD_B1_06 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_SD_B1_06 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_SD_B1_06 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.244 SW_PAD_CTL_PAD_GPIO_SD_B1_07 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B1_07)

SW_PAD_CTL Register

Address: 401F_8000h base + 3E0h offset = 401F_83E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B1_07 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_SD_B1_07 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_SD_B1_07 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_SD_B1_07 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_SD_B1_07 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_SD_B1_07

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B1_07 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_SD_B1_07 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_SD_B1_07 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_SD_B1_07 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.245 SW_PAD_CTL_PAD_GPIO_SD_B1_08 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B1_08)

SW_PAD_CTL Register

Address: 401F_8000h base + 3E4h offset = 401F_83E4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B1_08 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_SD_B1_08 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_SD_B1_08 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_SD_B1_08 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_SD_B1_08 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_SD_B1_08

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B1_08 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_SD_B1_08 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_SD_B1_08 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_SD_B1_08 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.246 SW_PAD_CTL_PAD_GPIO_SD_B1_09 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B1_09)

SW_PAD_CTL Register

Address: 401F_8000h base + 3E8h offset = 401F_83E8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B1_09 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_SD_B1_09 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_SD_B1_09 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_SD_B1_09 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_SD_B1_09 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_SD_B1_09

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B1_09 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_SD_B1_09 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_SD_B1_09 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_SD_B1_09 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.247 SW_PAD_CTL_PAD_GPIO_SD_B1_10 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B1_10)

SW_PAD_CTL Register

Address: 401F_8000h base + 3ECh offset = 401F_83ECh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B1_10 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_SD_B1_10 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_SD_B1_10 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_SD_B1_10 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_SD_B1_10 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_SD_B1_10

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B1_10 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_SD_B1_10 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_SD_B1_10 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_SD_B1_10 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.248 SW_PAD_CTL_PAD_GPIO_SD_B1_11 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B1_11)

SW_PAD_CTL Register

Address: 401F_8000h base + 3F0h offset = 401F_83F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PUS		PUE	PKE	ODE	Reserved			SPEED		DSE			Reserved		SRE
Reset	0	0	0	1	0	0	0	0	1	0	1	1	0	0	0	0

IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B1_11 field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: GPIO_SD_B1_11 0 **HYS_0_Hysteresis_Disabled** — Hysteresis Disabled 1 **HYS_1_Hysteresis_Enabled** — Hysteresis Enabled
15–14 PUS	Pull Up / Down Config. Field Select one out of next values for pad: GPIO_SD_B1_11 00 **PUS_0_100K_Ohm_Pull_Down** — 100K Ohm Pull Down 01 **PUS_1_47K_Ohm_Pull_Up** — 47K Ohm Pull Up 10 **PUS_2_100K_Ohm_Pull_Up** — 100K Ohm Pull Up 11 **PUS_3_22K_Ohm_Pull_Up** — 22K Ohm Pull Up
13 PUE	Pull / Keep Select Field Select one out of next values for pad: GPIO_SD_B1_11 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: GPIO_SD_B1_11 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled
11 ODE	Open Drain Enable Field Select one out of next values for pad: GPIO_SD_B1_11

Table continues on the next page...

IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_B1_11 field descriptions (continued)

Field	Description
	0 **ODE_0_Open_Drain_Disabled** — Open Drain Disabled 1 **ODE_1_Open_Drain_Enabled** — Open Drain Enabled
10–8 -	This field is reserved. Reserved
7–6 SPEED	Speed Field Select one out of next values for pad: GPIO_SD_B1_11 00 **SPEED_0_low_50MHz** — low(50MHz) 01 **SPEED_1_medium_100MHz** — medium(100MHz) 10 **SPEED_2_fast_150MHz** — fast(150MHz) 11 **SPEED_3_max_200MHz** — max(200MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: GPIO_SD_B1_11 000 **DSE_0_output_driver_disabled** — output driver disabled; 001 **DSE_1_R0_150_Ohm_3_3V_260_Ohm_1_8V** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V) 010 **DSE_2_R0_2** — R0/2 011 **DSE_3_R0_3** — R0/3 100 **DSE_4_R0_4** — R0/4 101 **DSE_5_R0_5** — R0/5 110 **DSE_6_R0_6** — R0/6 111 **DSE_7_R0_7** — R0/7
2–1 -	This field is reserved. Reserved
0 SRE	Slew Rate Field Select one out of next values for pad: GPIO_SD_B1_11 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6.249 ANATOP_USB_OTG1_ID_SELECT_INPUT DAISY Register (IOMUXC_ANATOP_USB_OTG1_ID_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 3F4h offset = 401F_83F4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_ANATOP_USB_OTG1_ID_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: anatop, In Pin: usb_otg_id 0 **GPIO_AD_B0_01_ALT3** — Selecting Pad: GPIO_AD_B0_01 for Mode: ALT3 1 **GPIO_AD_B1_02_ALT0** — Selecting Pad: GPIO_AD_B1_02 for Mode: ALT0 pin 14

11.6.250 ANATOP_USB_OTG2_ID_SELECT_INPUT DAISY Register (IOMUXC_ANATOP_USB_OTG2_ID_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 3F8h offset = 401F_83F8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_ANATOP_USB_OTG2_ID_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: anatop, In Pin: usb 0 **GPIO_AD_B0_00_ALT3** — Selecting Pad: GPIO_AD_B0_00 for Mode: ALT3 1 **GPIO_AD_B1_00_ALT0** — Selecting Pad: GPIO_AD_B1_00 for Mode: ALT0 pin 19

11.6.251 CCM_PMIC_READY_SELECT_INPUT DAISY Register (IOMUXC_CCM_PMIC_READY_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 3FCh offset = 401F_83FCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_CCM_PMIC_READY_SELECT_INPUT field descriptions

Field	Description
31–3 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: ccm, In Pin: pmic_vfuncional_ready 000 **GPIO_SD_B1_03_ALT6** — Selecting Pad: GPIO_SD_B1_03 for Mode: ALT6 001 **GPIO_AD_B0_12_ALT1** — Selecting Pad: GPIO_AD_B0_12 for Mode: ALT1 pin 24 010 **GPIO_AD_B1_01_ALT4** — Selecting Pad: GPIO_AD_B1_01 for Mode: ALT4 pin 18 011 **GPIO_AD_B1_08_ALT3** — Selecting Pad: GPIO_AD_B1_08 for Mode: ALT3 pin 22 100 **GPIO_EMC_32_ALT3** — Selecting Pad: GPIO_EMC_32 for Mode: ALT3 pin 28

11.6.252 CSI_DATA02_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA02_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 400h offset = 401F_8400h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

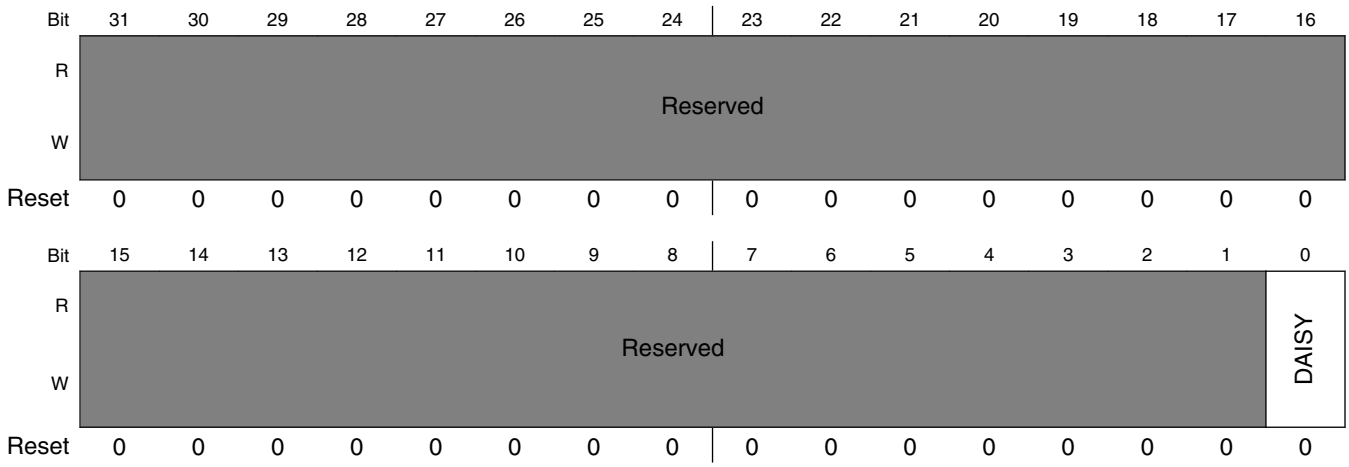
IOMUXC_CSI_DATA02_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: csi, In Pin: csi_d2 0 **GPIO_AD_B1_15_ALT4** — Selecting Pad: GPIO_AD_B1_15 for Mode: ALT4 pin 27 1 **GPIO_AD_B0_11_ALT4** — Selecting Pad: GPIO_AD_B0_11 for Mode: ALT4

11.6.253 CSI_DATA03_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA03_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 404h offset = 401F_8404h



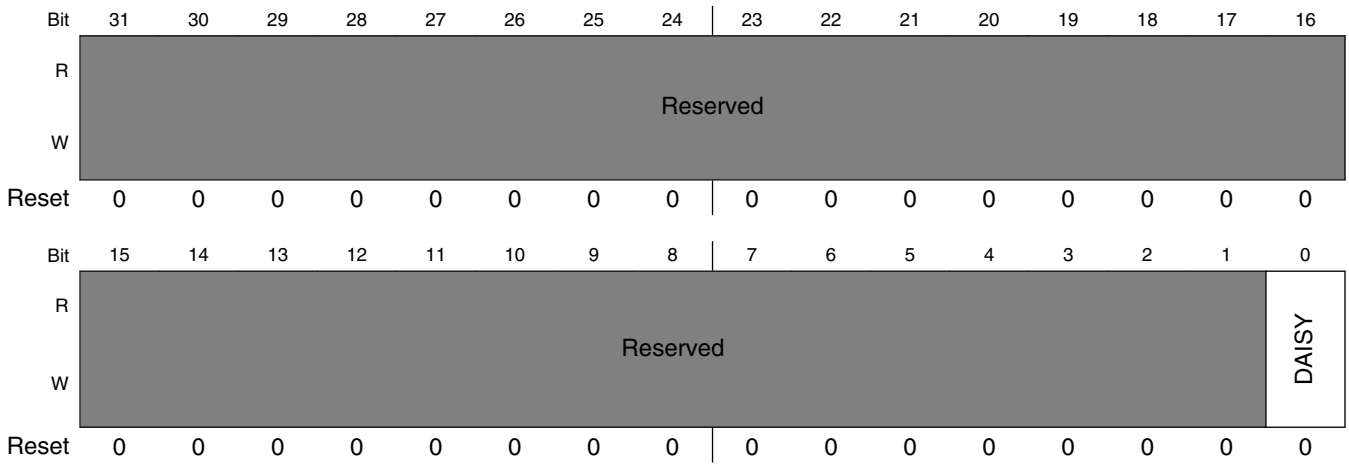
IOMUXC_CSI_DATA03_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: csi, In Pin: csi_d3 0 **GPIO_AD_B1_14_ALT4** — Selecting Pad: GPIO_AD_B1_14 for Mode: ALT4 pin 26 1 **GPIO_AD_B0_10_ALT4** — Selecting Pad: GPIO_AD_B0_10 for Mode: ALT4

11.6.254 CSI_DATA04_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA04_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 408h offset = 401F_8408h



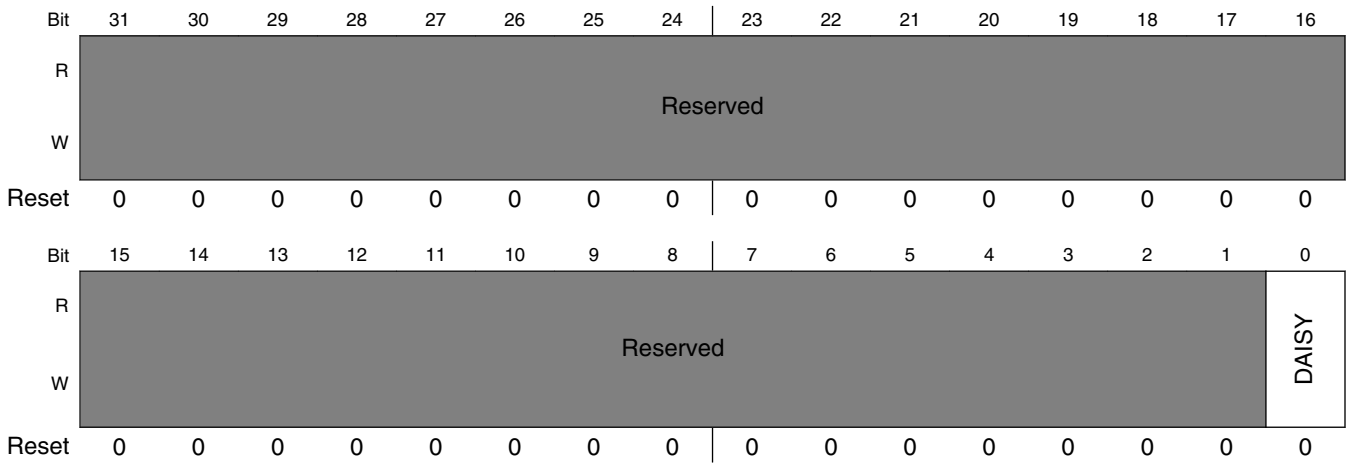
IOMUXC_CSI_DATA04_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: csi, In Pin: csi_d4 0 **GPIO_AD_B1_13_ALT4** — Selecting Pad: GPIO_AD_B1_13 for Mode: ALT4 pin 39 (T4.1) 1 **GPIO_AD_B0_09_ALT4** — Selecting Pad: GPIO_AD_B0_09 for Mode: ALT4

11.6.255 CSI_DATA05_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA05_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 40Ch offset = 401F_840Ch



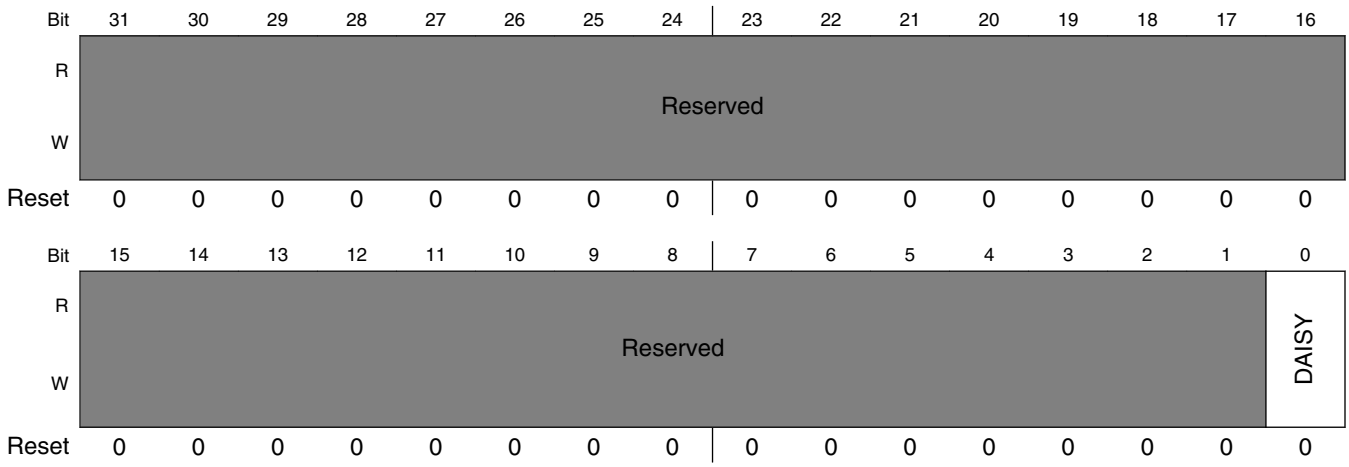
IOMUXC_CSI_DATA05_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: csi, In Pin: csi_d5 0 **GPIO_AD_B1_12_ALT4** — Selecting Pad: GPIO_AD_B1_12 for Mode: ALT4 pin 38 (T4.1) 1 **GPIO_AD_B0_08_ALT4** — Selecting Pad: GPIO_AD_B0_08 for Mode: ALT4

11.6.256 CSI_DATA06_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA06_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 410h offset = 401F_8410h



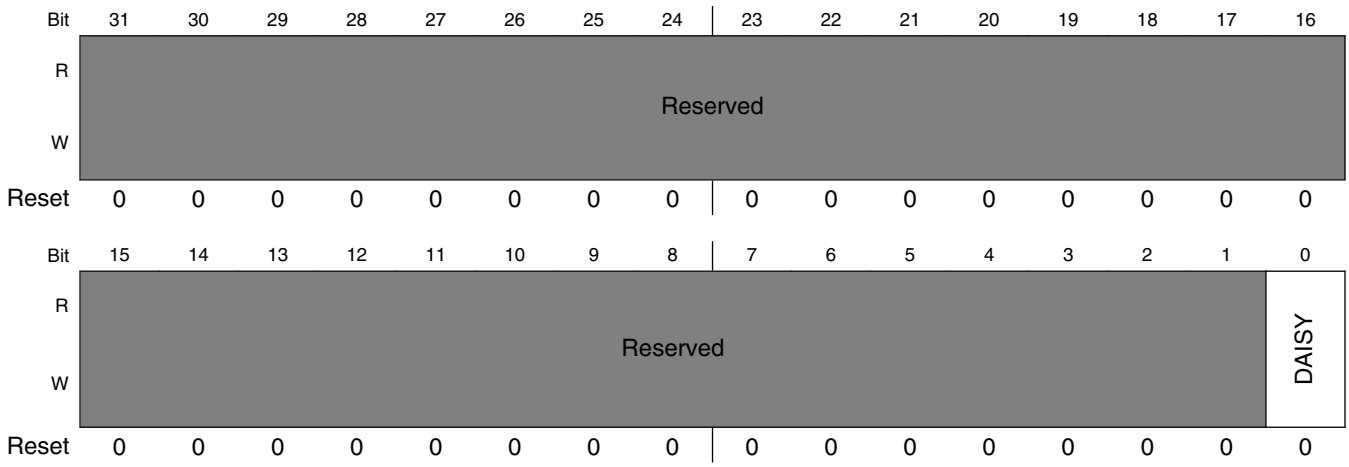
IOMUXC_CSI_DATA06_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: csi, In Pin: csi_d6 0 **GPIO_AD_B1_11_ALT4** — Selecting Pad: GPIO_AD_B1_11 for Mode: ALT4 pin 21 1 **GPIO_AD_B0_07_ALT4** — Selecting Pad: GPIO_AD_B0_07 for Mode: ALT4

11.6.257 CSI_DATA07_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA07_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 414h offset = 401F_8414h



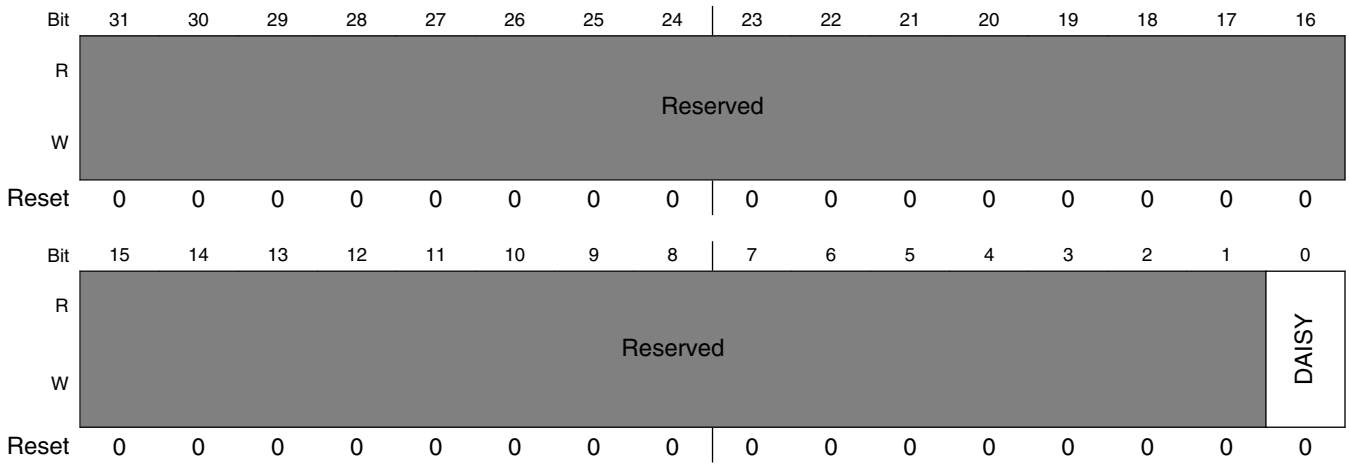
IOMUXC_CSI_DATA07_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: csi, In Pin: csi_d7 0 **GPIO_AD_B1_10_ALT4** — Selecting Pad: GPIO_AD_B1_10 for Mode: ALT4 pin 20 1 **GPIO_AD_B0_06_ALT4** — Selecting Pad: GPIO_AD_B0_06 for Mode: ALT4

11.6.258 CSI_DATA08_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA08_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 418h offset = 401F_8418h



IOMUXC_CSI_DATA08_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: csi, In Pin: csi_d8 0 **GPIO_AD_B1_09_ALT4** — Selecting Pad: GPIO_AD_B1_09 for Mode: ALT4 pin 23 1 **GPIO_AD_B0_05_ALT4** — Selecting Pad: GPIO_AD_B0_05 for Mode: ALT4

11.6.259 CSI_DATA09_SELECT_INPUT DAISY Register (IOMUXC_CSI_DATA09_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 41Ch offset = 401F_841Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_CSI_DATA09_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: csi, In Pin: csi_d9 0 **GPIO_AD_B1_08_ALT4** — Selecting Pad: GPIO_AD_B1_08 for Mode: ALT4 pin 22 1 **GPIO_AD_B0_04_ALT4** — Selecting Pad: GPIO_AD_B0_04 for Mode: ALT4

11.6.260 CSI_HSYNC_SELECT_INPUT DAISY Register (IOMUXC_CSI_HSYNC_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 420h offset = 401F_8420h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

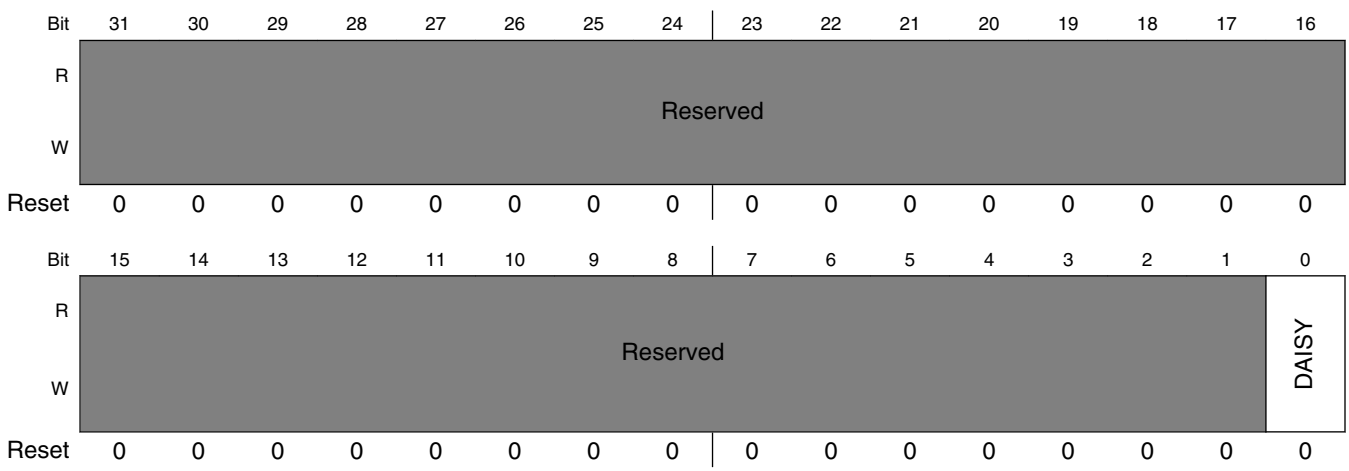
IOMUXC_CSI_HSYNC_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: csi, In Pin: csi_hsync 00 **GPIO_AD_B0_15_ALT4** — Selecting Pad: GPIO_AD_B0_15 for Mode: ALT4 01 **GPIO_AD_B1_07_ALT4** — Selecting Pad: GPIO_AD_B1_07 for Mode: ALT4 pin 16 10 **GPIO_B1_14_ALT2** — Selecting Pad: GPIO_B1_14 for Mode: ALT2

11.6.261 CSI_PIXCLK_SELECT_INPUT DAISY Register (IOMUXC_CSI_PIXCLK_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 424h offset = 401F_8424h



IOMUXC_CSI_PIXCLK_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: csi, In Pin: csi_pixclk 0 **GPIO_AD_B1_04_ALT4** — Selecting Pad: GPIO_AD_B1_04 for Mode: ALT4 pin 40 (T4.1) 1 **GPIO_B1_12_ALT2** — Selecting Pad: GPIO_B1_12 for Mode: ALT2 pin 35 (T4.1)

11.6.262 CSI_VSYNC_SELECT_INPUT DAISY Register (IOMUXC_CSI_VSYNC_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 428h offset = 401F_8428h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

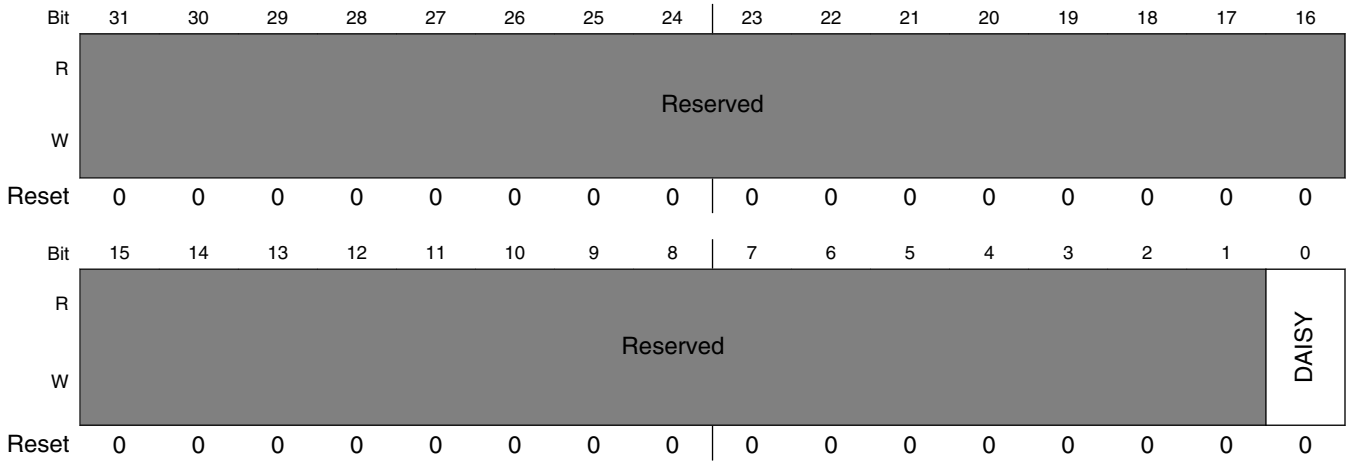
IOMUXC_CSI_VSYNC_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: csi, In Pin: csi_vsync 00 **GPIO_AD_B0_14_ALT4** — Selecting Pad: GPIO_AD_B0_14 for Mode: ALT4 01 **GPIO_AD_B1_06_ALT4** — Selecting Pad: GPIO_AD_B1_06 for Mode: ALT4 pin 17 10 **GPIO_B1_13_ALT2** — Selecting Pad: GPIO_B1_13 for Mode: ALT2 pin 34 (T4.1)

11.6.263 ENET_IPG_CLK_RMII_SELECT_INPUT DAISY Register (IOMUXC_ENET_IPG_CLK_RMII_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 42Ch offset = 401F_842Ch



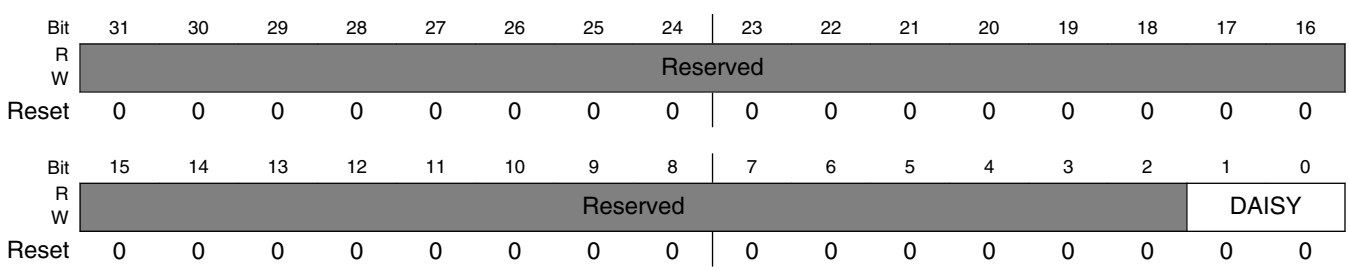
IOMUXC_ENET_IPG_CLK_RMII_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: enet, In Pin:rmii 0 **GPIO_EMC_25_ALT4** — Selecting Pad: GPIO_EMC_25 for Mode: ALT4 1 **GPIO_B1_10_ALT6** — Selecting Pad: GPIO_B1_10 for Mode: ALT6

11.6.264 ENET_MDIO_SELECT_INPUT DAISY Register (IOMUXC_ENET_MDIO_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 430h offset = 401F_8430h



IOMUXC_ENET_MDIO_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: enet, In Pin: mac0_mdio 00 **GPIO_AD_B1_05_ALT1** — Selecting Pad: GPIO_AD_B1_05 for Mode: ALT1 pin 41 (T4.1) 01 **GPIO_EMC_41_ALT4** — Selecting Pad: GPIO_EMC_41 for Mode: ALT4 10 **GPIO_B1_15_ALT0** — Selecting Pad: GPIO_B1_15 for Mode: ALT0

11.6.265 ENET0_RXDATA_SELECT_INPUT DAISY Register (IOMUXC_ENET0_RXDATA_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 434h offset = 401F_8434h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

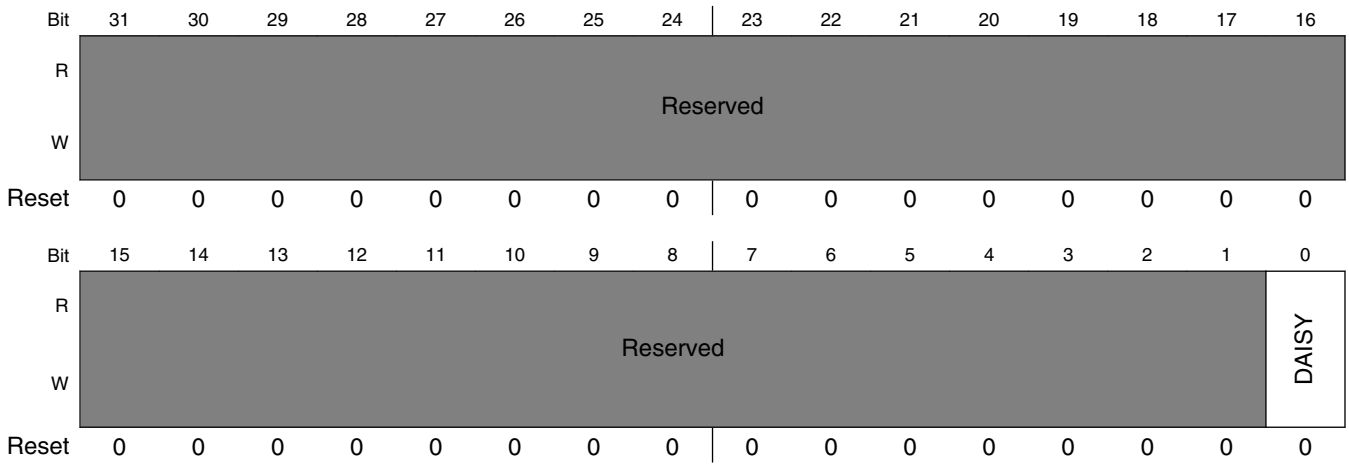
IOMUXC_ENET0_RXDATA_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: enet, In Pin: mac0_rxdata0 0 **GPIO_EMC_20_ALT3** — Selecting Pad: GPIO_EMC_20 for Mode: ALT3 1 **GPIO_B1_04_ALT3** — Selecting Pad: GPIO_B1_04 for Mode: ALT3

11.6.266 ENET1_RXDATA_SELECT_INPUT DAISY Register (IOMUXC_ENET1_RXDATA_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 438h offset = 401F_8438h



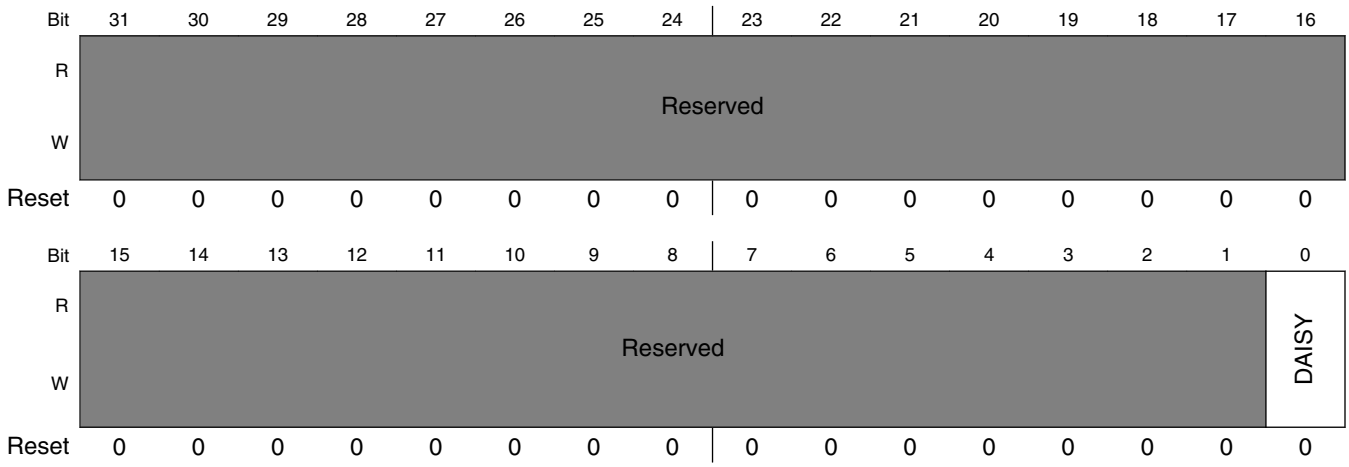
IOMUXC_ENET1_RXDATA_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: enet, In Pin: mac0_rxdata1 0 **GPIO_EMC_19_ALT3** — Selecting Pad: GPIO_EMC_19 for Mode: ALT3 1 **GPIO_B1_05_ALT3** — Selecting Pad: GPIO_B1_05 for Mode: ALT3

11.6.267 ENET_RXEN_SELECT_INPUT DAISY Register (IOMUXC_ENET_RXEN_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 43Ch offset = 401F_843Ch



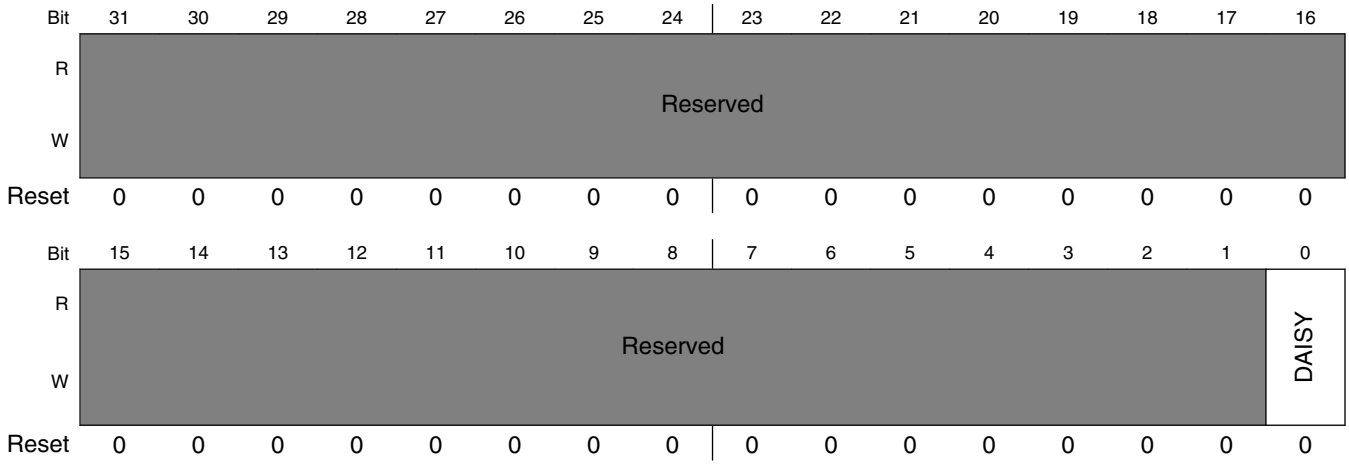
IOMUXC_ENET_RXEN_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: enet, In Pin: mac0_rxen 0 **GPIO_EMC_23_ALT3** — Selecting Pad: GPIO_EMC_23 for Mode: ALT3 1 **GPIO_B1_06_ALT3** — Selecting Pad: GPIO_B1_06 for Mode: ALT3

11.6.268 ENET_RXERR_SELECT_INPUT DAISY Register (IOMUXC_ENET_RXERR_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 440h offset = 401F_8440h



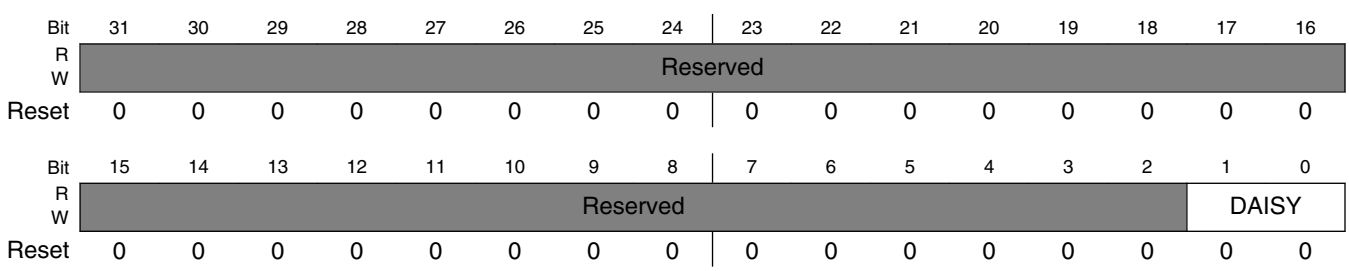
IOMUXC_ENET_RXERR_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: enet, In Pin: mac0_rxerr 0 **GPIO_EMC_26_ALT3** — Selecting Pad: GPIO_EMC_26 for Mode: ALT3 1 **GPIO_B1_11_ALT3** — Selecting Pad: GPIO_B1_11 for Mode: ALT3

11.6.269 ENET0_TIMER_SELECT_INPUT DAISY Register (IOMUXC_ENET0_TIMER_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 444h offset = 401F_8444h



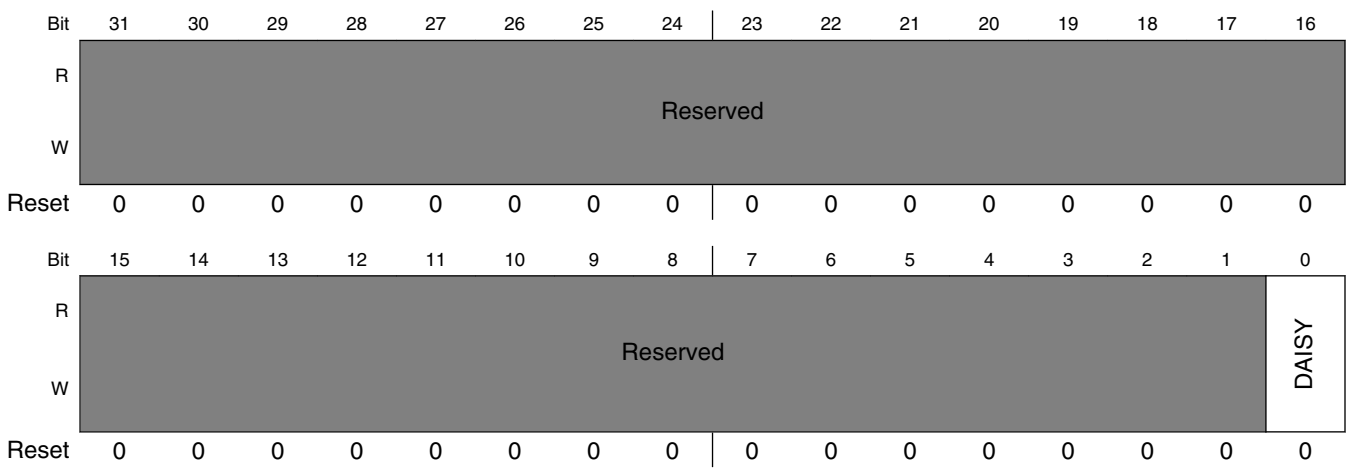
IOMUXC_ENET0_TIMER_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: enet, In Pin: mac0_timer0 00 **GPIO_AD_B0_15_ALT3** — Selecting Pad: GPIO_AD_B0_15 for Mode: ALT3 01 **GPIO_AD_B0_11_ALT7** — Selecting Pad: GPIO_AD_B0_11 for Mode: ALT7 10 **GPIO_B1_12_ALT3** — Selecting Pad: GPIO_B1_12 for Mode: ALT3 pin 35 (T4.1)

11.6.270 ENET_TXCLK_SELECT_INPUT DAISY Register (IOMUXC_ENET_TXCLK_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 448h offset = 401F_8448h



IOMUXC_ENET_TXCLK_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: enet, In Pin: mac0_txclk 0 **GPIO_EMC_25_ALT3** — Selecting Pad: GPIO_EMC_25 for Mode: ALT3 1 **GPIO_B1_10_ALT3** — Selecting Pad: GPIO_B1_10 for Mode: ALT3

11.6.271 FLEXCAN1_RX_SELECT_INPUT DAISY Register (IOMUXC_FLEXCAN1_RX_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 44Ch offset = 401F_844Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_FLEXCAN1_RX_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: flexcan1, In Pin: RX 00 **GPIO_SD_B1_03_ALT4** — Selecting Pad: GPIO_SD_B1_03 for Mode: ALT4 01 **GPIO_EMC_18_ALT3** — Selecting Pad: GPIO_EMC_18 for Mode: ALT3 10 **GPIO_AD_B1_09_ALT2** — Selecting Pad: GPIO_AD_B1_09 for Mode: ALT2 pin 23 11 **GPIO_B0_03_ALT2** — Selecting Pad: GPIO_B0_03 for Mode: ALT2 pin 13

11.6.272 FLEXCAN2_RX_SELECT_INPUT DAISY Register (IOMUXC_FLEXCAN2_RX_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 450h offset = 401F_8450h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_FLEXCAN2_RX_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: flexcan2, In Pin: RX 00 **GPIO_EMC_10_ALT3** — Selecting Pad: GPIO_EMC_10 for Mode: ALT3 01 **GPIO_AD_B0_03_ALT0** — Selecting Pad: GPIO_AD_B0_03 for Mode: ALT0 pin 10 **GPIO_AD_B0_15_ALT6** — Selecting Pad: GPIO_AD_B0_15 for Mode: ALT6 11 **GPIO_B1_09_ALT6** — Selecting Pad: GPIO_B1_09 for Mode: ALT6

11.6.273 FLEXPWM1_PWMA3_SELECT_INPUT DAISY Register (IOMUXC_FLEXPWM1_PWMA3_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 454h offset = 401F_8454h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																													DAISY		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

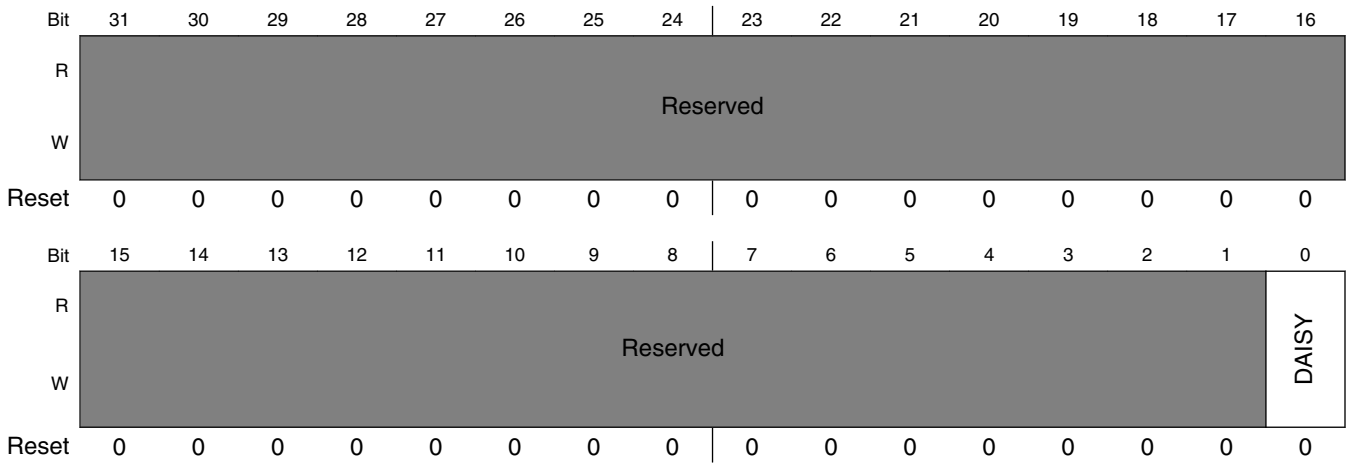
IOMUXC_FLEXPWM1_PWMA3_SELECT_INPUT field descriptions

Field	Description
31–3 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: flexpwm1, In Pin: pwma3 000 **GPIO_SD_B1_00_ALT2** — Selecting Pad: GPIO_SD_B1_00 for Mode: ALT2 001 **GPIO_EMC_12_ALT4** — Selecting Pad: GPIO_EMC_12 for Mode: ALT4 010 **GPIO_EMC_38_ALT1** — Selecting Pad: GPIO_EMC_38 for Mode: ALT1 011 **GPIO_AD_B0_10_ALT1** — Selecting Pad: GPIO_AD_B0_10 for Mode: ALT1 100 **GPIO_B1_00_ALT6** — Selecting Pad: GPIO_B1_00 for Mode: ALT6 pin 8

11.6.274 FLEXPWM1_PWMA0_SELECT_INPUT DAISY Register (IOMUXC_FLEXPWM1_PWMA0_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 458h offset = 401F_8458h



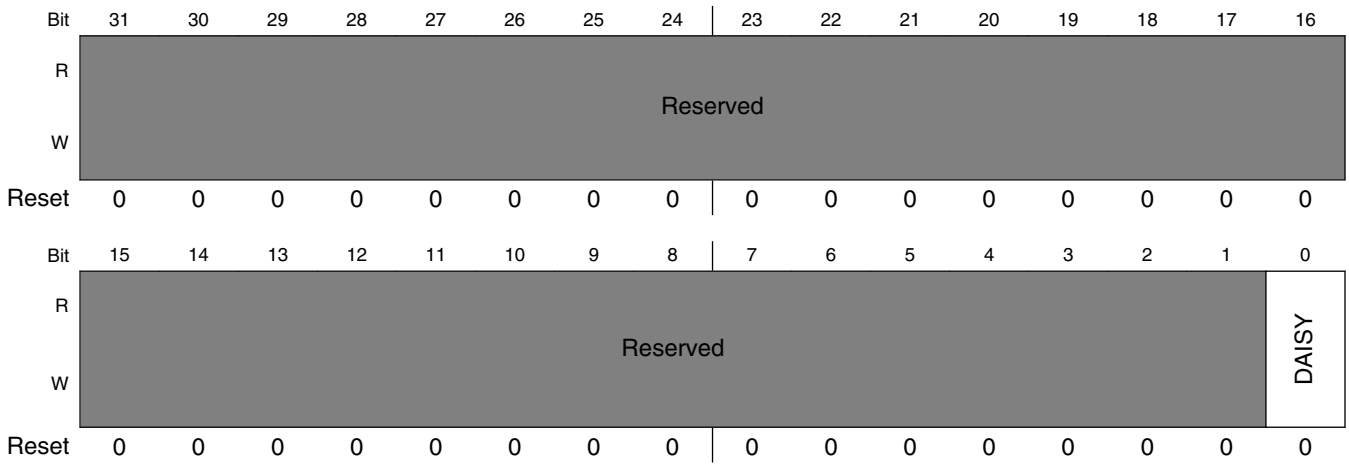
IOMUXC_FLEXPWM1_PWMA0_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: flexpwm1, In Pin: pwma0 0 **GPIO_EMC_23_ALT1** — Selecting Pad: GPIO_EMC_23 for Mode: ALT1 1 **GPIO_SD_B0_00_ALT1** — Selecting Pad: GPIO_SD_B0_00 for Mode: ALT1

11.6.275 FLEXPWM1_PWMA1_SELECT_INPUT DAISY Register (IOMUXC_FLEXPWM1_PWMA1_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 45Ch offset = 401F_845Ch



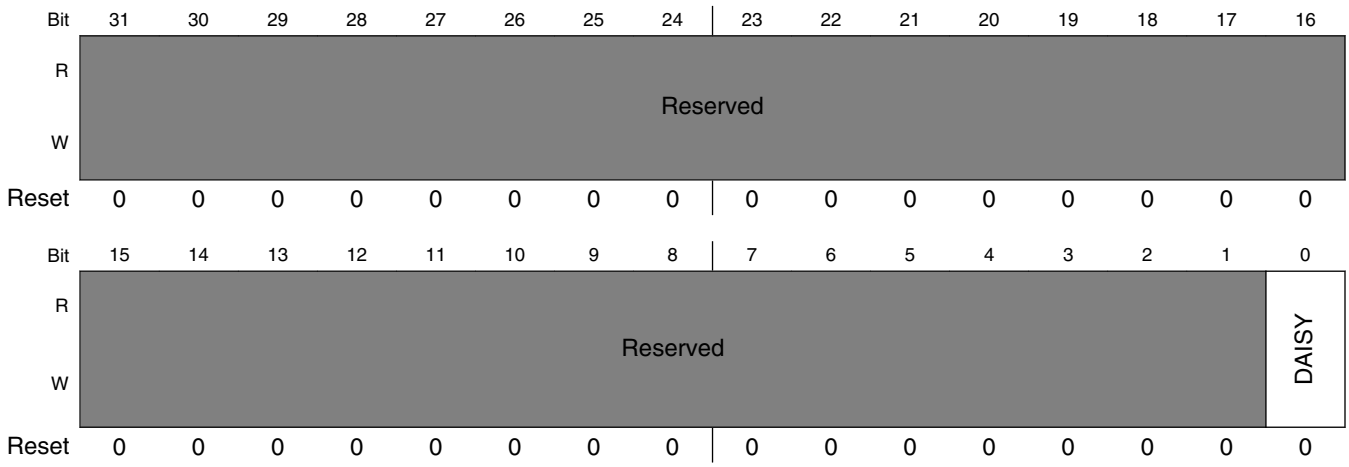
IOMUXC_FLEXPWM1_PWMA1_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: flexpwm1, In Pin: pwma1 0 **GPIO_EMC_25_ALT1** — Selecting Pad: GPIO_EMC_25 for Mode: ALT1 1 **GPIO_SD_B0_02_ALT1** — Selecting Pad: GPIO_SD_B0_02 for Mode: ALT1

11.6.276 FLEXPWM1_PWMA2_SELECT_INPUT DAISY Register (IOMUXC_FLEXPWM1_PWMA2_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 460h offset = 401F_8460h



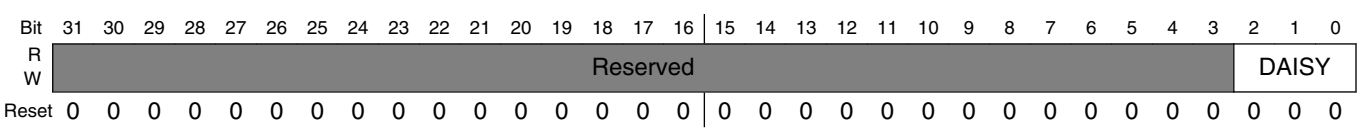
IOMUXC_FLEXPWM1_PWMA2_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: flexpwm1, In Pin: pwma2 0 **GPIO_EMC_27_ALT1** — Selecting Pad: GPIO_EMC_27 for Mode: ALT1 1 **GPIO_SD_B0_04_ALT1** — Selecting Pad: GPIO_SD_B0_04 for Mode: ALT1

11.6.277 FLEXPWM1_PWMB3_SELECT_INPUT DAISY Register (IOMUXC_FLEXPWM1_PWMB3_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 464h offset = 401F_8464h



IOMUXC_FLEXPWM1_PWMB3_SELECT_INPUT field descriptions

Field	Description
31–3 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: flexpwm1, In Pin: pwmb3 000 **GPIO_SD_B1_01_ALT2** — Selecting Pad: GPIO_SD_B1_01 for Mode: ALT2 001 **GPIO_EMC_13_ALT4** — Selecting Pad: GPIO_EMC_13 for Mode: ALT4 010 **GPIO_EMC_39_ALT1** — Selecting Pad: GPIO_EMC_39 for Mode: ALT1 011 **GPIO_AD_B0_11_ALT1** — Selecting Pad: GPIO_AD_B0_11 for Mode: ALT1 100 **GPIO_B1_01_ALT6** — Selecting Pad: GPIO_B1_01 for Mode: ALT6 pin 7

11.6.278 FLEXPWM1_PWMB0_SELECT_INPUT DAISY Register (IOMUXC_FLEXPWM1_PWMB0_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 468h offset = 401F_8468h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

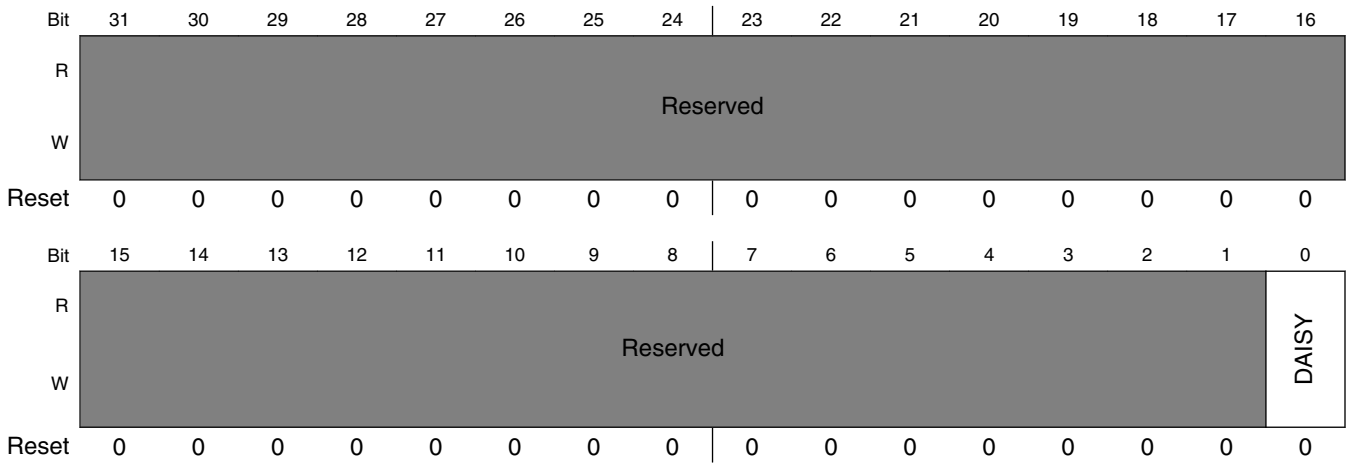
IOMUXC_FLEXPWM1_PWMB0_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: flexpwm1, In Pin: pwmb0 0 **GPIO_EMC_24_ALT1** — Selecting Pad: GPIO_EMC_24 for Mode: ALT1 1 **GPIO_SD_B0_01_ALT1** — Selecting Pad: GPIO_SD_B0_01 for Mode: ALT1

11.6.279 FLEXPWM1_PWMB1_SELECT_INPUT DAISY Register (IOMUXC_FLEXPWM1_PWMB1_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 46Ch offset = 401F_846Ch



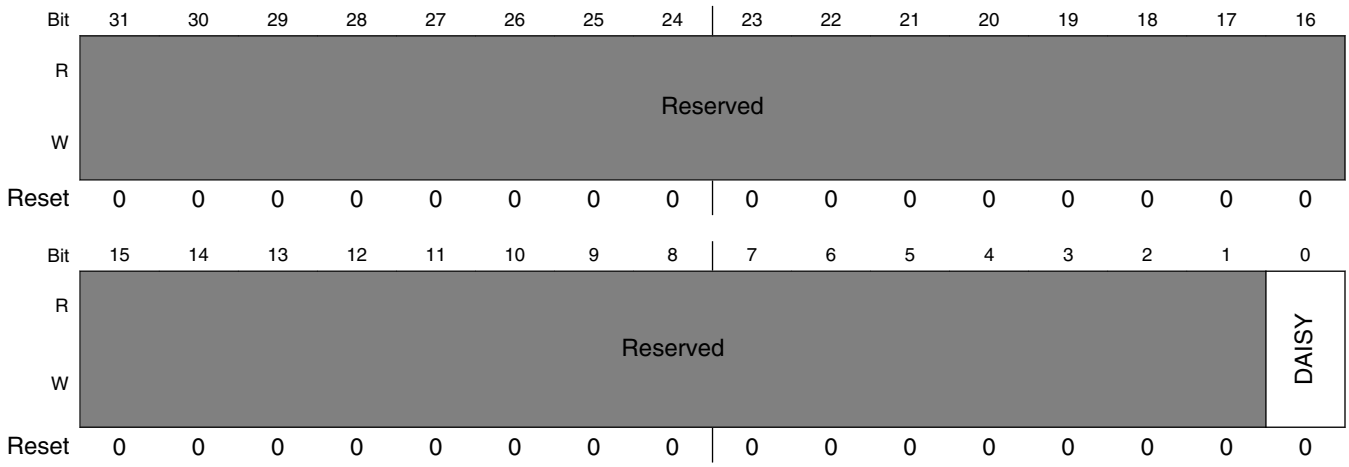
IOMUXC_FLEXPWM1_PWMB1_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: flexpwm1, In Pin: pwmb1 0 **GPIO_EMC_26_ALT1** — Selecting Pad: GPIO_EMC_26 for Mode: ALT1 1 **GPIO_SD_B0_03_ALT1** — Selecting Pad: GPIO_SD_B0_03 for Mode: ALT1

11.6.280 FLEXPWM1_PWMB2_SELECT_INPUT DAISY Register (IOMUXC_FLEXPWM1_PWMB2_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 470h offset = 401F_8470h



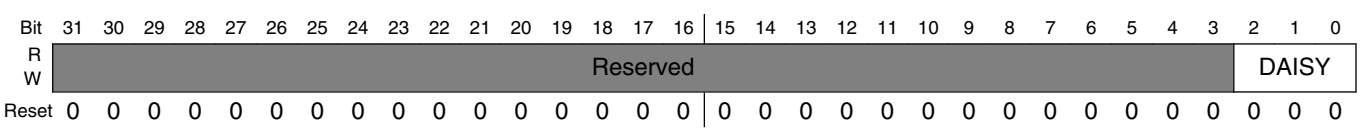
IOMUXC_FLEXPWM1_PWMB2_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: flexpwm1, In Pin: pwmb2 0 **GPIO_EMC_28_ALT1** — Selecting Pad: GPIO_EMC_28 for Mode: ALT1 1 **GPIO_SD_B0_05_ALT1** — Selecting Pad: GPIO_SD_B0_05 for Mode: ALT1

11.6.281 FLEXPWM2_PWMA3_SELECT_INPUT DAISY Register (IOMUXC_FLEXPWM2_PWMA3_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 474h offset = 401F_8474h



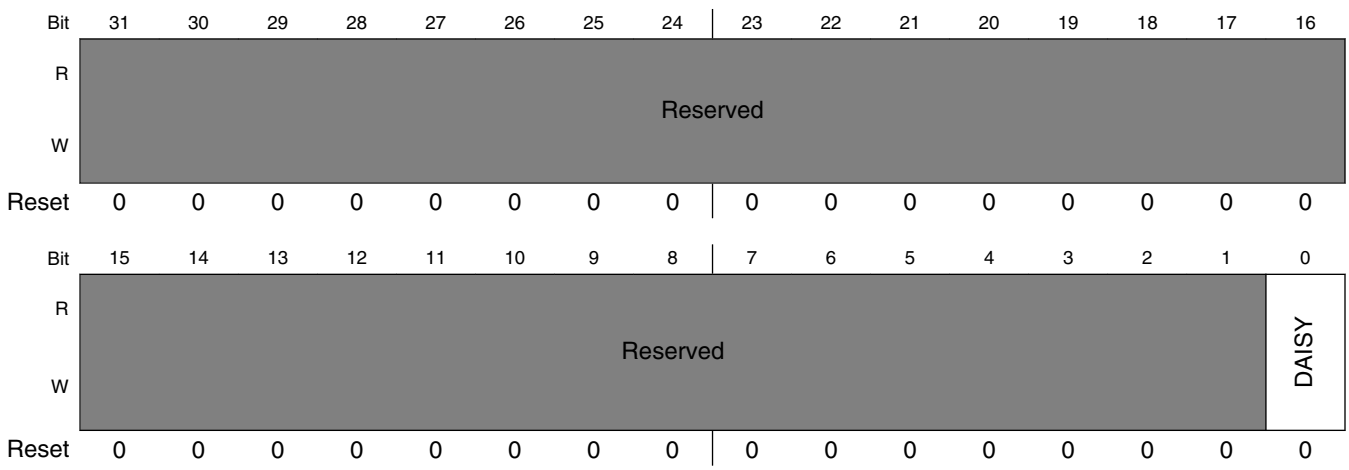
IOMUXC_FLEXPWM2_PWMA3_SELECT_INPUT field descriptions

Field	Description
31–3 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: flexpwm2, In Pin: pwma3 000 **GPIO_SD_B1_02_ALT2** — Selecting Pad: GPIO_SD_B1_02 for Mode: ALT2 001 **GPIO_EMC_19_ALT1** — Selecting Pad: GPIO_EMC_19 for Mode: ALT1 010 **GPIO_AD_B0_00_ALT0** — Selecting Pad: GPIO_AD_B0_00 for Mode: ALT0 011 **GPIO_AD_B0_09_ALT1** — Selecting Pad: GPIO_AD_B0_09 for Mode: ALT1 100 **GPIO_B1_02_ALT6** — Selecting Pad: GPIO_B1_02 for Mode: ALT6 pin 36 (T4.1)

11.6.282 FLEXPWM2_PWMA0_SELECT_INPUT DAISY Register (IOMUXC_FLEXPWM2_PWMA0_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 478h offset = 401F_8478h



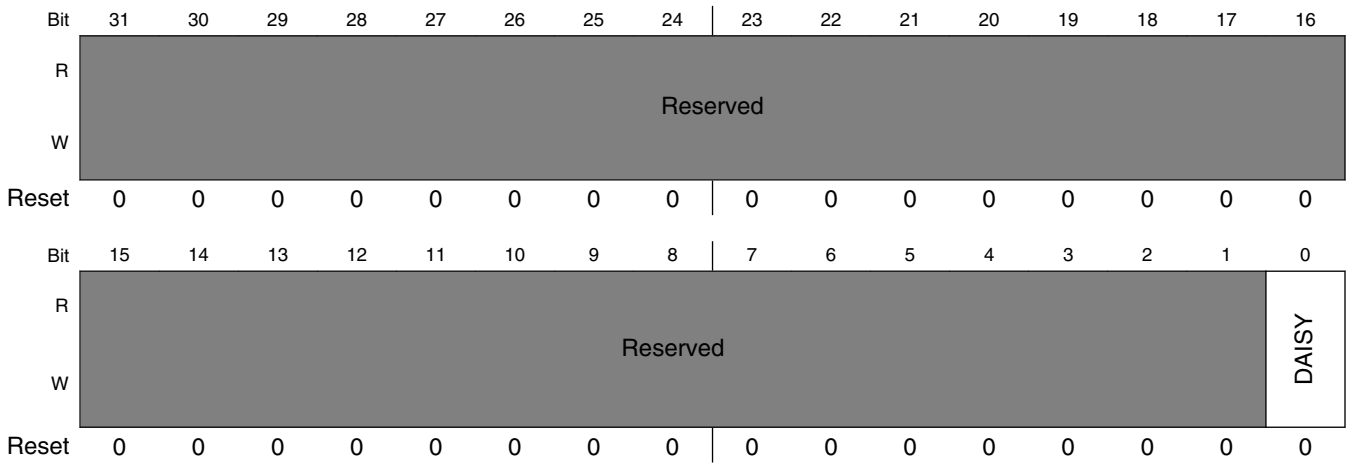
IOMUXC_FLEXPWM2_PWMA0_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: flexpwm2, In Pin: pwma0 0 **GPIO_EMC_06_ALT1** — Selecting Pad: GPIO_EMC_06 for Mode: ALT1 pin 4 1 **GPIO_B0_06_ALT2** — Selecting Pad: GPIO_B0_06 for Mode: ALT2 pin 42 (MM)

11.6.283 FLEXPWM2_PWMA1_SELECT_INPUT DAISY Register (IOMUXC_FLEXPWM2_PWMA1_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 47Ch offset = 401F_847Ch



IOMUXC_FLEXPWM2_PWMA1_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: flexpwm2, In Pin: pwma1 0 **GPIO_EMC_08_ALT1** — Selecting Pad: GPIO_EMC_08 for Mode: ALT1 pin 5 1 **GPIO_B0_08_ALT2** — Selecting Pad: GPIO_B0_08 for Mode: ALT2 pin 44 (MM)

11.6.284 FLEXPWM2_PWMA2_SELECT_INPUT DAISY Register (IOMUXC_FLEXPWM2_PWMA2_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 480h offset = 401F_8480h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_FLEXPWM2_PWMA2_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: flexpwm2, In Pin: pwma2 0 **GPIO_EMC_10_ALT1** — Selecting Pad: GPIO_EMC_10 for Mode: ALT1 1 **GPIO_B0_10_ALT2** — Selecting Pad: GPIO_B0_10 for Mode: ALT2 pin 6

11.6.285 FLEXPWM2_PWMB3_SELECT_INPUT DAISY Register (IOMUXC_FLEXPWM2_PWMB3_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 484h offset = 401F_8484h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_FLEXPWM2_PWMB3_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: flexpwm2, In Pin: pwmb3 00 **GPIO_SD_B1_03_ALT2** — Selecting Pad: GPIO_SD_B1_03 for Mode: ALT2 01 **GPIO_EMC_20_ALT1** — Selecting Pad: GPIO_EMC_20 for Mode: ALT1 10 **GPIO_AD_B0_01_ALT0** — Selecting Pad: GPIO_AD_B0_01 for Mode: ALT0 11 **GPIO_B1_03_ALT6** — Selecting Pad: GPIO_B1_03 for Mode: ALT6 pin 37 (T4.1)

11.6.286 FLEXPWM2_PWMB0_SELECT_INPUT DAISY Register (IOMUXC_FLEXPWM2_PWMB0_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 488h offset = 401F_8488h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

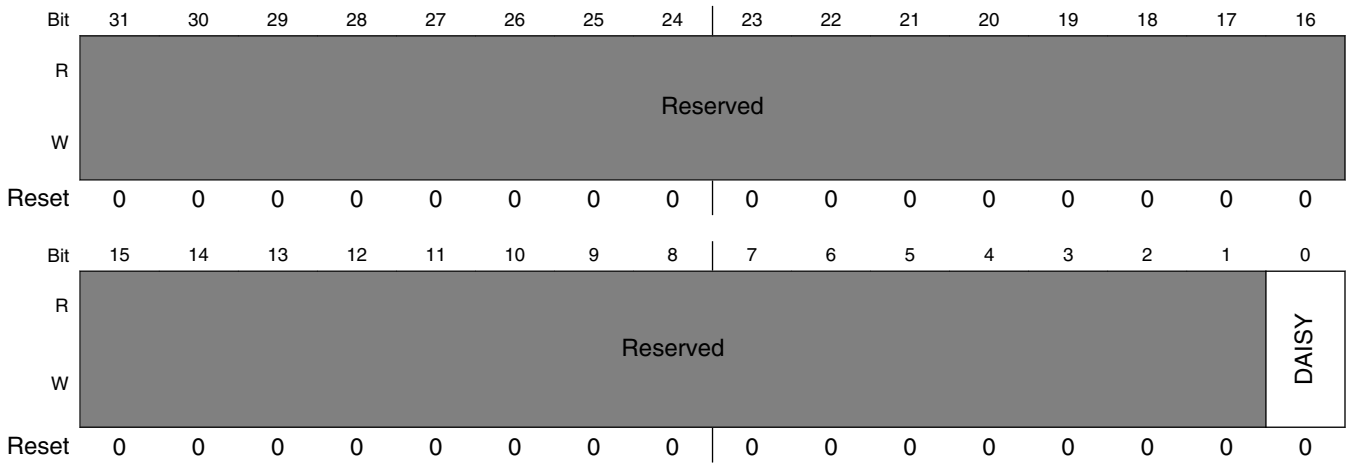
IOMUXC_FLEXPWM2_PWMB0_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: flexpwm2, In Pin: pwmb0 0 **GPIO_EMC_07_ALT1** — Selecting Pad: GPIO_EMC_07 for Mode: ALT1 pin 33 1 **GPIO_B0_07_ALT2** — Selecting Pad: GPIO_B0_07 for Mode: ALT2 pin 43 (MM)

11.6.287 FLEXPWM2_PWMB1_SELECT_INPUT DAISY Register (IOMUXC_FLEXPWM2_PWMB1_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 48Ch offset = 401F_848Ch



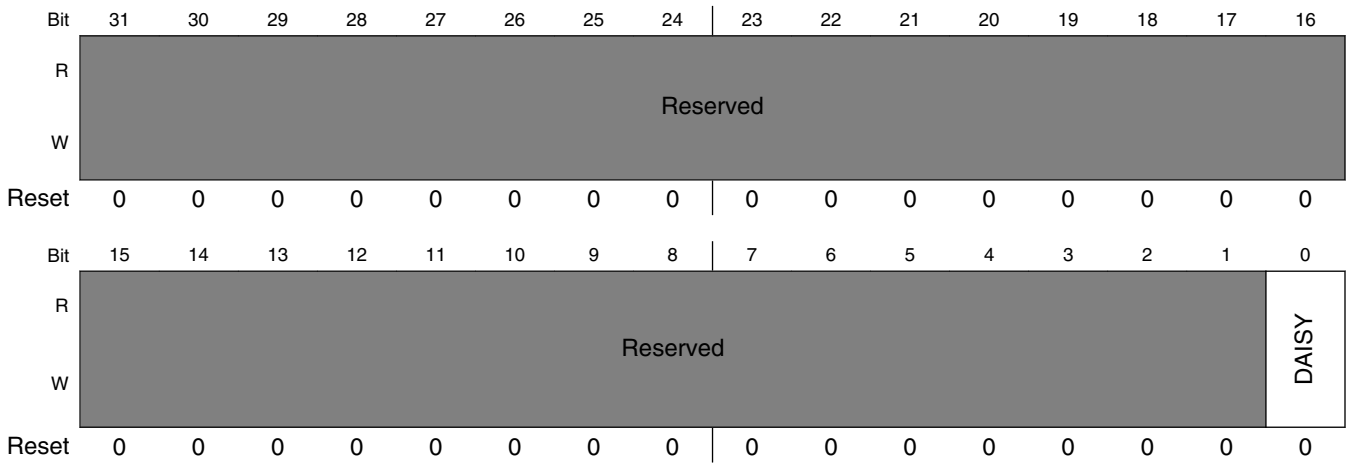
IOMUXC_FLEXPWM2_PWMB1_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: flexpwm2, In Pin: pwmb1 0 **GPIO_EMC_09_ALT1** — Selecting Pad: GPIO_EMC_09 for Mode: ALT1 1 **GPIO_B0_09_ALT2** — Selecting Pad: GPIO_B0_09 for Mode: ALT2 pin 45 (MM)

11.6.288 FLEXPWM2_PWMB2_SELECT_INPUT DAISY Register (IOMUXC_FLEXPWM2_PWMB2_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 490h offset = 401F_8490h



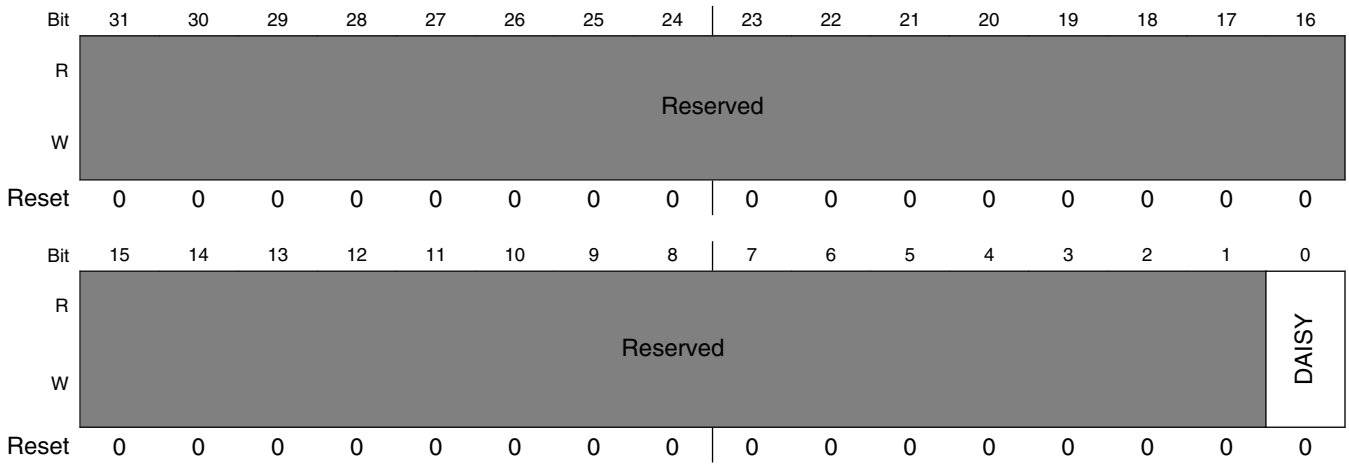
IOMUXC_FLEXPWM2_PWMB2_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: flexpwm2, In Pin: pwmb2 0 **GPIO_EMC_11_ALT1** — Selecting Pad: GPIO_EMC_11 for Mode: ALT1 1 **GPIO_B0_11_ALT2** — Selecting Pad: GPIO_B0_11 for Mode: ALT2 pin 9

11.6.289 FLEXPWM4_PWMA0_SELECT_INPUT DAISY Register (IOMUXC_FLEXPWM4_PWMA0_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 494h offset = 401F_8494h



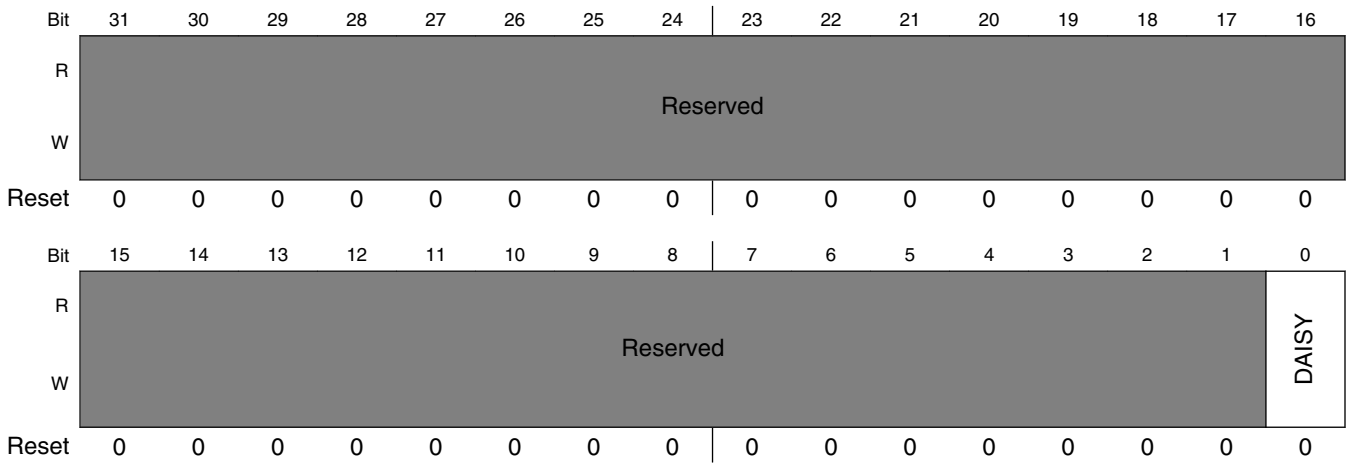
IOMUXC_FLEXPWM4_PWMA0_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: flexpwm4, In Pin: pwma0 0 **GPIO_EMC_00_ALT1** — Selecting Pad: GPIO_EMC_00 for Mode: ALT1 1 **GPIO_AD_B1_08_ALT1** — Selecting Pad: GPIO_AD_B1_08 for Mode: ALT1 pin 22

11.6.290 FLEXPWM4_PWMA1_SELECT_INPUT DAISY Register (IOMUXC_FLEXPWM4_PWMA1_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 498h offset = 401F_8498h



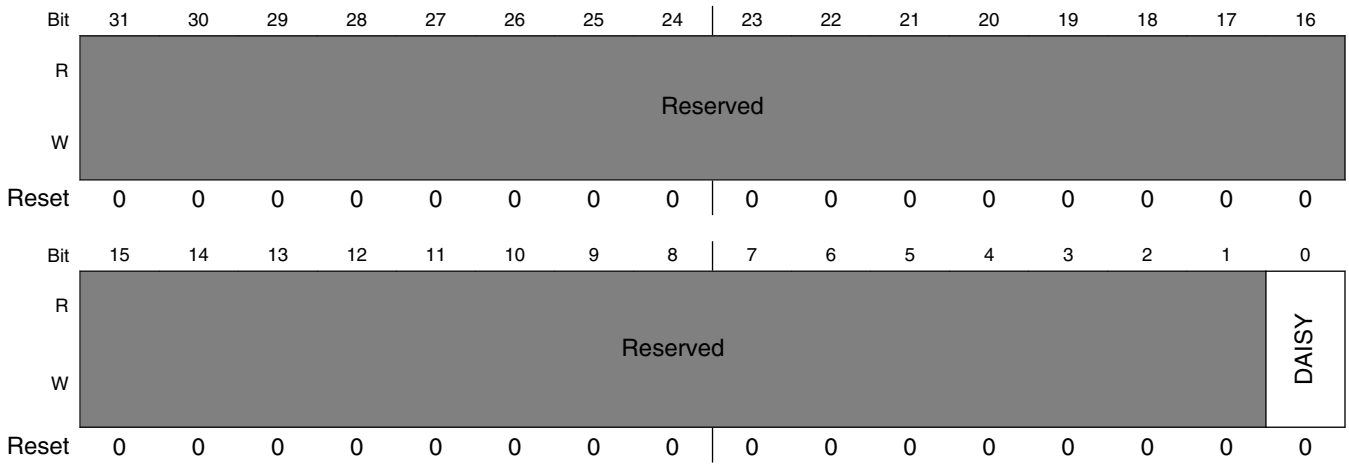
IOMUXC_FLEXPWM4_PWMA1_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: flexpwm4, In Pin: pwma1 0 **GPIO_EMC_02_ALT1** — Selecting Pad: GPIO_EMC_02 for Mode: ALT1 1 **GPIO_AD_B1_09_ALT1** — Selecting Pad: GPIO_AD_B1_09 for Mode: ALT1 pin 23

11.6.291 FLEXPWM4_PWMA2_SELECT_INPUT DAISY Register (IOMUXC_FLEXPWM4_PWMA2_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 49Ch offset = 401F_849Ch



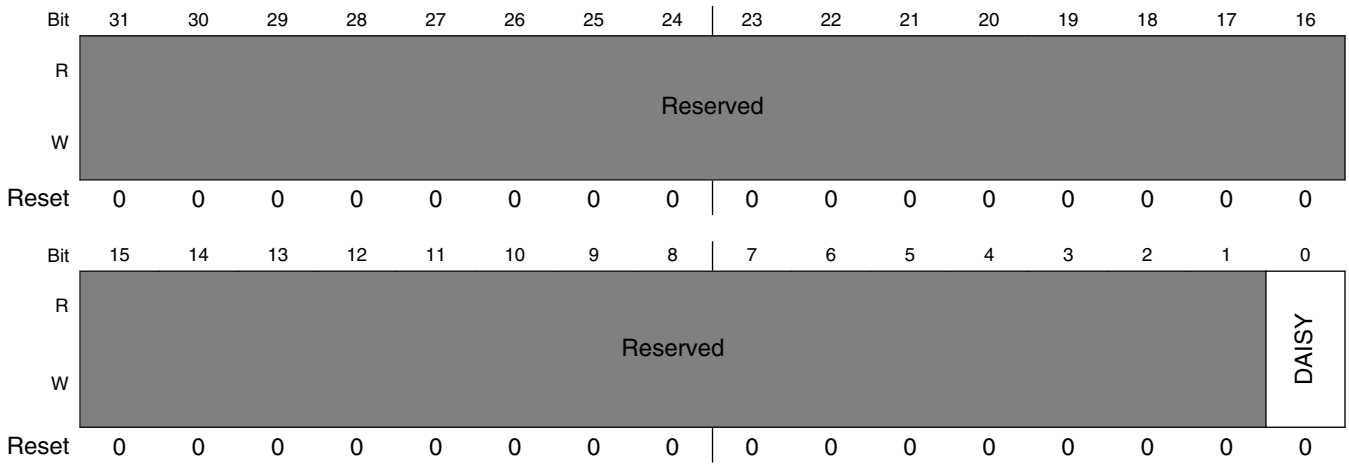
IOMUXC_FLEXPWM4_PWMA2_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: flexpwm4, In Pin: pwma2 0 **GPIO_EMC_04_ALT1** — Selecting Pad: GPIO_EMC_04 for Mode: ALT1 pin 2 1 **GPIO_B1_14_ALT1** — Selecting Pad: GPIO_B1_14 for Mode: ALT1

11.6.292 FLEXPWM4_PWMA3_SELECT_INPUT DAISY Register (IOMUXC_FLEXPWM4_PWMA3_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 4A0h offset = 401F_84A0h



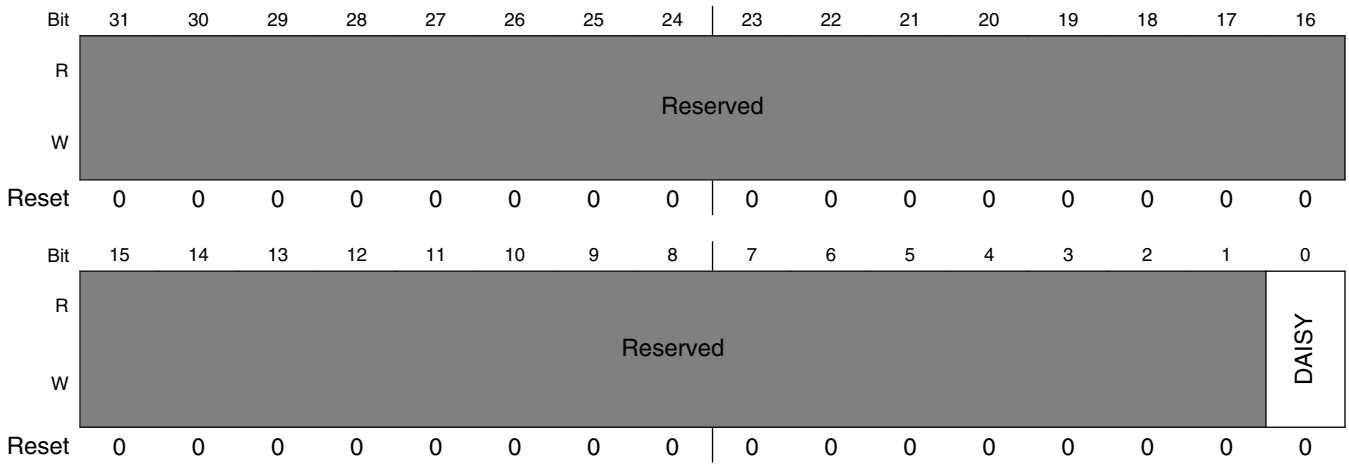
IOMUXC_FLEXPWM4_PWMA3_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: flexpwm4, In Pin: pwma3 0 **GPIO_EMC_17_ALT1** — Selecting Pad: GPIO_EMC_17 for Mode: ALT1 1 **GPIO_B1_15_ALT1** — Selecting Pad: GPIO_B1_15 for Mode: ALT1

11.6.293 FLEXSPIA_DQS_SELECT_INPUT DAISY Register (IOMUXC_FLEXSPIA_DQS_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 4A4h offset = 401F_84A4h



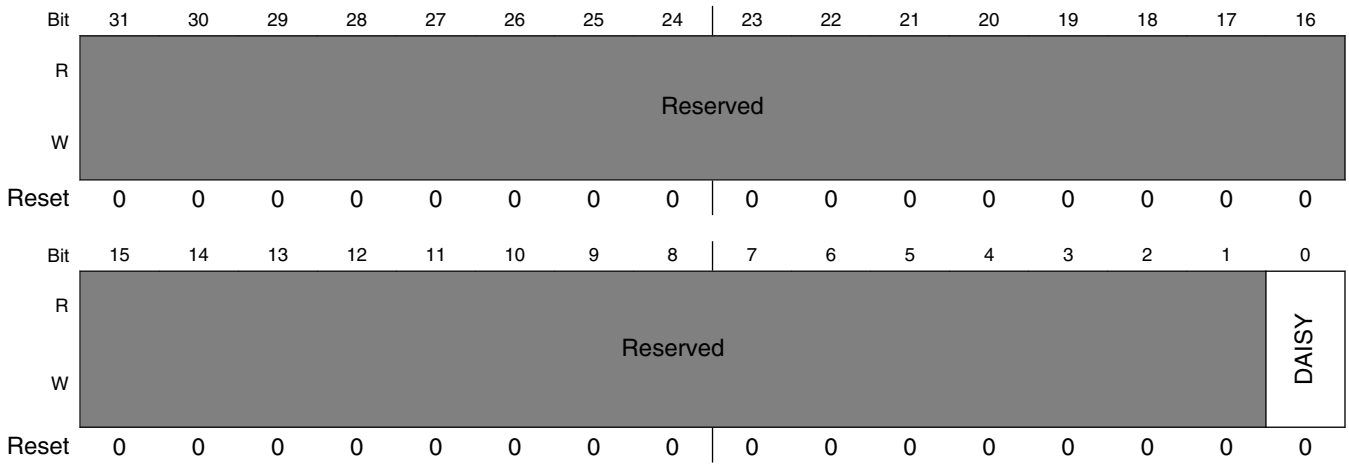
IOMUXC_FLEXSPIA_DQS_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: flexspi, In Pin: dqs 0 **GPIO_SD_B1_05_ALT1** — Selecting Pad: GPIO_SD_B1_05 for Mode: ALT1 1 **GPIO_AD_B1_09_ALT0** — Selecting Pad: GPIO_AD_B1_09 for Mode: ALT0 pin 23

11.6.294 FLEXSPIA_DATA0_SELECT_INPUT DAISY Register (IOMUXC_FLEXSPIA_DATA0_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 4A8h offset = 401F_84A8h



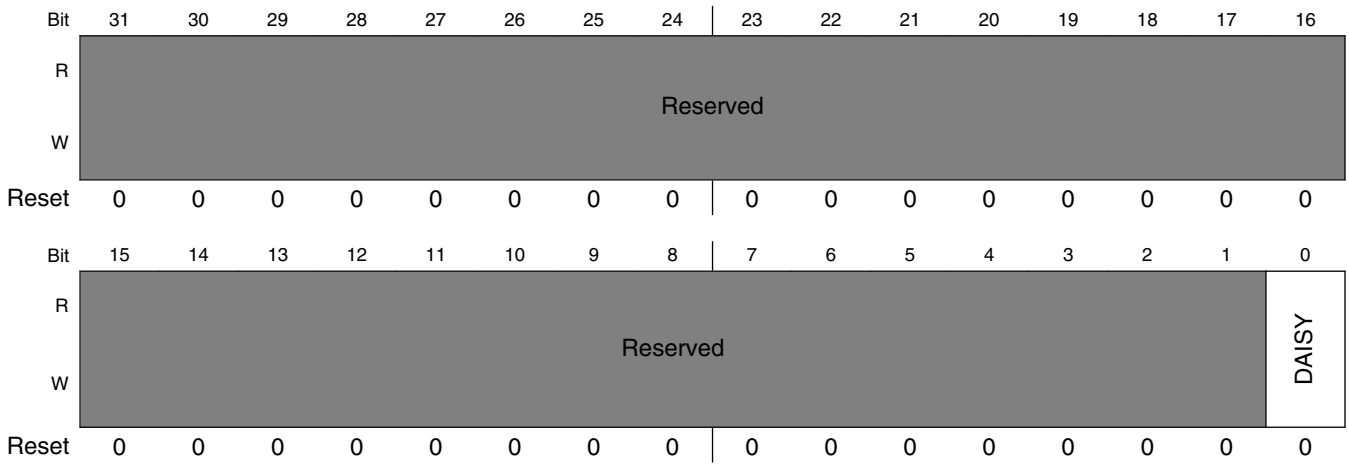
IOMUXC_FLEXSPIA_DATA0_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: flexspi, In Pin: data0 0 **GPIO_SD_B1_08_ALT1** — Selecting Pad: GPIO_SD_B1_08 for Mode: ALT1 1 **GPIO_AD_B1_13_ALT0** — Selecting Pad: GPIO_AD_B1_13 for Mode: ALT0 pin 39 (T4.1)

11.6.295 FLEXSPIA_DATA1_SELECT_INPUT DAISY Register (IOMUXC_FLEXSPIA_DATA1_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 4ACh offset = 401F_84ACh



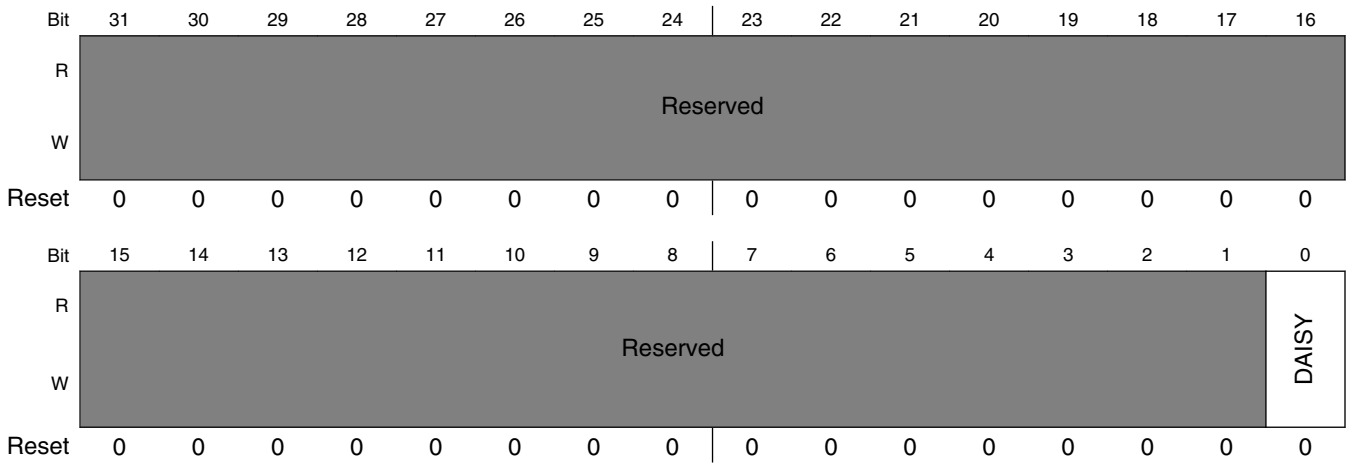
IOMUXC_FLEXSPIA_DATA1_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: flexspi, In Pin: data1 0 **GPIO_SD_B1_09_ALT1** — Selecting Pad: GPIO_SD_B1_09 for Mode: ALT1 1 **GPIO_AD_B1_12_ALT0** — Selecting Pad: GPIO_AD_B1_12 for Mode: ALT0 pin 38 (T4.1)

11.6.296 FLEXSPIA_DATA2_SELECT_INPUT DAISY Register (IOMUXC_FLEXSPIA_DATA2_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 4B0h offset = 401F_84B0h



IOMUXC_FLEXSPIA_DATA2_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: flexspi, In Pin: data2 0 **GPIO_SD_B1_10_ALT1** — Selecting Pad: GPIO_SD_B1_10 for Mode: ALT1 1 **GPIO_AD_B1_11_ALT0** — Selecting Pad: GPIO_AD_B1_11 for Mode: ALT0 pin 21

11.6.297 FLEXSPIA_DATA3_SELECT_INPUT DAISY Register (IOMUXC_FLEXSPIA_DATA3_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 4B4h offset = 401F_84B4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

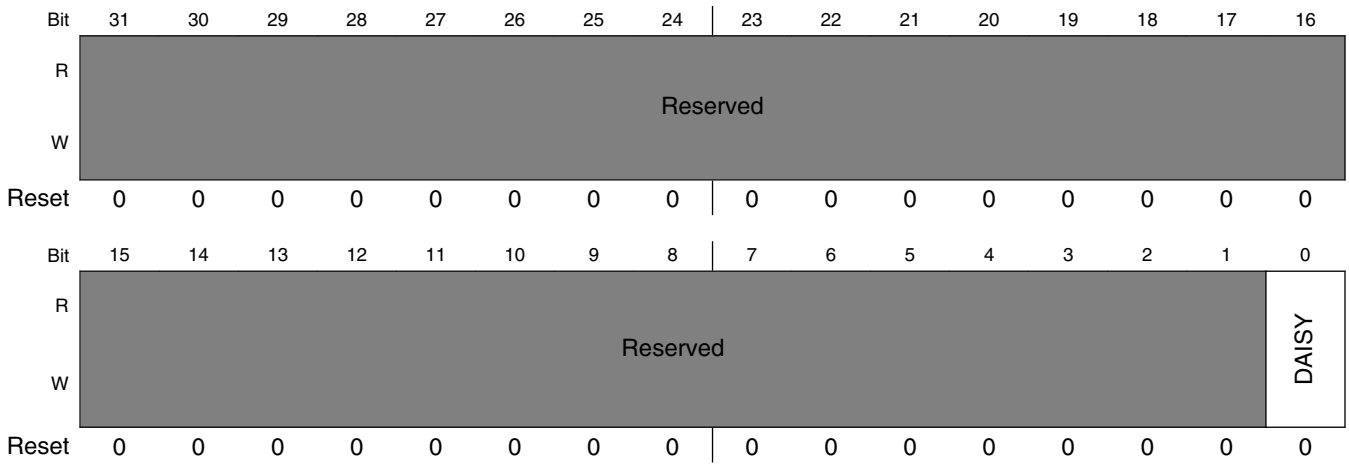
IOMUXC_FLEXSPIA_DATA3_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: flexspi, In Pin: data3 0 **GPIO_SD_B1_11_ALT1** — Selecting Pad: GPIO_SD_B1_11 for Mode: ALT1 1 **GPIO_AD_B1_10_ALT0** — Selecting Pad: GPIO_AD_B1_10 for Mode: ALT0 pin 20

11.6.298 FLEXSPIB_DATA0_SELECT_INPUT DAISY Register (IOMUXC_FLEXSPIB_DATA0_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 4B8h offset = 401F_84B8h



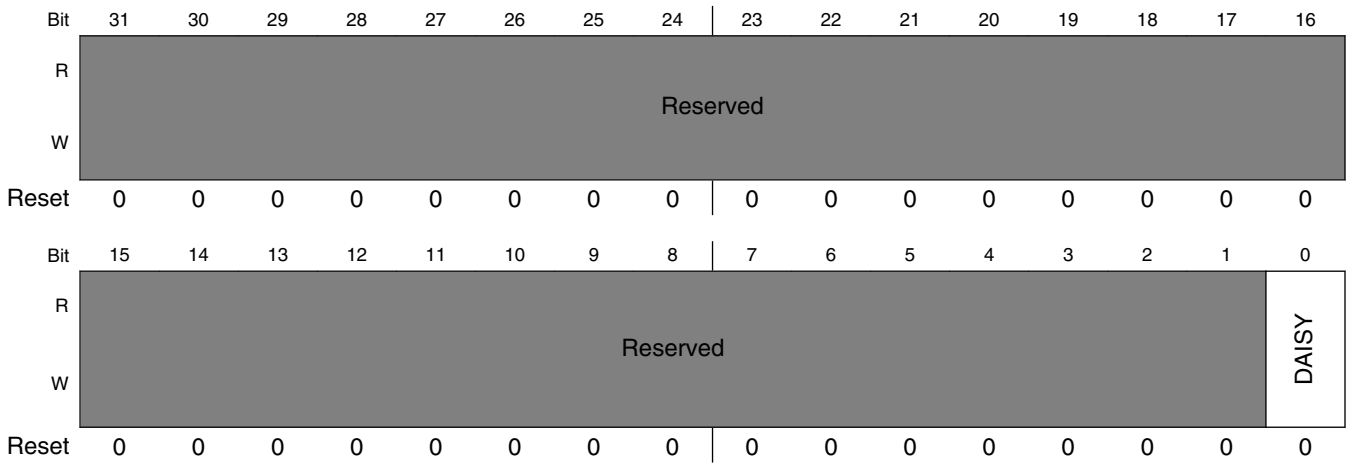
IOMUXC_FLEXSPIB_DATA0_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: flexspi, In Pin: data0 0 **GPIO_SD_B1_03_ALT1** — Selecting Pad: GPIO_SD_B1_03 for Mode: ALT1 1 **GPIO_AD_B1_07_ALT0** — Selecting Pad: GPIO_AD_B1_07 for Mode: ALT0 pin 16

11.6.299 FLEXSPIB_DATA1_SELECT_INPUT DAISY Register (IOMUXC_FLEXSPIB_DATA1_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 4BCh offset = 401F_84BCh



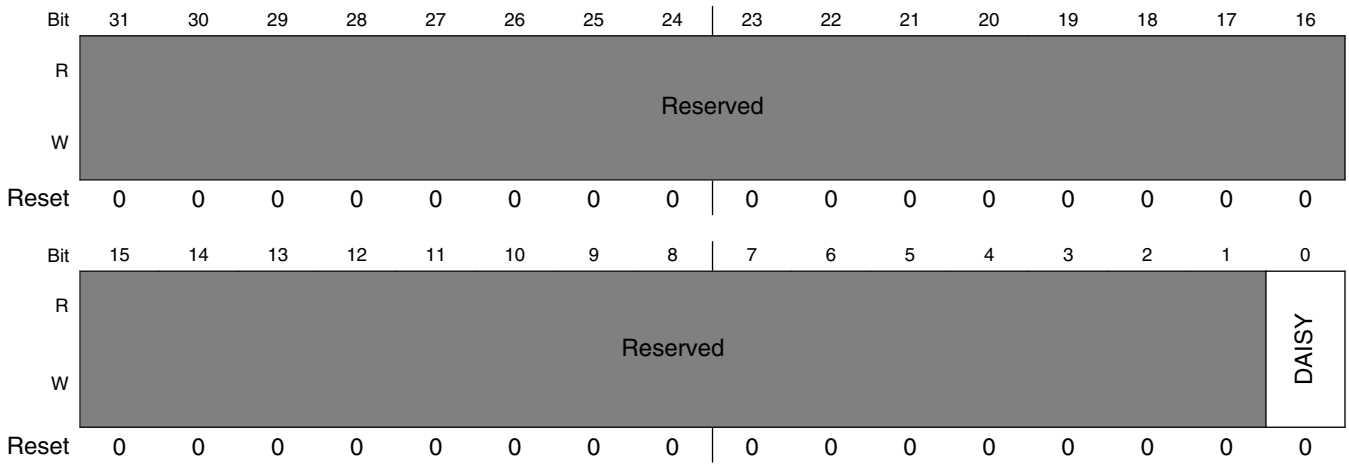
IOMUXC_FLEXSPIB_DATA1_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: flexspi, In Pin: data1 0 **GPIO_SD_B1_02_ALT1** — Selecting Pad: GPIO_SD_B1_02 for Mode: ALT1 1 **GPIO_AD_B1_06_ALT0** — Selecting Pad: GPIO_AD_B1_06 for Mode: ALT0 pin 17

11.6.300 FLEXSPIB_DATA2_SELECT_INPUT DAISY Register (IOMUXC_FLEXSPIB_DATA2_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 4C0h offset = 401F_84C0h



IOMUXC_FLEXSPIB_DATA2_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: flexspi, In Pin: data2 0 **GPIO_SD_B1_01_ALT1** — Selecting Pad: GPIO_SD_B1_01 for Mode: ALT1 1 **GPIO_AD_B1_05_ALT0** — Selecting Pad: GPIO_AD_B1_05 for Mode: ALT0 pin 41 (T4.1)

11.6.301 FLEXSPIB_DATA3_SELECT_INPUT DAISY Register (IOMUXC_FLEXSPIB_DATA3_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 4C4h offset = 401F_84C4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

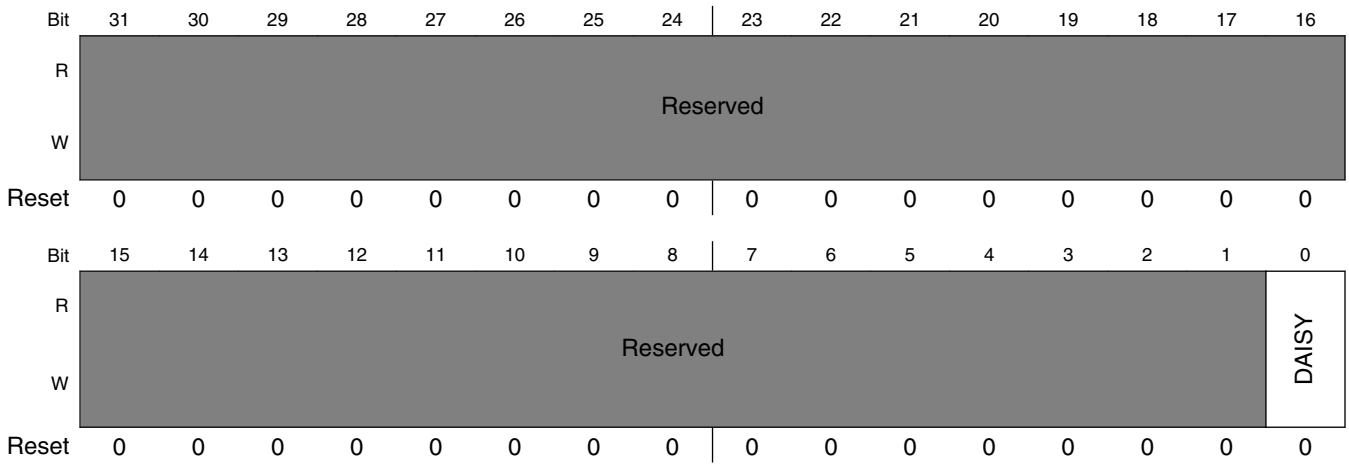
IOMUXC_FLEXSPIB_DATA3_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: flexspi, In Pin: data3 0 **GPIO_SD_B1_00_ALT1** — Selecting Pad: GPIO_SD_B1_00 for Mode: ALT1 1 **GPIO_AD_B1_04_ALT0** — Selecting Pad: GPIO_AD_B1_04 for Mode: ALT0 pin 40 (T4.1)

11.6.302 FLEXSPIA_SCK_SELECT_INPUT DAISY Register (IOMUXC_FLEXSPIA_SCK_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 4C8h offset = 401F_84C8h



IOMUXC_FLEXSPIA_SCK_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: flexspi, In Pin: sck_fa 0 **GPIO_SD_B1_07_ALT1** — Selecting Pad: GPIO_SD_B1_07 for Mode: ALT1 1 **GPIO_AD_B1_14_ALT0** — Selecting Pad: GPIO_AD_B1_14 for Mode: ALT0 pin 26

11.6.303 LPI2C1_SCL_SELECT_INPUT DAISY Register (IOMUXC_LPI2C1_SCL_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 4CCh offset = 401F_84CCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

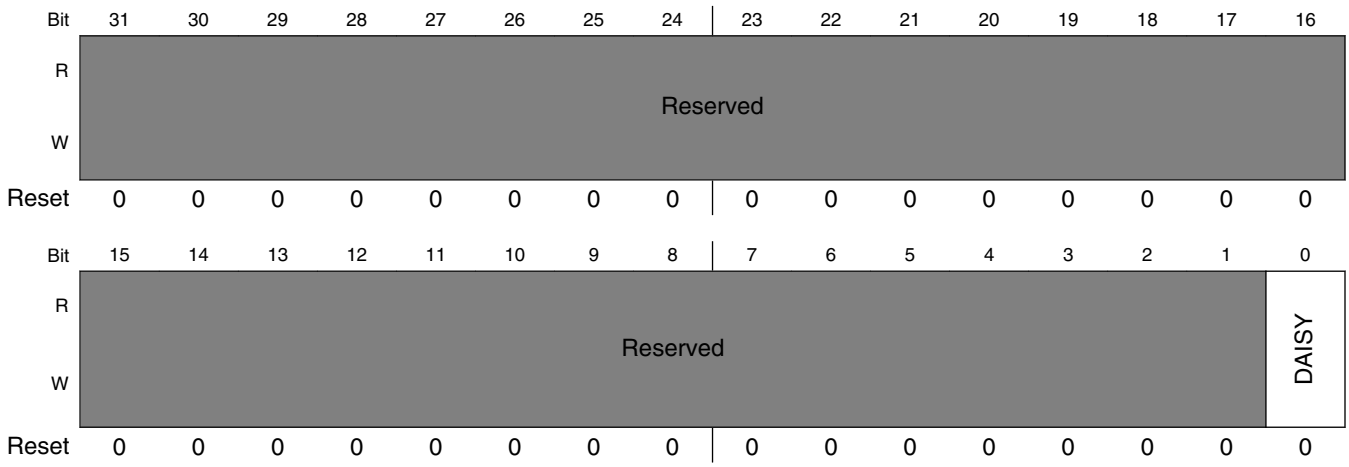
IOMUXC_LPI2C1_SCL_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: lpi2c1, In Pin: lpi2c_scl 0 **GPIO_SD_B1_04_ALT2** — Selecting Pad: GPIO_SD_B1_04 for Mode: ALT2 1 **GPIO_AD_B1_00_ALT3** — Selecting Pad: GPIO_AD_B1_00 for Mode: ALT3 pin 19

11.6.304 LPI2C1_SDA_SELECT_INPUT DAISY Register (IOMUXC_LPI2C1_SDA_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 4D0h offset = 401F_84D0h



IOMUXC_LPI2C1_SDA_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: lpi2c1, In Pin: lpi2c_sda 0 **GPIO_SD_B1_05_ALT2** — Selecting Pad: GPIO_SD_B1_05 for Mode: ALT2 1 **GPIO_AD_B1_01_ALT3** — Selecting Pad: GPIO_AD_B1_01 for Mode: ALT3 pin 18

11.6.305 LPI2C2_SCL_SELECT_INPUT DAISY Register (IOMUXC_LPI2C2_SCL_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 4D4h offset = 401F_84D4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LPI2C2_SCL_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: lpi2c2, In Pin: lpi2c_scl 0 **GPIO_SD_B1_11_ALT3** — Selecting Pad: GPIO_SD_B1_11 for Mode: ALT3 1 **GPIO_B0_04_ALT2** — Selecting Pad: GPIO_B0_04 for Mode: ALT2 pin 40 (MM)

11.6.306 LPI2C2_SDA_SELECT_INPUT DAISY Register (IOMUXC_LPI2C2_SDA_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 4D8h offset = 401F_84D8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LPI2C2_SDA_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: lpi2c2, In Pin: lpi2c_sda 0 **GPIO_SD_B1_10_ALT3** — Selecting Pad: GPIO_SD_B1_10 for Mode: ALT3 1 **GPIO_B0_05_ALT2** — Selecting Pad: GPIO_B0_05 for Mode: ALT2 pin 41 (MM)

11.6.307 LPI2C3_SCL_SELECT_INPUT DAISY Register (IOMUXC_LPI2C3_SCL_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 4DCh offset = 401F_84DCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LPI2C3_SCL_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: lpi2c3, In Pin: lpi2c_scl 00 **GPIO_EMC_22_ALT2** — Selecting Pad: GPIO_EMC_22 for Mode: ALT2 01 **GPIO_SD_B0_00_ALT2** — Selecting Pad: GPIO_SD_B0_00 for Mode: ALT2 10 **GPIO_AD_B1_07_ALT1** — Selecting Pad: GPIO_AD_B1_07 for Mode: ALT1 pin 16

11.6.308 LPI2C3_SDA_SELECT_INPUT DAISY Register (IOMUXC_LPI2C3_SDA_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 4E0h offset = 401F_84E0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

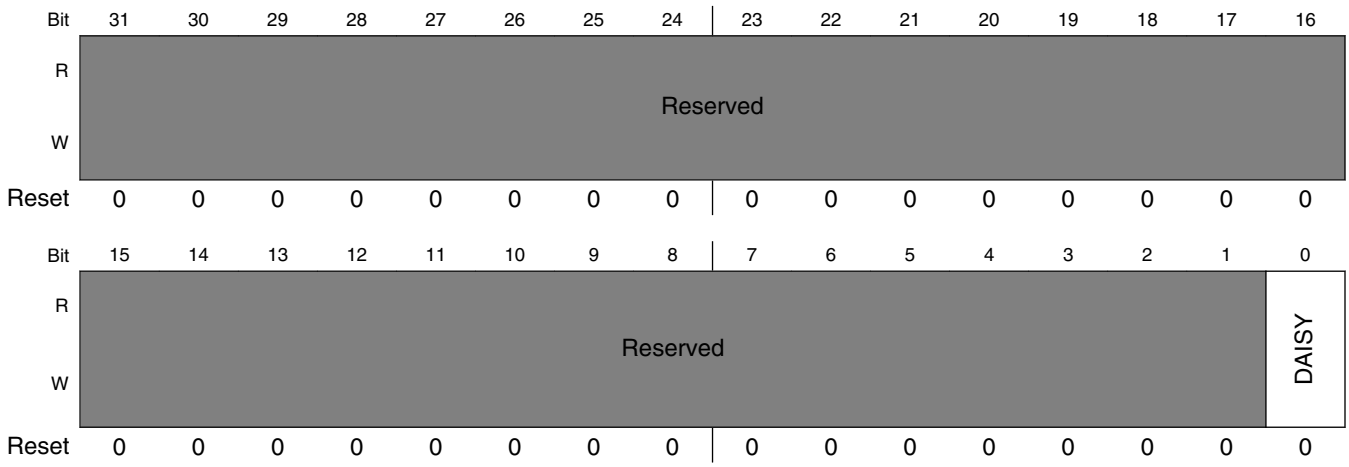
IOMUXC_LPI2C3_SDA_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: lpi2c3, In Pin: lpi2c_sda 00 **GPIO_EMC_21_ALT2** — Selecting Pad: GPIO_EMC_21 for Mode: ALT2 01 **GPIO_SD_B0_01_ALT2** — Selecting Pad: GPIO_SD_B0_01 for Mode: ALT2 10 **GPIO_AD_B1_06_ALT1** — Selecting Pad: GPIO_AD_B1_06 for Mode: ALT1 pin 17

11.6.309 LPI2C4_SCL_SELECT_INPUT DAISY Register (IOMUXC_LPI2C4_SCL_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 4E4h offset = 401F_84E4h



IOMUXC_LPI2C4_SCL_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: lpi2c4, In Pin: lpi2c_scl 0 **GPIO_EMC_12_ALT2** — Selecting Pad: GPIO_EMC_12 for Mode: ALT2 1 **GPIO_AD_B0_12_ALT0** — Selecting Pad: GPIO_AD_B0_12 for Mode: ALT0 pin 24

11.6.310 LPI2C4_SDA_SELECT_INPUT DAISY Register (IOMUXC_LPI2C4_SDA_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 4E8h offset = 401F_84E8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

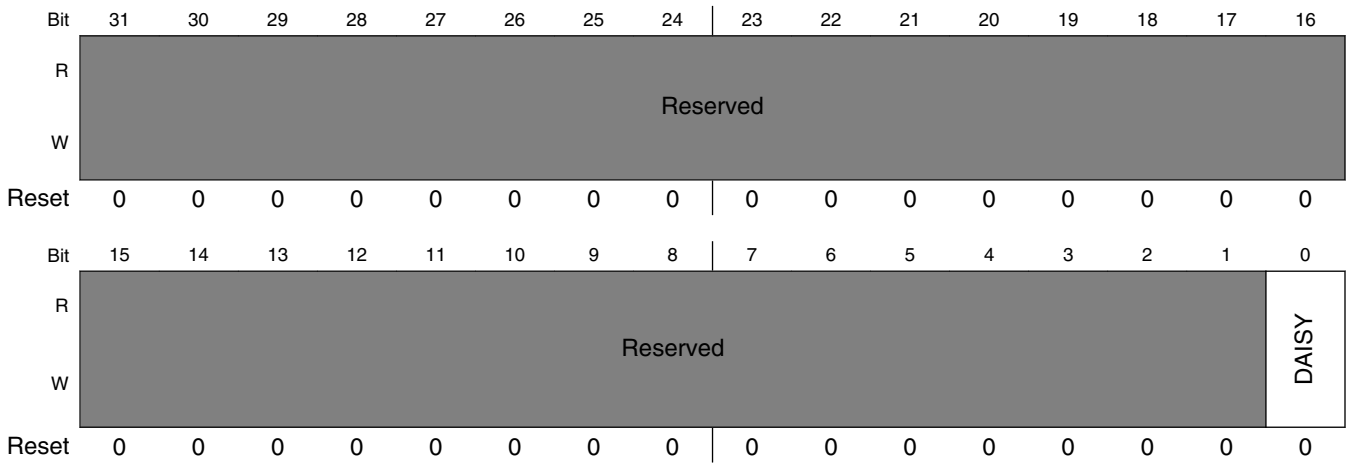
IOMUXC_LPI2C4_SDA_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: lpi2c4, In Pin: lpi2c_sda 0 **GPIO_EMC_11_ALT2** — Selecting Pad: GPIO_EMC_11 for Mode: ALT2 1 **GPIO_AD_B0_13_ALT0** — Selecting Pad: GPIO_AD_B0_13 for Mode: ALT0 pin 25

11.6.311 LPSPI1_PCS0_SELECT_INPUT DAISY Register (IOMUXC_LPSPI1_PCS0_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 4ECh offset = 401F_84ECh



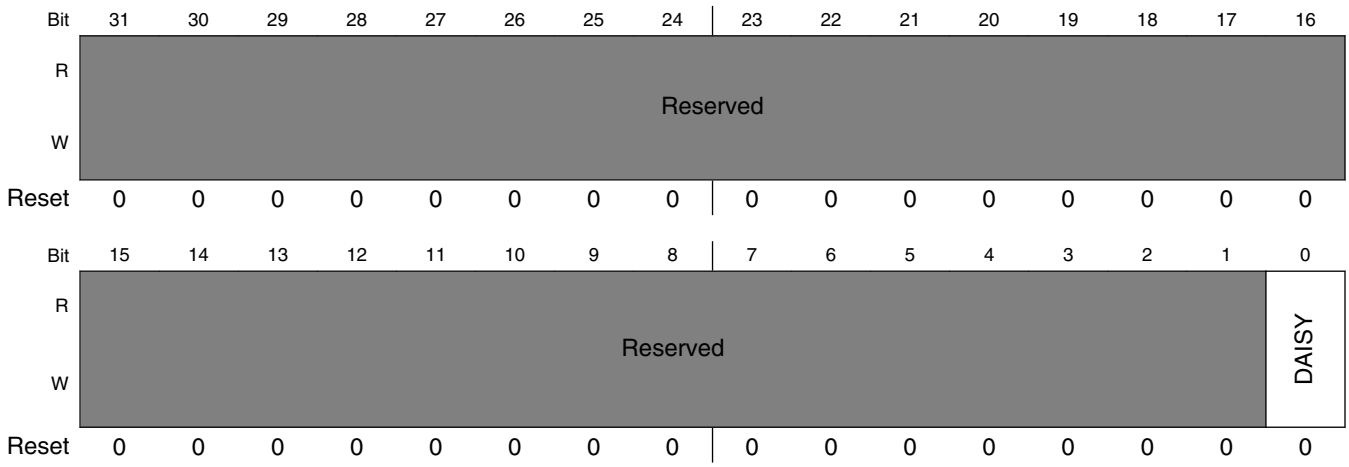
IOMUXC_LPSPI1_PCS0_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: lpspi1, In Pin: lpspi_pcs0 0 **GPIO_SD_B0_01_ALT4** — Selecting Pad: GPIO_SD_B0_01 for Mode: ALT4 1 **GPIO_EMC_30_ALT3** — Selecting Pad: GPIO_EMC_30 for Mode: ALT3

11.6.312 LPSPI1_SCK_SELECT_INPUT DAISY Register (IOMUXC_LPSPI1_SCK_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 4F0h offset = 401F_84F0h



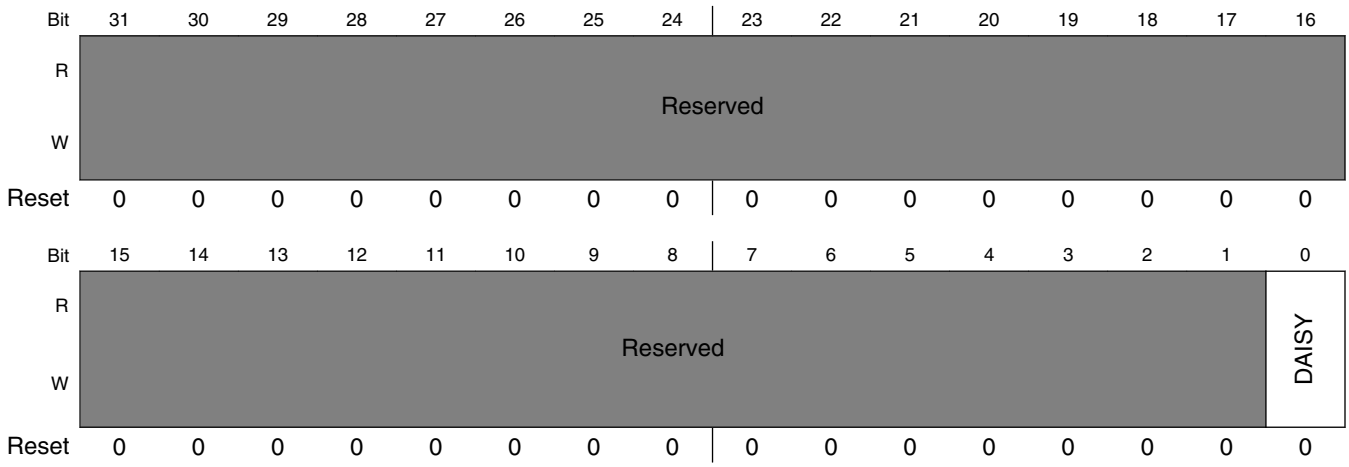
IOMUXC_LPSPI1_SCK_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: lpspi1, In Pin: lpspi_sck 0 **GPIO_EMC_27_ALT3** — Selecting Pad: GPIO_EMC_27 for Mode: ALT3 1 **GPIO_SD_B0_00_ALT4** — Selecting Pad: GPIO_SD_B0_00 for Mode: ALT4

11.6.313 LPSPI1_SDI_SELECT_INPUT DAISY Register (IOMUXC_LPSPI1_SDI_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 4F4h offset = 401F_84F4h



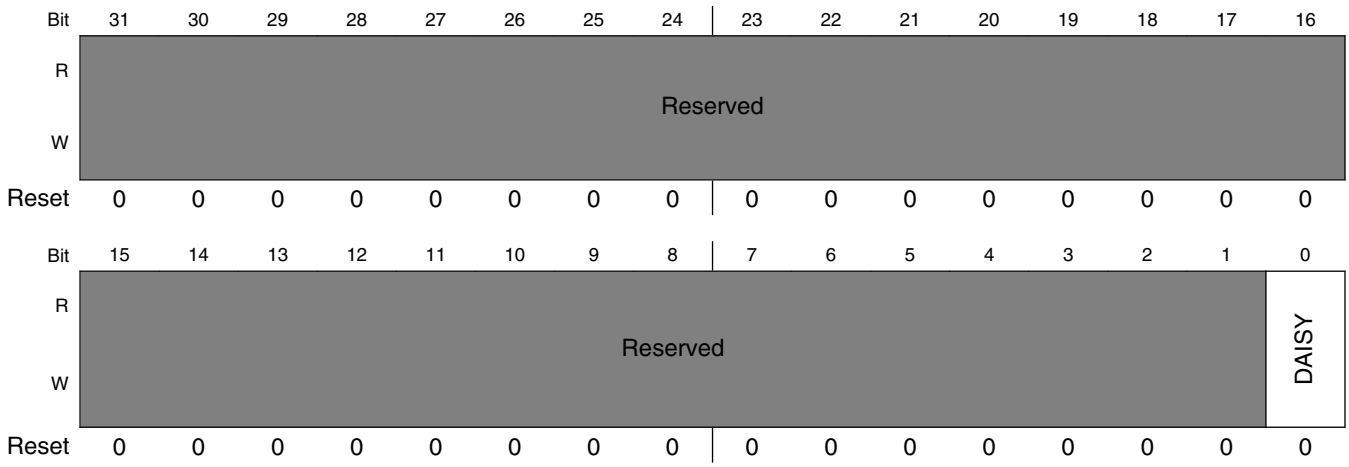
IOMUXC_LPSPI1_SDI_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: lpspi1, In Pin: lpspi_sdi 0 **GPIO_EMC_29_ALT3** — Selecting Pad: GPIO_EMC_29 for Mode: ALT3 1 **GPIO_SD_B0_03_ALT4** — Selecting Pad: GPIO_AD_B0_03 for Mode: ALT4 pin 0

11.6.314 LPSPI1_SDO_SELECT_INPUT DAISY Register (IOMUXC_LPSPI1_SDO_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 4F8h offset = 401F_84F8h



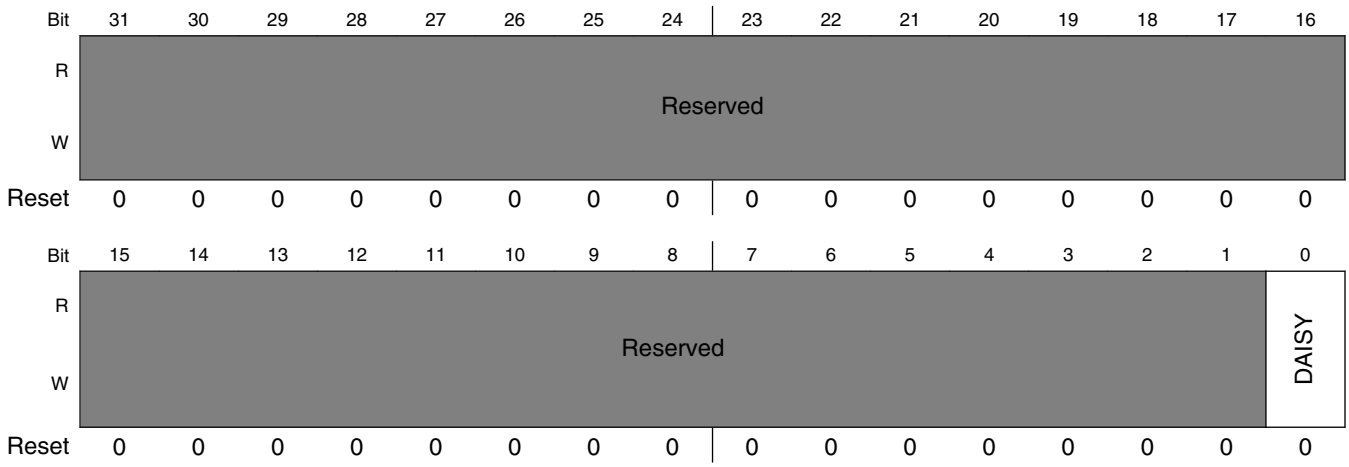
IOMUXC_LPSPI1_SDO_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: lpspi1, In Pin: lpspi_sdo 0 **GPIO_EMC_28_ALT3** — Selecting Pad: GPIO_EMC_28 for Mode: ALT3 1 **GPIO_SD_B0_02_ALT4** — Selecting Pad: GPIO_SD_B0_02 for Mode: ALT4

11.6.315 LPSPI2_PCS0_SELECT_INPUT DAISY Register (IOMUXC_LPSPI2_PCS0_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 4FCh offset = 401F_84FCh



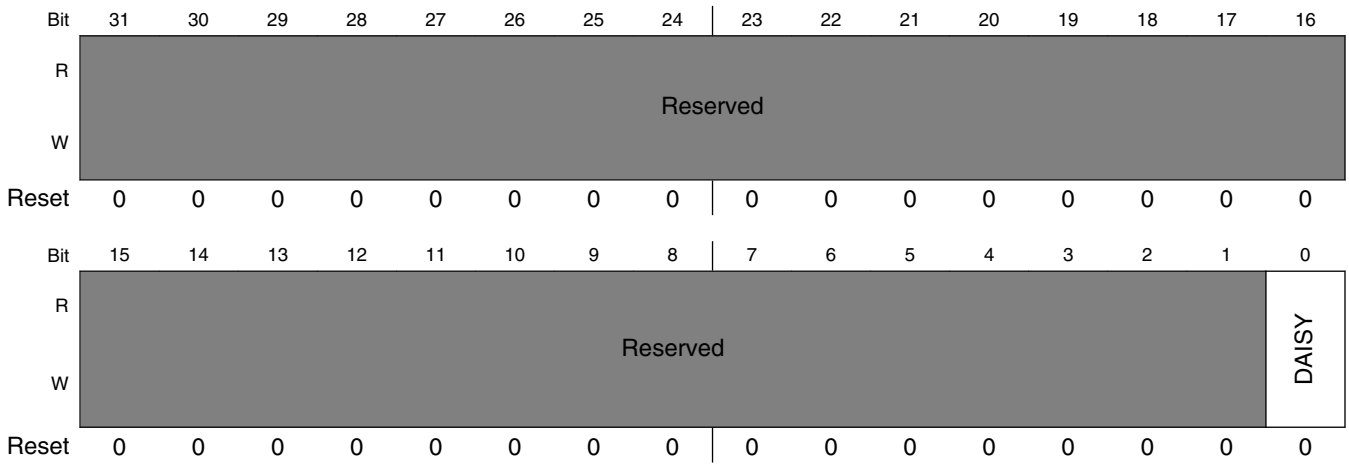
IOMUXC_LPSPI2_PCS0_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: lpspi2, In Pin: lpspi_pcs0 0 **GPIO_SD_B1_06_ALT4** — Selecting Pad: GPIO_SD_B1_06 for Mode: ALT4 1 **GPIO_EMC_01_ALT2** — Selecting Pad: GPIO_EMC_01 for Mode: ALT2

11.6.316 LPSPI2_SCK_SELECT_INPUT DAISY Register (IOMUXC_LPSPI2_SCK_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 500h offset = 401F_8500h



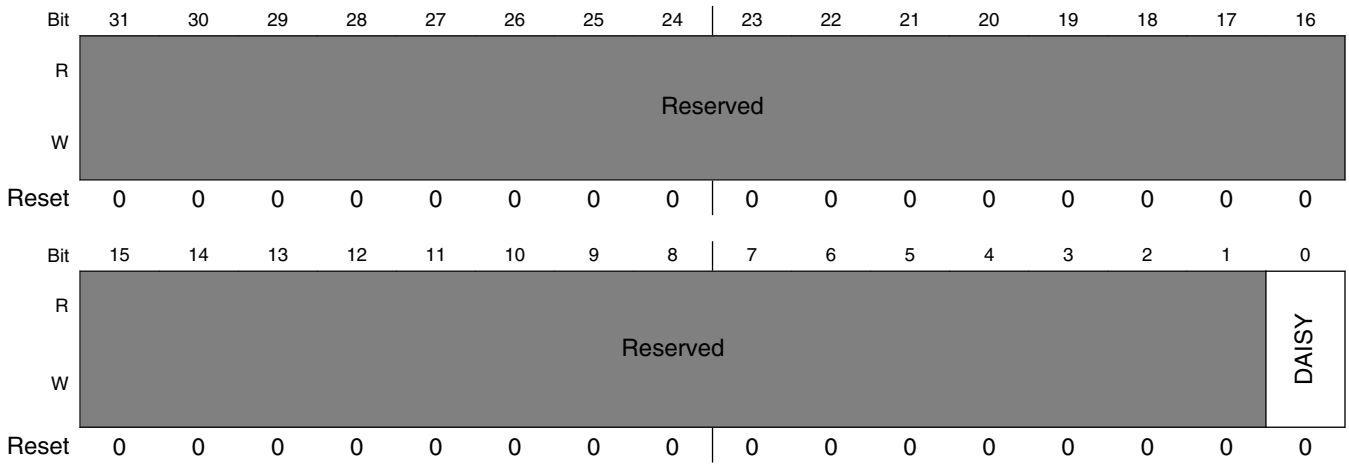
IOMUXC_LPSPI2_SCK_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: lpspi2, In Pin: lpspi_sck 0 **GPIO_SD_B1_07_ALT4** — Selecting Pad: GPIO_SD_B1_07 for Mode: ALT4 1 **GPIO_EMC_00_ALT2** — Selecting Pad: GPIO_EMC_00 for Mode: ALT2

11.6.317 LPSPI2_SDI_SELECT_INPUT DAISY Register (IOMUXC_LPSPI2_SDI_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 504h offset = 401F_8504h



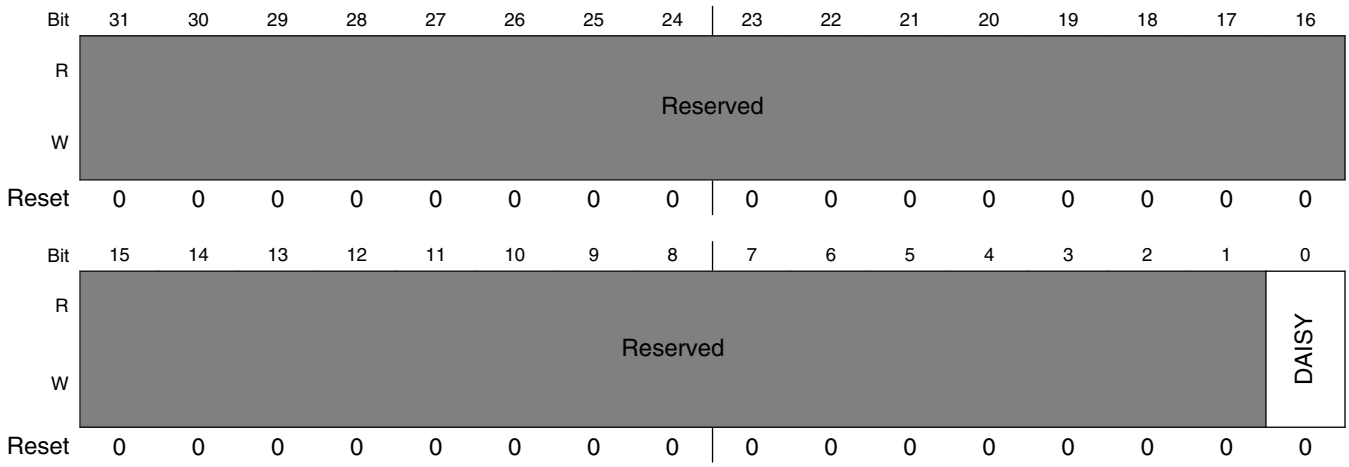
IOMUXC_LPSPI2_SDI_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: lpspi2, In Pin: lpspi_sdi 0 **GPIO_SD_B1_09_ALT4** — Selecting Pad: GPIO_SD_B1_09 for Mode: ALT4 1 **GPIO_EMC_03_ALT2** — Selecting Pad: GPIO_EMC_03 for Mode: ALT2

11.6.318 LPSPI2_SDO_SELECT_INPUT DAISY Register (IOMUXC_LPSPI2_SDO_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 508h offset = 401F_8508h



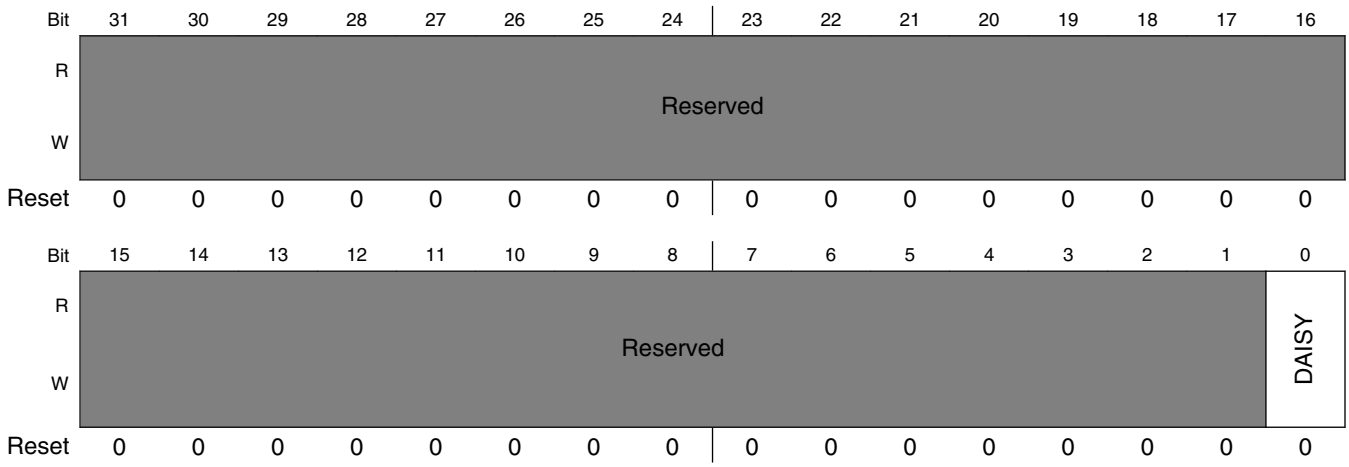
IOMUXC_LPSPI2_SDO_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: lpspi2, In Pin: lpspi_sdo 0 **GPIO_SD_B1_08_ALT4** — Selecting Pad: GPIO_SD_B1_08 for Mode: ALT4 1 **GPIO_EMC_02_ALT2** — Selecting Pad: GPIO_EMC_02 for Mode: ALT2

11.6.319 LPSPI3_PCS0_SELECT_INPUT DAISY Register (IOMUXC_LPSPI3_PCS0_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 50Ch offset = 401F_850Ch



IOMUXC_LPSPI3_PCS0_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: lpspi3, In Pin: lpspi_pcs0 0 **GPIO_AD_B0_03_ALT7** — Selecting Pad: GPIO_AD_B0_03 for Mode: ALT7 pin 0 1 **GPIO_AD_B1_12_ALT2** — Selecting Pad: GPIO_AD_B1_12 for Mode: ALT2 pin 38 (T4.1)

11.6.320 LPSPI3_SCK_SELECT_INPUT DAISY Register (IOMUXC_LPSPI3_SCK_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 510h offset = 401F_8510h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

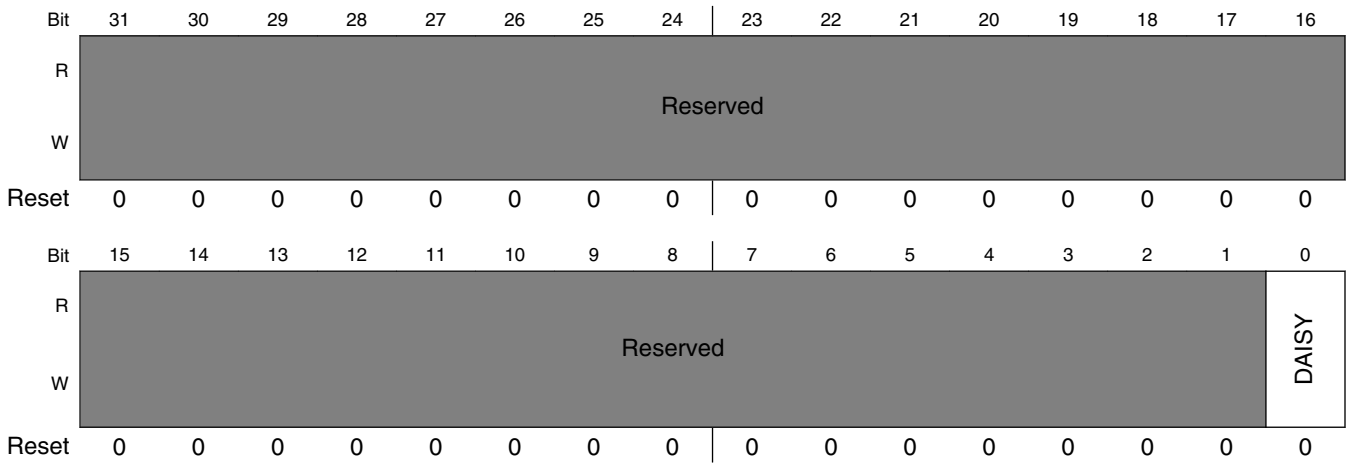
IOMUXC_LPSPI3_SCK_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: lpspi3, In Pin: lpspi_sck 0 **GPIO_AD_B0_00_ALT7** — Selecting Pad: GPIO_AD_B0_00 for Mode: ALT7 1 **GPIO_AD_B1_15_ALT2** — Selecting Pad: GPIO_AD_B1_15 for Mode: ALT2 pin 27

11.6.321 LPSPI3_SDI_SELECT_INPUT DAISY Register (IOMUXC_LPSPI3_SDI_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 514h offset = 401F_8514h



IOMUXC_LPSPI3_SDI_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: lpspi3, In Pin: lpspi_sdi 1 **GPIO_AD_B1_13_ALT2** — Selecting Pad: GPIO_AD_B1_13 for Mode: ALT2 pin 39 (T4.1) 0 **GPIO_AD_B0_02_ALT7** — Selecting Pad: GPIO_AD_B0_02 for Mode: ALT7 pin 1

11.6.322 LPSPI3_SDO_SELECT_INPUT DAISY Register (IOMUXC_LPSPI3_SDO_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 518h offset = 401F_8518h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LPSPI3_SDO_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: lpspi3, In Pin: lpspi_sdo 1 **GPIO_AD_B1_14_ALT2** — Selecting Pad: GPIO_AD_B1_14 for Mode: ALT2 pin 26 0 **GPIO_AD_B0_01_ALT7** — Selecting Pad: GPIO_AD_B0_01 for Mode: ALT7

11.6.323 LPSPI4_PCS0_SELECT_INPUT DAISY Register (IOMUXC_LPSPI4_PCS0_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 51Ch offset = 401F_851Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LPSPI4_PCS0_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: lpspi4, In Pin: lpspi_pcs0 0 **GPIO_B0_00_ALT3** — Selecting Pad: GPIO_B0_00 for Mode: ALT3 pin 10 1 **GPIO_B1_04_ALT1** — Selecting Pad:GPIO_B1_04 for Mode: ALT1

11.6.324 LPSPI4_SCK_SELECT_INPUT DAISY Register (IOMUXC_LPSPI4_SCK_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 520h offset = 401F_8520h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

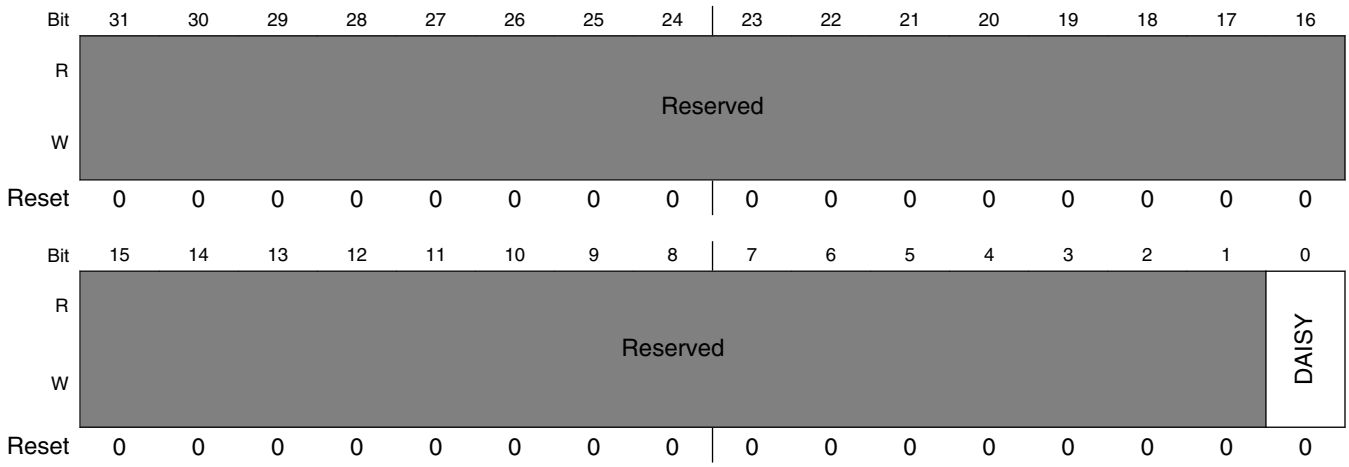
IOMUXC_LPSPI4_SCK_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: lpspi4, In Pin: lpspi_sck 0 **GPIO_B0_03_ALT3** — Selecting Pad: GPIO_B0_03 for Mode: ALT3 pin 13 1 **GPIO_B1_07_ALT1** — Selecting Pad: GPIO_B1_07 for Mode: ALT1

11.6.325 LPSPI4_SDI_SELECT_INPUT DAISY Register (IOMUXC_LPSPI4_SDI_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 524h offset = 401F_8524h



IOMUXC_LPSPI4_SDI_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: lpspi4, In Pin: lpspi_sdi 0 **GPIO_B0_01_ALT3** — Selecting Pad: GPIO_B0_01 for Mode: ALT3 pin 12 1 **GPIO_B1_05_ALT1** — Selecting Pad: GPIO_B1_05 for Mode: ALT1

11.6.326 LPSPI4_SDO_SELECT_INPUT DAISY Register (IOMUXC_LPSPI4_SDO_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 528h offset = 401F_8528h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

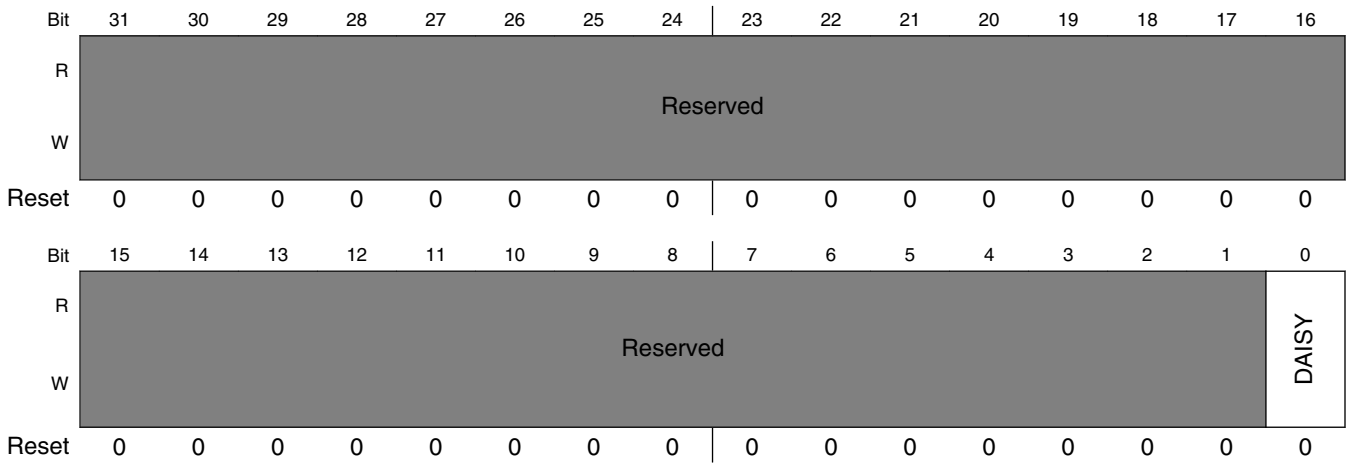
IOMUXC_LPSPI4_SDO_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: lpspi4, In Pin: lpspi_sdo 0 **GPIO_B0_02_ALT3** — Selecting Pad: GPIO_B0_02 for Mode: ALT3 pin 11 1 **GPIO_B1_06_ALT1** — Selecting Pad: GPIO_B1_06 for Mode: ALT1

11.6.327 LPUART2_RX_SELECT_INPUT DAISY Register (IOMUXC_LPUART2_RX_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 52Ch offset = 401F_852Ch



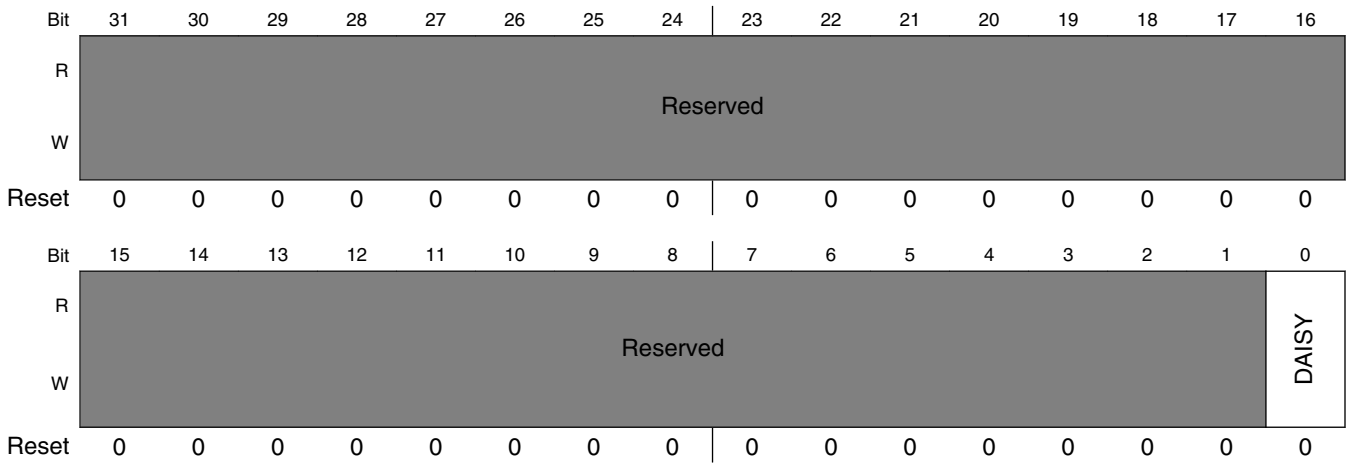
IOMUXC_LPUART2_RX_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: lpuart2, In Pin: lpuart_rx 0 **GPIO_SD_B1_10_ALT2** — Selecting Pad: GPIO_SD_B1_10 for Mode: ALT2 1 **GPIO_AD_B1_03_ALT2** — Selecting Pad: GPIO_AD_B1_03 for Mode: ALT2 pin 15

11.6.328 LPUART2_TX_SELECT_INPUT DAISY Register (IOMUXC_LPUART2_TX_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 530h offset = 401F_8530h



IOMUXC_LPUART2_TX_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: lpuart2, In Pin: lpuart_tx 0 **GPIO_SD_B1_11_ALT2** — Selecting Pad: GPIO_SD_B1_11 for Mode: ALT2 1 **GPIO_AD_B1_02_ALT2** — Selecting Pad: GPIO_AD_B1_02 for Mode: ALT2 pin 14

11.6.329 LPUART3_CTS_B_SELECT_INPUT DAISY Register (IOMUXC_LPUART3_CTS_B_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 534h offset = 401F_8534h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LPUART3_CTS_B_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: lpuart3, In Pin: lpuart_cts_b 0 **GPIO_EMC_15_ALT2** — Selecting Pad: GPIO_EMC_15 for Mode: ALT2 1 **GPIO_AD_B1_04_ALT2** — Selecting Pad: GPIO_AD_B1_04 for Mode: ALT2 pin 40 (T4.1)

11.6.330 LPUART3_RX_SELECT_INPUT DAISY Register (IOMUXC_LPUART3_RX_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 538h offset = 401F_8538h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LPUART3_RX_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: lpuart3, In Pin: lpuart_rx 01 **GPIO_EMC_14_ALT2** — Selecting Pad: GPIO_EMC_14 for Mode: ALT2 00 **GPIO_AD_B1_07_ALT2** — Selecting Pad: GPIO_AD_B1_07 for Mode: ALT2 pin 16 10 **GPIO_B0_09_ALT3** — Selecting Pad: GPIO_B0_09 for Mode: ALT3 pin 45 (MM)

11.6.331 LPUART3_TX_SELECT_INPUT DAISY Register (IOMUXC_LPUART3_TX_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 53Ch offset = 401F_853Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LPUART3_TX_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: lpuart3, In Pin: lpuart_tx 01 **GPIO_EMC_13_ALT2** — Selecting Pad: GPIO_EMC_13 for Mode: ALT2 00 **GPIO_AD_B1_06_ALT2** — Selecting Pad: GPIO_AD_B1_06 for Mode: ALT2 pin 17 10 **GPIO_B0_08_ALT3** — Selecting Pad: GPIO_B0_08 for Mode: ALT3 pin 44 (MM)

11.6.332 LPUART4_RX_SELECT_INPUT DAISY Register (IOMUXC_LPUART4_RX_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 540h offset = 401F_8540h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LPUART4_RX_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: lpuart4, In Pin: lpuart_rx 00 **GPIO_SD_B1_01_ALT4** — Selecting Pad: GPIO_SD_B1_01 for Mode: ALT4 01 **GPIO_EMC_20_ALT2** — Selecting Pad: GPIO_EMC_20 for Mode: ALT2 10 **GPIO_B1_01_ALT2** — Selecting Pad: GPIO_B1_01 for Mode: ALT2 pin 7

11.6.333 LPUART4_TX_SELECT_INPUT DAISY Register (IOMUXC_LPUART4_TX_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 544h offset = 401F_8544h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LPUART4_TX_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: lpuart4, In Pin: lpuart_tx 00 **GPIO_SD_B1_00_ALT4** — Selecting Pad: GPIO_SD_B1_00 for Mode: ALT4 01 **GPIO_EMC_19_ALT2** — Selecting Pad: GPIO_EMC_19 for Mode: ALT2 10 **GPIO_B1_00_ALT2** — Selecting Pad: GPIO_B1_00 for Mode: ALT2 pin 8

11.6.334 LPUART5_RX_SELECT_INPUT DAISY Register (IOMUXC_LPUART5_RX_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 548h offset = 401F_8548h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

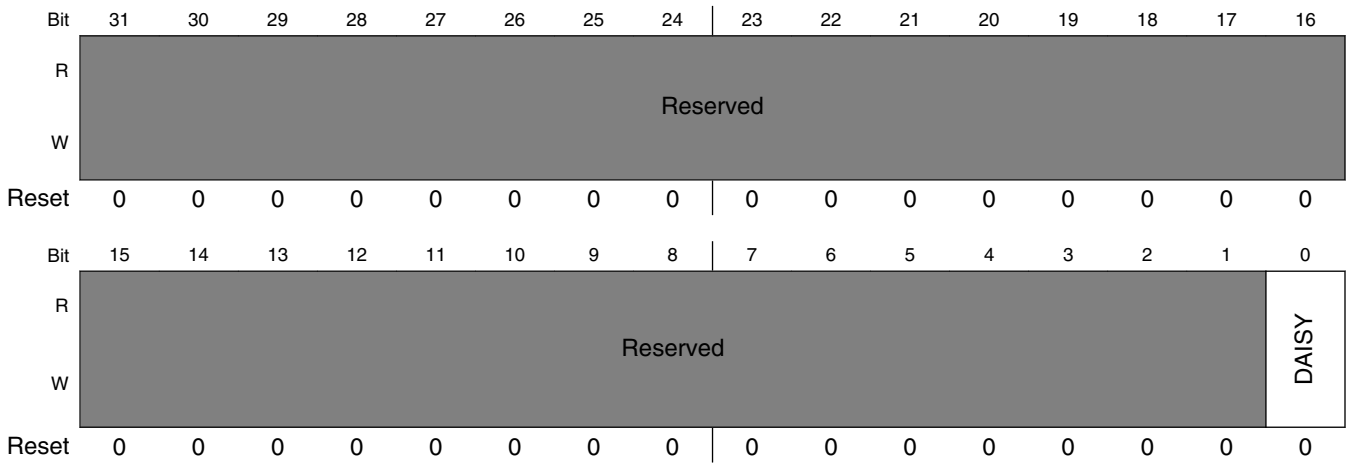
IOMUXC_LPUART5_RX_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: lpuart5, In Pin: lpuart_rx 0 **GPIO_EMC_24_ALT2** — Selecting Pad: GPIO_EMC_24 for Mode: ALT2 1 **GPIO_B1_13_ALT1** — Selecting Pad: GPIO_B1_13 for Mode: ALT1 pin 34 (T4.1)

11.6.335 LPUART5_TX_SELECT_INPUT DAISY Register (IOMUXC_LPUART5_TX_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 54Ch offset = 401F_854Ch



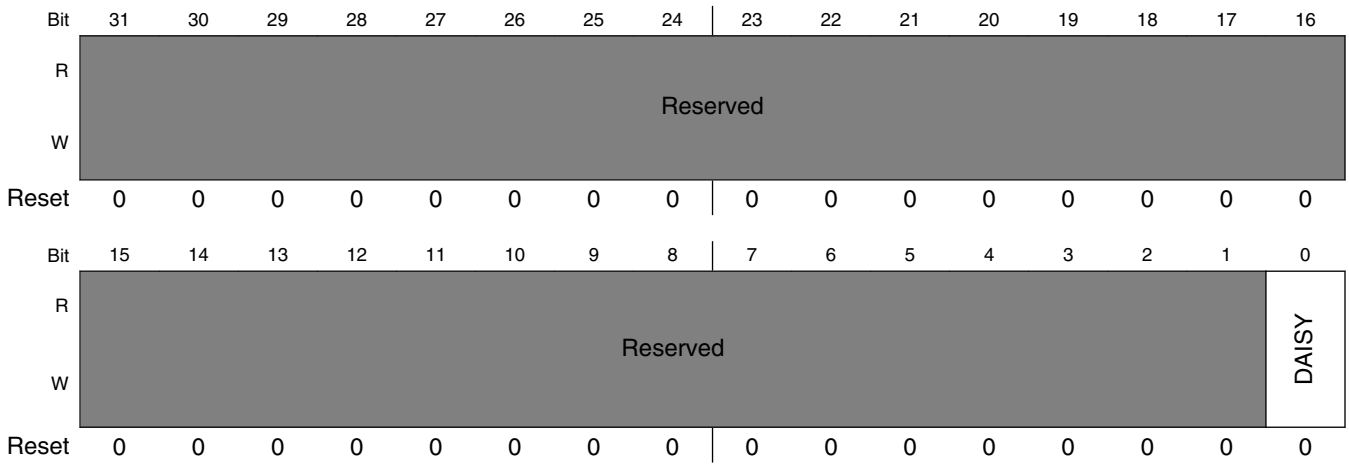
IOMUXC_LPUART5_TX_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: lpuart5, In Pin: lpuart_tx 0 **GPIO_EMC_23_ALT2** — Selecting Pad: GPIO_EMC_23 for Mode: ALT2 1 **GPIO_B1_12_ALT1** — Selecting Pad: GPIO_B1_12 for Mode: ALT1 pin 35 (T4.1)

11.6.336 LPUART6_RX_SELECT_INPUT DAISY Register (IOMUXC_LPUART6_RX_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 550h offset = 401F_8550h



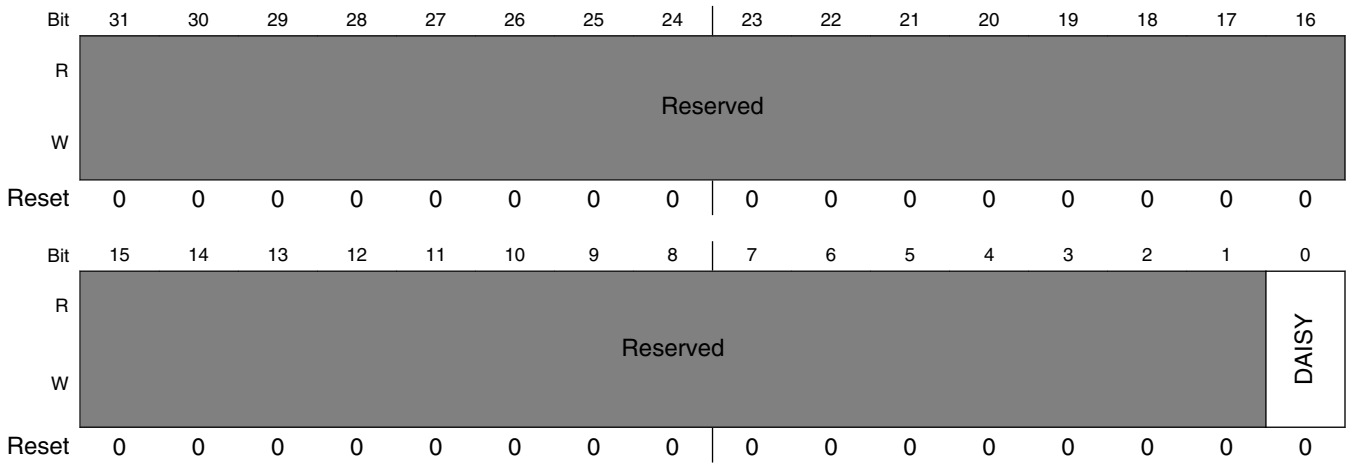
IOMUXC_LPUART6_RX_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: lpuart6, In Pin: lpuart_rx 0 **GPIO_EMC_26_ALT2** — Selecting Pad: GPIO_EMC_26 for Mode: ALT2 1 **GPIO_AD_B0_03_ALT2** — Selecting Pad: GPIO_AD_B0_03 for Mode: ALT2 pin 0

11.6.337 LPUART6_TX_SELECT_INPUT DAISY Register (IOMUXC_LPUART6_TX_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 554h offset = 401F_8554h



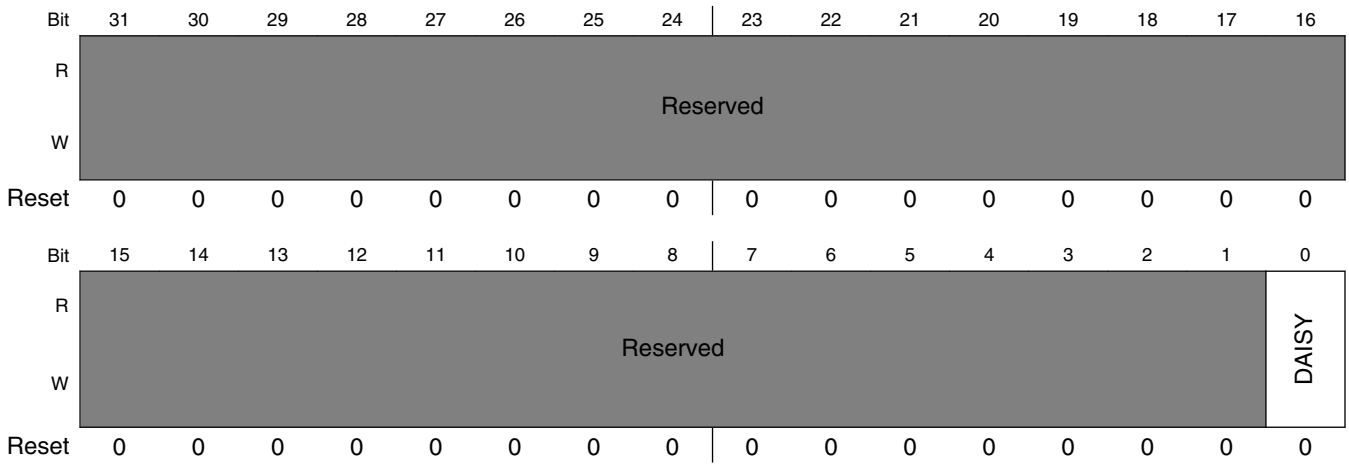
IOMUXC_LPUART6_TX_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: lpuart6, In Pin: lpuart_tx 0 **GPIO_EMC_25_ALT2** — Selecting Pad: GPIO_EMC_25 for Mode: ALT2 1 **GPIO_AD_B0_02_ALT2** — Selecting Pad: GPIO_AD_B0_02 for Mode: ALT2 pin 1

11.6.338 LPUART7_RX_SELECT_INPUT DAISY Register (IOMUXC_LPUART7_RX_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 558h offset = 401F_8558h



IOMUXC_LPUART7_RX_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: lpuart7, In Pin: lpuart_rx 0 **GPIO_SD_B1_09_ALT2** — Selecting Pad: GPIO_SD_B1_09 for Mode: ALT2 1 **GPIO_EMC_32_ALT2** — Selecting Pad: GPIO_EMC_32 for Mode: ALT2 pin 28

11.6.339 LPUART7_TX_SELECT_INPUT DAISY Register (IOMUXC_LPUART7_TX_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 55Ch offset = 401F_855Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LPUART7_TX_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: lpuart7, In Pin: lpuart_tx 0 **GPIO_SD_B1_08_ALT2** — Selecting Pad: GPIO_SD_B1_08 for Mode: ALT2 1 **GPIO_EMC_31_ALT2** — Selecting Pad:GPIO_EMC_31 for Mode: ALT2 pin 29

11.6.340 LPUART8_RX_SELECT_INPUT DAISY Register (IOMUXC_LPUART8_RX_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 560h offset = 401F_8560h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_LPUART8_RX_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: lpuart8, In Pin: lpuart_rx 00 **GPIO_SD_B0_05_ALT2** — Selecting Pad: GPIO_SD_B0_05 for Mode: ALT2 01 **GPIO_AD_B1_11_ALT2** — Selecting Pad: GPIO_AD_B1_11 for Mode: ALT2 pin 21 10 **GPIO_EMC_39_ALT2** — Selecting Pad: GPIO_EMC_39 for Mode: ALT2

11.6.341 LPUART8_TX_SELECT_INPUT DAISY Register (IOMUXC_LPUART8_TX_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 564h offset = 401F_8564h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

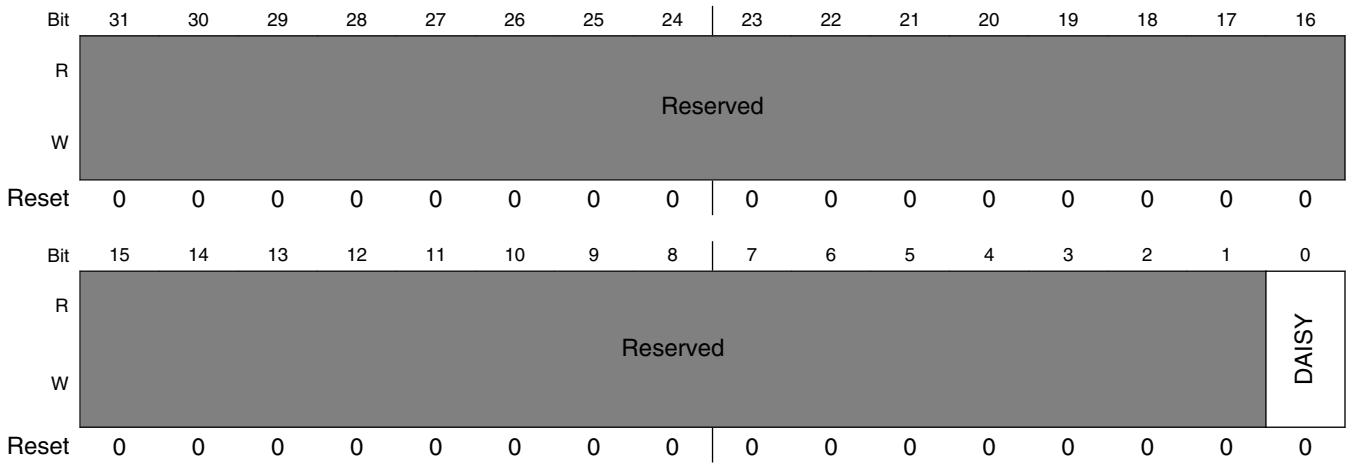
IOMUXC_LPUART8_TX_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: lpuart8, In Pin: lpuart_tx 00 **GPIO_SD_B0_04_ALT2** — Selecting Pad: GPIO_SD_B0_04 for Mode: ALT2 01 **GPIO_AD_B1_10_ALT2** — Selecting Pad: GPIO_AD_B1_10 for Mode: ALT2 pin 20 10 **GPIO_EMC_38_ALT2** — Selecting Pad: GPIO_EMC_38 for Mode: ALT2

11.6.342 NMI_GLUE_NMI_SELECT_INPUT DAISY Register (IOMUXC_NMI_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 568h offset = 401F_8568h



IOMUXC_NMI_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: nmi_glue, In Pin: ipp_ind_nmi 0 **GPIO_AD_B0_12_ALT7** — Selecting Pad: GPIO_AD_B0_12 for Mode: ALT7 pin 24 1 **WAKEUP_ALT7** — Selecting Pad: WAKEUP for Mode: ALT7

11.6.343 QTIMER2_TIMER0_SELECT_INPUT DAISY Register (IOMUXC_QTIMER2_TIMER0_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 56Ch offset = 401F_856Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

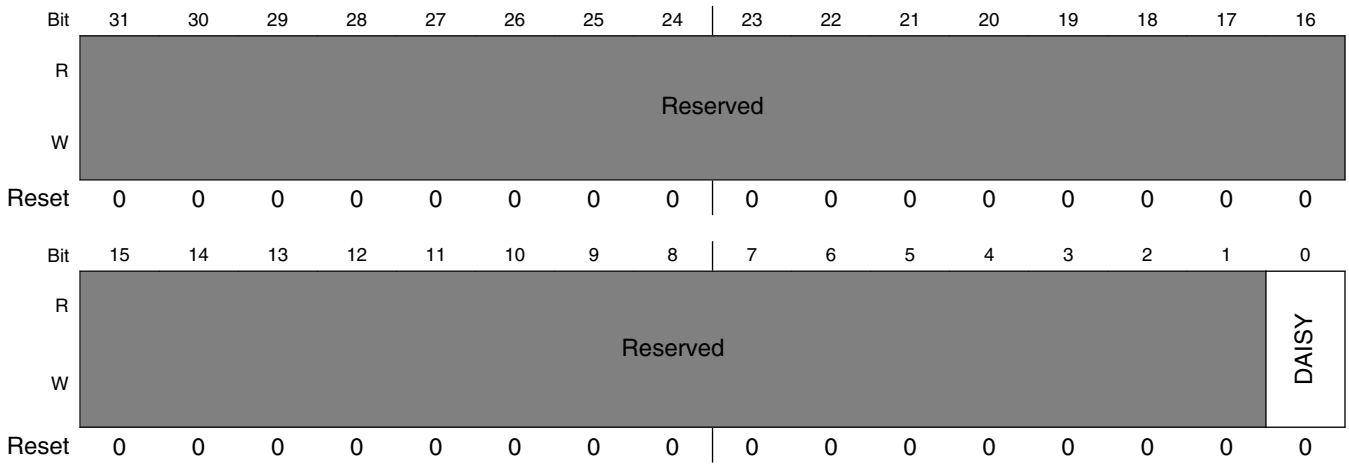
IOMUXC_QTIMER2_TIMER0_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: qtimer2, In Pin: timer0 0 **GPIO_EMC_19_ALT4** — Selecting Pad: GPIO_EMC_19 for Mode: ALT4 1 **GPIO_B0_03_ALT1** — Selecting Pad: GPIO_B0_03 for Mode: ALT1 pin 13

11.6.344 QTIMER2_TIMER1_SELECT_INPUT DAISY Register (IOMUXC_QTIMER2_TIMER1_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 570h offset = 401F_8570h



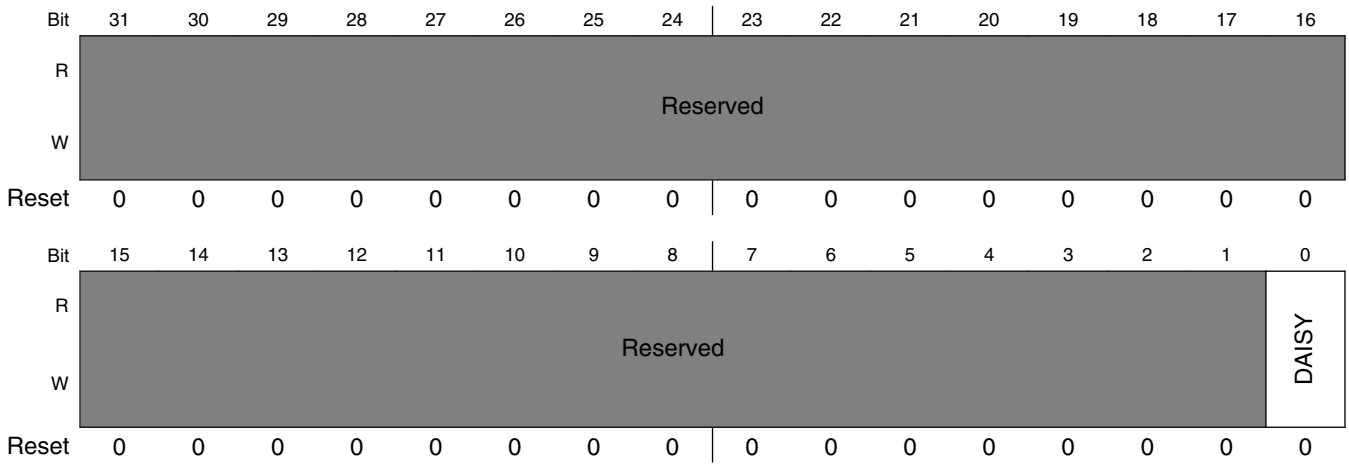
IOMUXC_QTIMER2_TIMER1_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: qtimer2, In Pin: timer1 0 **GPIO_EMC_20_ALT4** — Selecting Pad: GPIO_EMC_20 for Mode: ALT4 1 **GPIO_B0_04_ALT1** — Selecting Pad: GPIO_B0_04 for Mode: ALT1 pin 40 (MM)

11.6.345 QTIMER2_TIMER2_SELECT_INPUT DAISY Register (IOMUXC_QTIMER2_TIMER2_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 574h offset = 401F_8574h



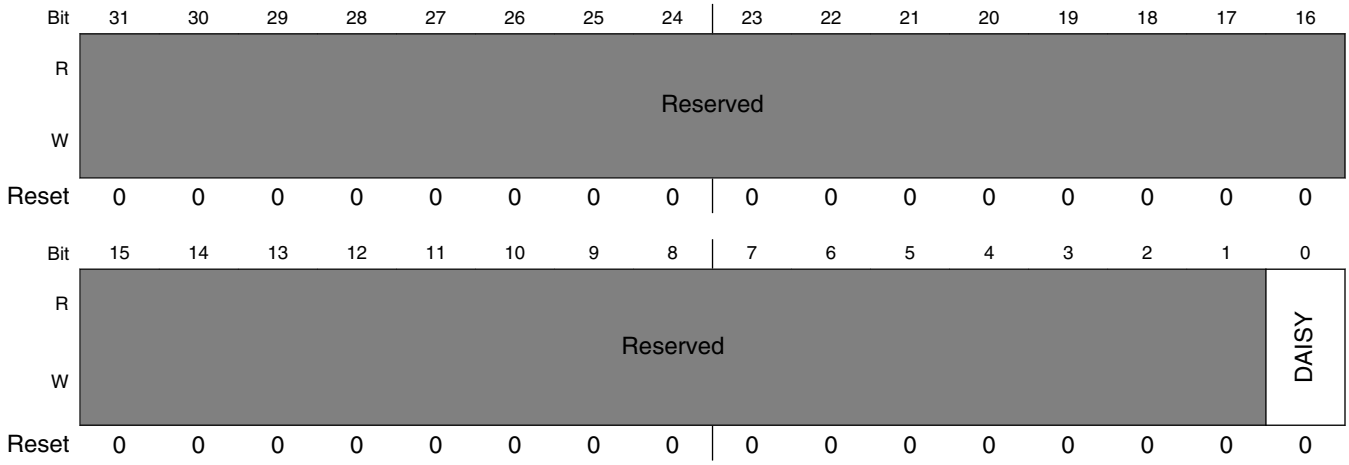
IOMUXC_QTIMER2_TIMER2_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: qtimer2, In Pin: timer2 0 **GPIO_EMC_21_ALT4** — Selecting Pad: GPIO_EMC_21 for Mode: ALT4 1 **GPIO_B0_05_ALT1** — Selecting Pad: GPIO_B0_05 for Mode: ALT1 pin 41 (MM)

11.6.346 QTIMER2_TIMER3_SELECT_INPUT DAISY Register (IOMUXC_QTIMER2_TIMER3_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 578h offset = 401F_8578h



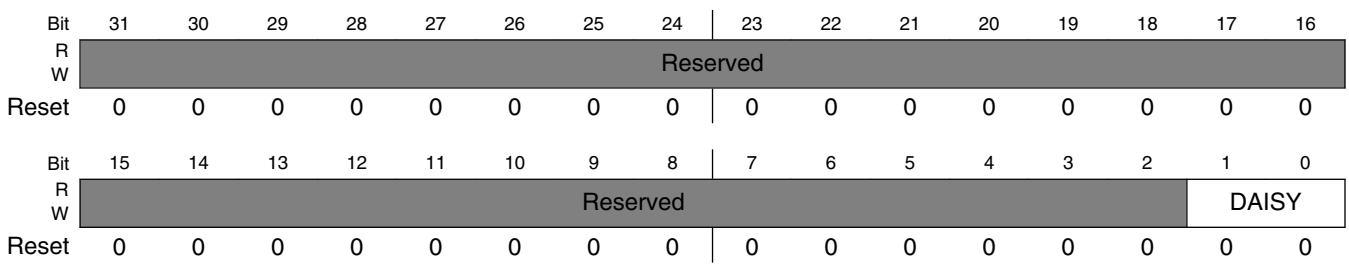
IOMUXC_QTIMER2_TIMER3_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: qtimer2, In Pin: timer3 0 **GPIO_EMC_22_ALT4** — Selecting Pad: GPIO_EMC_22 for Mode: ALT4 1 **GPIO_B1_09_ALT1** — Selecting Pad: GPIO_B1_09 for Mode: ALT1

11.6.347 QTIMER3_TIMER0_SELECT_INPUT DAISY Register (IOMUXC_QTIMER3_TIMER0_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 57Ch offset = 401F_857Ch



IOMUXC_QTIMER3_TIMER0_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: qtimer3, In Pin: timer0 00 **GPIO_EMC_15_ALT4** — Selecting Pad: GPIO_EMC_15 for Mode: ALT4 01 **GPIO_AD_B1_00_ALT1** — Selecting Pad: GPIO_AD_B1_00 for Mode: ALT1 pin 19 10 **GPIO_B0_06_ALT1** — Selecting Pad: GPIO_B0_06 for Mode: ALT1 pin 42 (MM)

11.6.348 QTIMER3_TIMER1_SELECT_INPUT DAISY Register (IOMUXC_QTIMER3_TIMER1_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 580h offset = 401F_8580h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_QTIMER3_TIMER1_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: qtimer3, In Pin: timer1 01 **GPIO_EMC_16_ALT4** — Selecting Pad: GPIO_EMC_16 for Mode: ALT4 00 **GPIO_AD_B1_01_ALT1** — Selecting Pad: GPIO_AD_B1_01 for Mode: ALT1 pin 18 10 **GPIO_B0_07_ALT1** — Selecting Pad: GPIO_B0_07 for Mode: ALT1 pin 43 (MM)

11.6.349 QTIMER3_TIMER2_SELECT_INPUT DAISY Register (IOMUXC_QTIMER3_TIMER2_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 584h offset = 401F_8584h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_QTIMER3_TIMER2_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: qtimer3, In Pin: timer2 00 **GPIO_EMC_17_ALT4** — Selecting Pad: GPIO_EMC_17 for Mode: ALT4 01 **GPIO_AD_B1_02_ALT1** — Selecting Pad: GPIO_AD_B1_02 for Mode: ALT1 pin 14 10 **GPIO_B0_08_ALT1** — Selecting Pad: GPIO_B0_08 for Mode: ALT1 pin 44 (MM)

11.6.350 QTIMER3_TIMER3_SELECT_INPUT DAISY Register (IOMUXC_QTIMER3_TIMER3_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 588h offset = 401F_8588h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_QTIMER3_TIMER3_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: qtimer3, In Pin: timer3 00 **GPIO_EMC_18_ALT4** — Selecting Pad: GPIO_EMC_18 for Mode: ALT4 01 **GPIO_AD_B1_03_ALT1** — Selecting Pad: GPIO_AD_B1_03 for Mode: ALT1 pin 15 10 **GPIO_B1_10_ALT1** — Selecting Pad: GPIO_B1_10 for Mode: ALT1

11.6.351 SAI1_MCLK2_SELECT_INPUT DAISY Register (IOMUXC_SAI1_MCLK2_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 58Ch offset = 401F_858Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_SAI1_MCLK2_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: sai1, In Pin:sai_mclk2 00 **GPIO_SD_B1_03_ALT3** — Selecting Pad: GPIO_SD_B1_03 for Mode: ALT3 01 **GPIO_AD_B1_09_ALT3** — Selecting Pad: GPIO_AD_B1_09 for Mode: ALT3 pin 23 10 **GPIO_B0_13_ALT3** — Selecting Pad: GPIO_B0_13 for Mode: ALT3

11.6.352 SAI1_RX_BCLK_SELECT_INPUT DAISY Register (IOMUXC_SAI1_RX_BCLK_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 590h offset = 401F_8590h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_SAI1_RX_BCLK_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: sai1, In Pin: sai_rx_bclk 00 **GPIO_SD_B1_05_ALT3** — Selecting Pad: GPIO_SD_B1_05 for Mode: ALT3 01 **GPIO_AD_B1_11_ALT3** — Selecting Pad: GPIO_AD_B1_11 for Mode: ALT3 pin 21 10 **GPIO_B0_15_ALT3** — Selecting Pad: GPIO_B0_15 for Mode: ALT3

11.6.353 SAI1_RX_DATA0_SELECT_INPUT DAISY Register (IOMUXC_SAI1_RX_DATA0_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 594h offset = 401F_8594h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_SAI1_RX_DATA0_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: sai1, In Pin: sai_rx_data0 00 **GPIO_SD_B1_06_ALT3** — Selecting Pad: GPIO_SD_B1_06 for Mode: ALT3 01 **GPIO_AD_B1_12_ALT3** — Selecting Pad: GPIO_AD_B1_12 for Mode: ALT3 pin 38 (T4.1) 10 **GPIO_B1_00_ALT3** — Selecting Pad: GPIO_B1_00 for Mode: ALT3 pin 8

11.6.354 SAI1_RX_DATA1_SELECT_INPUT DAISY Register (IOMUXC_SAI1_RX_DATA1_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 598h offset = 401F_8598h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

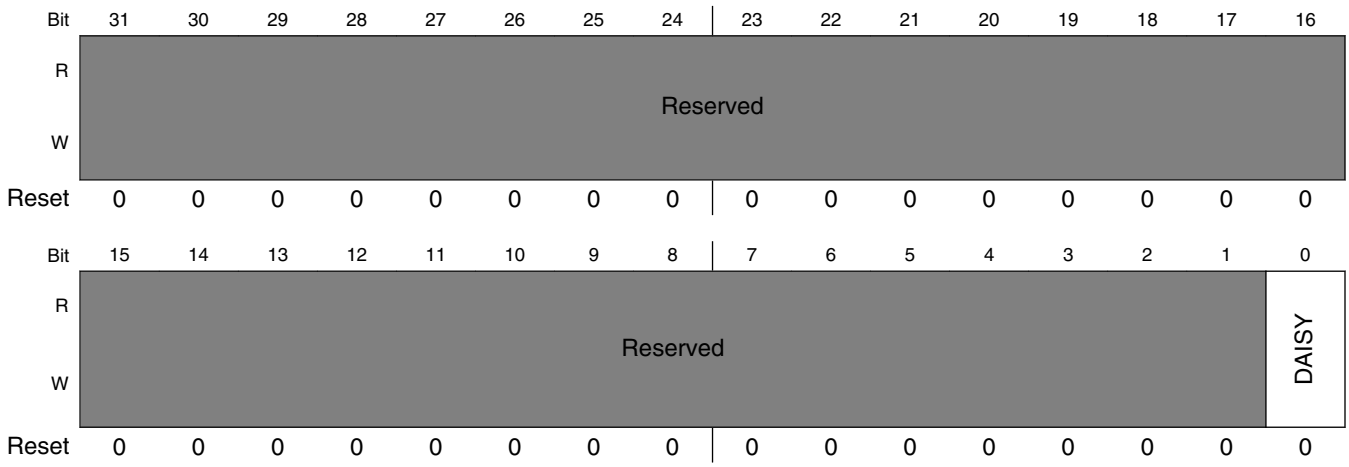
IOMUXC_SAI1_RX_DATA1_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: sai1, In Pin: sai_rx_data1 0 **GPIO_SD_B1_00_ALT3** — Selecting Pad: GPIO_SD_B1_00 for Mode: ALT3 1 **GPIO_B0_10_ALT3** — Selecting Pad: GPIO_B0_10 for Mode: ALT3 pin 6

11.6.355 SAI1_RX_DATA2_SELECT_INPUT DAISY Register (IOMUXC_SAI1_RX_DATA2_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 59Ch offset = 401F_859Ch



IOMUXC_SAI1_RX_DATA2_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: sai1, In Pin: sai_rx_data2 0 **GPIO_SD_B1_01_ALT3** — Selecting Pad: GPIO_SD_B1_01 for Mode: ALT3 1 **GPIO_B0_11_ALT3** — Selecting Pad: GPIO_B0_11 for Mode: ALT3 pin 9

11.6.356 SAI1_RX_DATA3_SELECT_INPUT DAISY Register (IOMUXC_SAI1_RX_DATA3_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 5A0h offset = 401F_85A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_SAI1_RX_DATA3_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: sai1, In Pin: sai_rx_data3 0 **GPIO_SD_B1_02_ALT3** — Selecting Pad: GPIO_SD_B1_02 for Mode: ALT3 1 **GPIO_B0_12_ALT3** — Selecting Pad: GPIO_B0_12 for Mode: ALT3 pin 32

11.6.357 SAI1_RX_SYNC_SELECT_INPUT DAISY Register (IOMUXC_SAI1_RX_SYNC_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 5A4h offset = 401F_85A4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_SAI1_RX_SYNC_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: sai1, In Pin: sai_rx_sync 00 **GPIO_SD_B1_04_ALT3** — Selecting Pad: GPIO_SD_B1_04 for Mode: ALT3 01 **GPIO_AD_B1_10_ALT3** — Selecting Pad: GPIO_AD_B1_10 for Mode: ALT3 pin 20 10 **GPIO_B0_14_ALT3** — Selecting Pad: GPIO_B0_14 for Mode: ALT3

11.6.358 SAI1_TX_BCLK_SELECT_INPUT DAISY Register (IOMUXC_SAI1_TX_BCLK_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 5A8h offset = 401F_85A8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_SAI1_TX_BCLK_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: sai1, In Pin: sai_tx_bclk 00 **GPIO_SD_B1_08_ALT3** — Selecting Pad: GPIO_SD_B1_08 for Mode: ALT3 01 **GPIO_AD_B1_14_ALT3** — Selecting Pad: GPIO_AD_B1_14 for Mode: ALT3 pin 26 10 **GPIO_B1_02_ALT3** — Selecting Pad: GPIO_B1_02 for Mode: ALT3 pin 36 (T4.1)

11.6.359 SAI1_TX_SYNC_SELECT_INPUT DAISY Register (IOMUXC_SAI1_TX_SYNC_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 5ACh offset = 401F_85ACh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

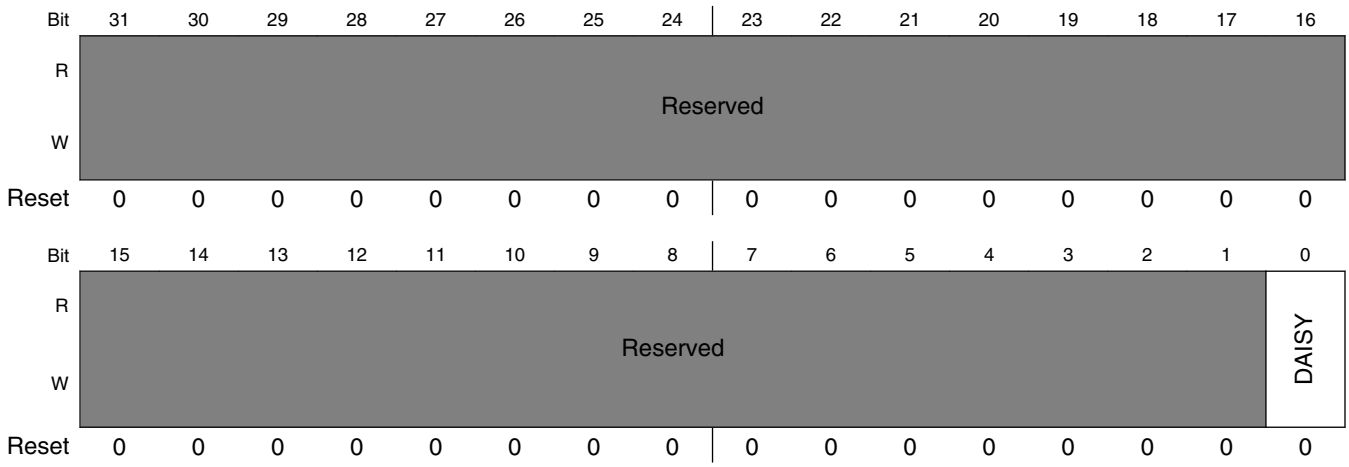
IOMUXC_SAI1_TX_SYNC_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: sai1, In Pin: sai_tx_sync 00 **GPIO_SD_B1_09_ALT3** — Selecting Pad: GPIO_SD_B1_09 for Mode: ALT3 01 **GPIO_AD_B1_15_ALT3** — Selecting Pad: GPIO_AD_B1_15 for Mode: ALT3pin 27 10 **GPIO_B1_03_ALT3** — Selecting Pad: GPIO_B1_03 for Mode: ALT3 pin 37 (T4.1)

11.6.360 SAI2_MCLK2_SELECT_INPUT DAISY Register (IOMUXC_SAI2_MCLK2_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 5B0h offset = 401F_85B0h



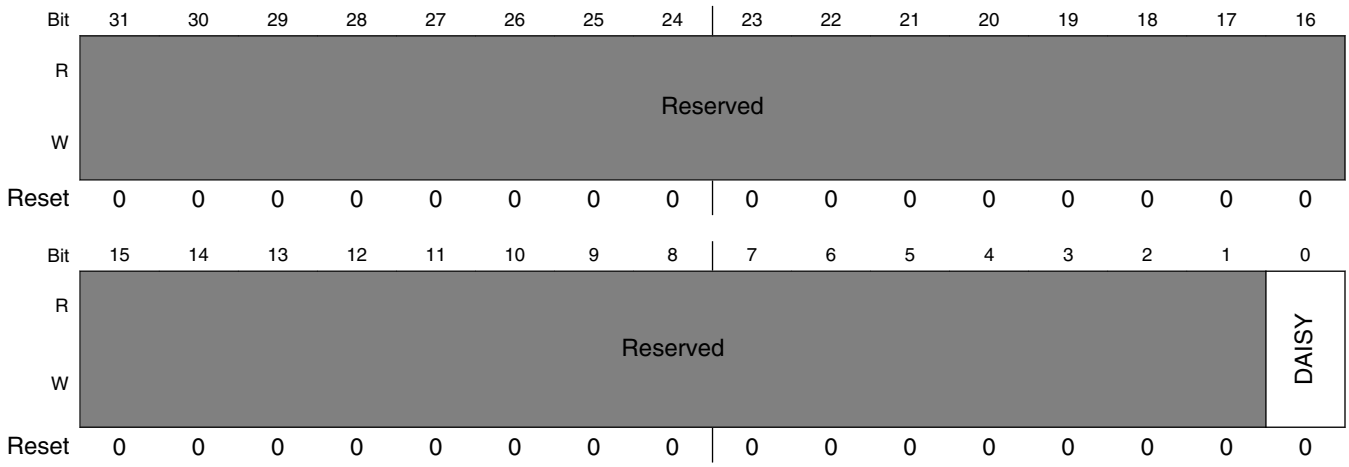
IOMUXC_SAI2_MCLK2_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: sai2, In Pin:sai_mclk2 0 **GPIO_EMC_07_ALT2** — Selecting Pad: GPIO_EMC_07 for Mode: ALT2 pin 33 1 **GPIO_AD_B0_10_ALT3** — Selecting Pad: GPIO_AD_B0_10 for Mode: ALT3

11.6.361 SAI2_RX_BCLK_SELECT_INPUT DAISY Register (IOMUXC_SAI2_RX_BCLK_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 5B4h offset = 401F_85B4h



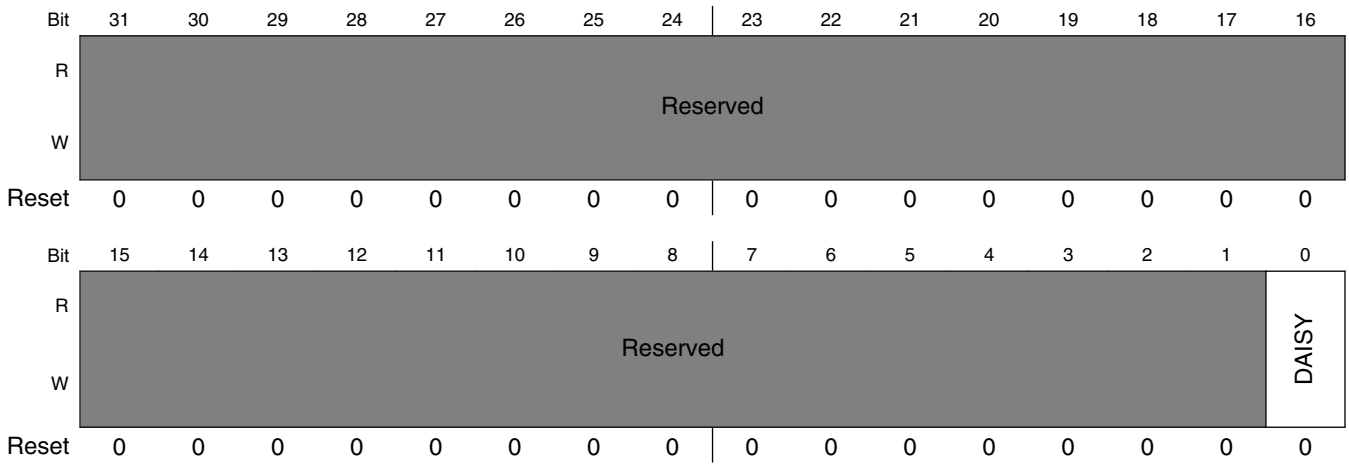
IOMUXC_SAI2_RX_BCLK_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: sai2, In Pin: sai_rx_bclk 0 **GPIO_EMC_10_ALT2** — Selecting Pad: GPIO_EMC_10 for Mode: ALT2 1 **GPIO_AD_B0_06_ALT3** — Selecting Pad: GPIO_AD_B0_06 for Mode: ALT3

11.6.362 SAI2_RX_DATA0_SELECT_INPUT DAISY Register (IOMUXC_SAI2_RX_DATA0_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 5B8h offset = 401F_85B8h



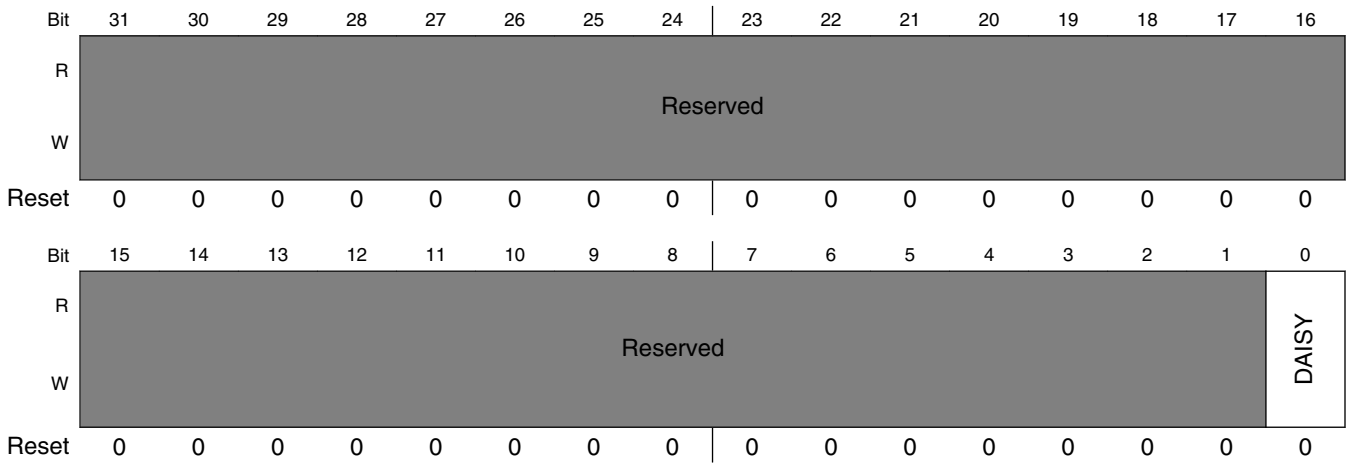
IOMUXC_SAI2_RX_DATA0_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: sai2, In Pin: sai_rx_data0 0 **GPIO_EMC_08_ALT2** — Selecting Pad: GPIO_EMC_08 for Mode: ALT2 pin 5 1 **GPIO_AD_B0_08_ALT3** — Selecting Pad: GPIO_AD_B0_08 for Mode: ALT3

11.6.363 SAI2_RX_SYNC_SELECT_INPUT DAISY Register (IOMUXC_SAI2_RX_SYNC_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 5BCh offset = 401F_85BCh



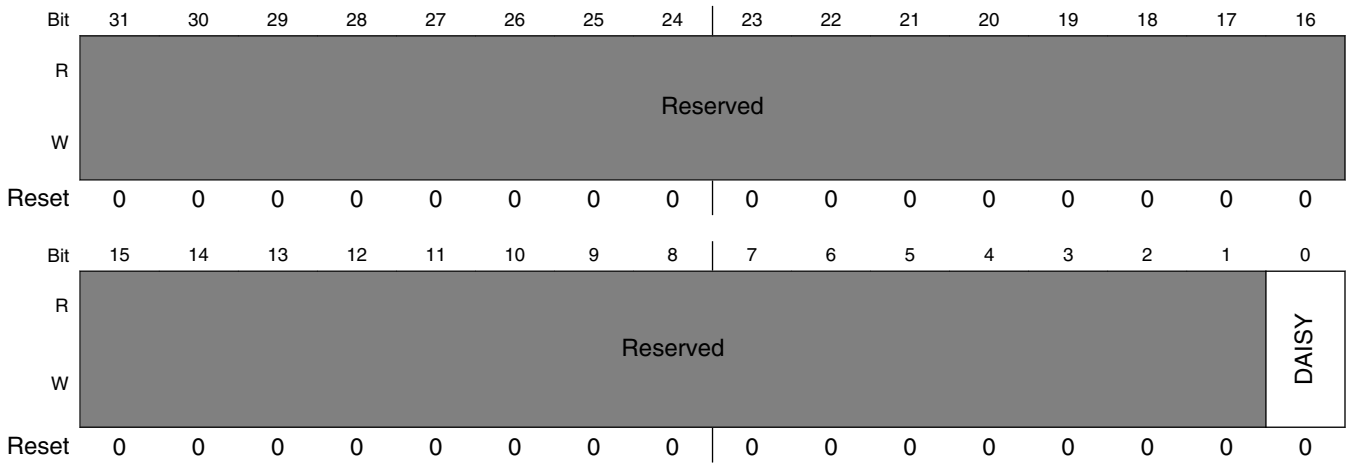
IOMUXC_SAI2_RX_SYNC_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: sai2, In Pin: sai_rx_sync 0 **GPIO_EMC_09_ALT2** — Selecting Pad: GPIO_EMC_09 for Mode: ALT2 1 **GPIO_AD_B0_07_ALT3** — Selecting Pad: GPIO_AD_B0_07 for Mode: ALT3

11.6.364 SAI2_TX_BCLK_SELECT_INPUT DAISY Register (IOMUXC_SAI2_TX_BCLK_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 5C0h offset = 401F_85C0h



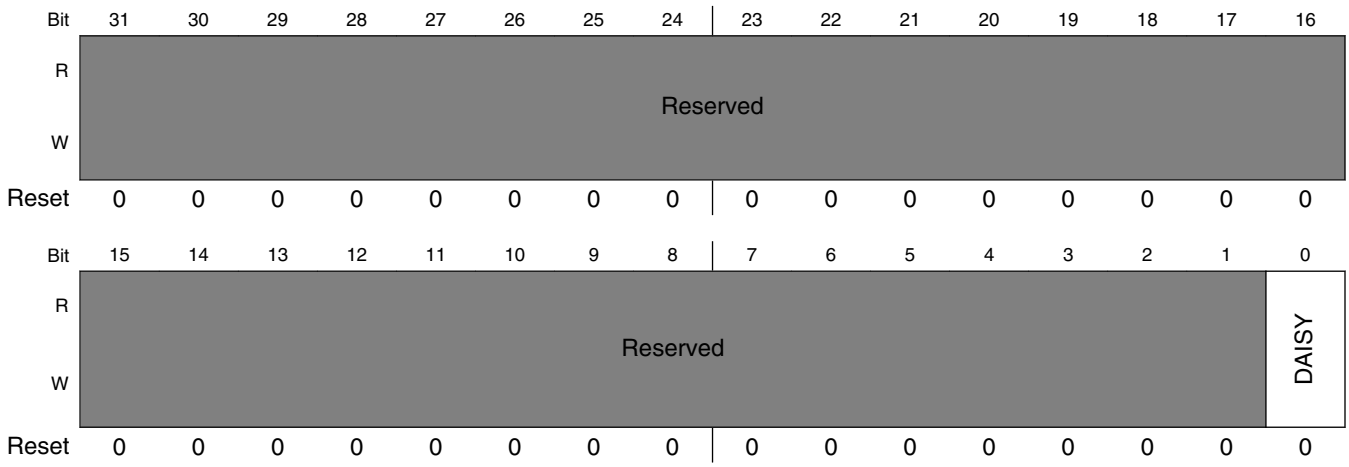
IOMUXC_SAI2_TX_BCLK_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: sai2, In Pin: sai_tx_bclk 0 **GPIO_EMC_06_ALT2** — Selecting Pad: GPIO_EMC_06 for Mode: ALT2 pin 4 1 **GPIO_AD_B0_05_ALT3** — Selecting Pad: GPIO_AD_B0_05 for Mode: ALT3

11.6.365 SAI2_TX_SYNC_SELECT_INPUT DAISY Register (IOMUXC_SAI2_TX_SYNC_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 5C4h offset = 401F_85C4h



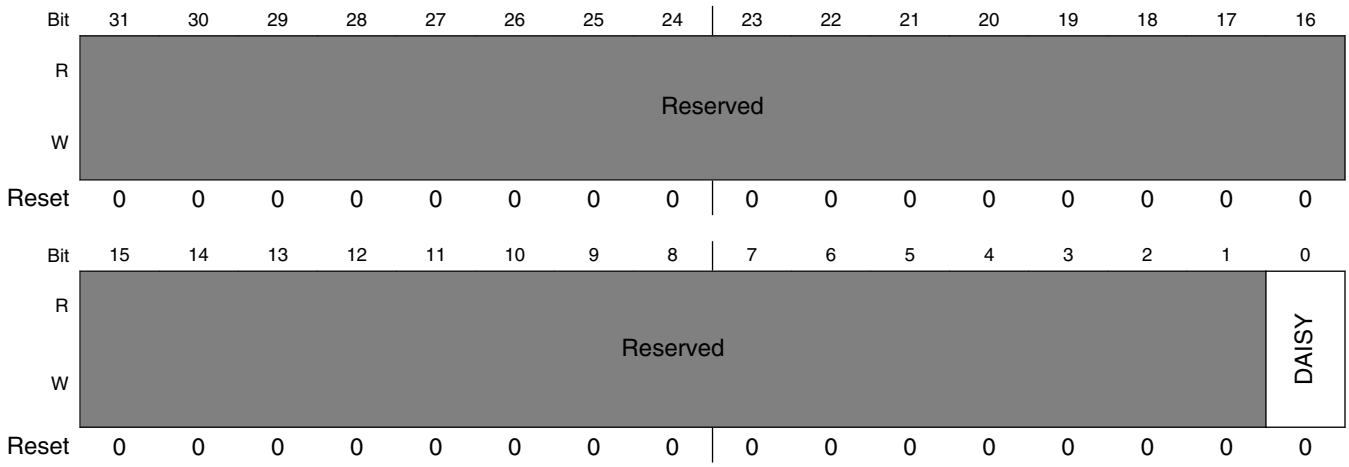
IOMUXC_SAI2_TX_SYNC_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: sai2, In Pin: sai_tx_sync 0 **GPIO_EMC_05_ALT2** — Selecting Pad: GPIO_EMC_05 for Mode: ALT2 pin 3 1 **GPIO_AD_B0_04_ALT3** — Selecting Pad: GPIO_AD_B0_04 for Mode: ALT3

11.6.366 SPDIF_IN_SELECT_INPUT DAISY Register (IOMUXC_SPDIF_IN_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 5C8h offset = 401F_85C8h



IOMUXC_SPDIF_IN_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: spdif, In Pin: spdif_in 1 **GPIO_EMC_16_ALT3** — Selecting Pad: GPIO_EMC_16 for Mode: ALT3 0 **GPIO_AD_B1_03_ALT3** — Selecting Pad: GPIO_AD_B1_03 for Mode: ALT3 pin 15

11.6.367 USB_OTG2_OC_SELECT_INPUT DAISY Register (IOMUXC_USB_OTG2_OC_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 5CCh offset = 401F_85CCh

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_USB_OTG2_OC_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: usb, In Pin: otg2_oc 0 **GPIO_AD_B0_14_ALT0** — Selecting Pad: GPIO_AD_B0_14 for Mode: ALT0 1 **GPIO_EMC_40_ALT3** — Selecting Pad: GPIO_EMC_40 for Mode: ALT3

11.6.368 USB_OTG1_OC_SELECT_INPUT DAISY Register (IOMUXC_USB_OTG1_OC_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 5D0h offset = 401F_85D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_USB_OTG1_OC_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: usb, In Pin: otg_oc 1 **GPIO_AD_B1_03_ALT0** — Selecting Pad: GPIO_AD_B1_03 for Mode: ALT0 pin 15 0 **GPIO_AD_B0_03_ALT3** — Selecting Pad: GPIO_AD_B0_03 for Mode: ALT3 pin 0

11.6.369 USDHC1_CD_B_SELECT_INPUT DAISY Register (IOMUXC_USDHC1_CD_B_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 5D4h offset = 401F_85D4h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

IOMUXC_USDHC1_CD_B_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: usdhc1, In Pin: cd_b 00 **GPIO_EMC_35_ALT6** — Selecting Pad: GPIO_EMC_35 for Mode: ALT6 01 **GPIO_AD_B1_02_ALT6** — Selecting Pad: GPIO_AD_B1_02 for Mode: ALT6 pin 14 10 **GPIO_B1_12_ALT6** — Selecting Pad: GPIO_B1_12 for Mode: ALT6 pin 35 (T4.1)

11.6.370 USDHC1_WP_SELECT_INPUT DAISY Register (IOMUXC_USDHC1_WP_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 5D8h offset = 401F_85D8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

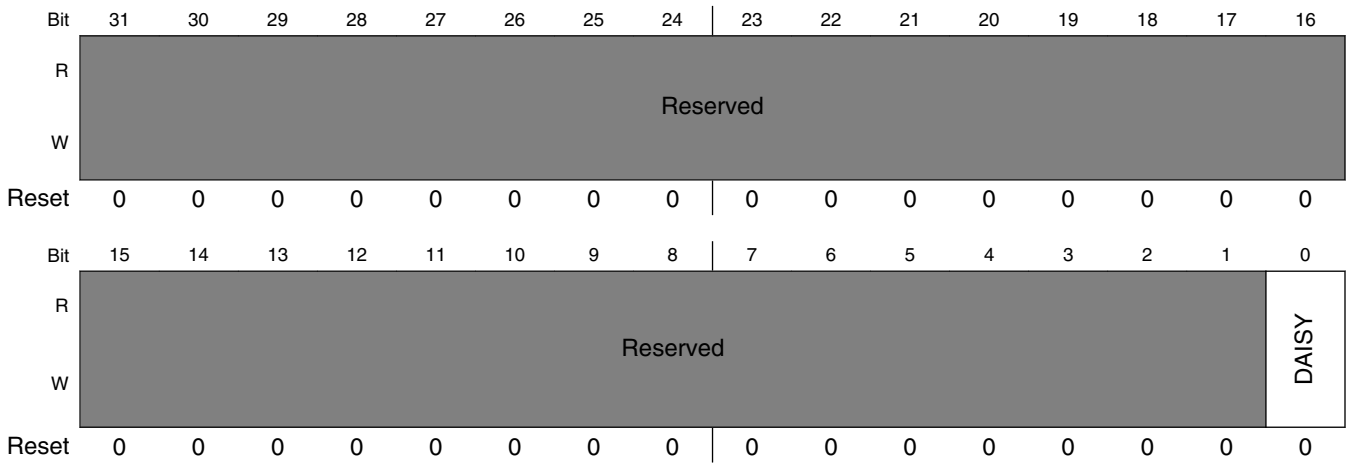
IOMUXC_USDHC1_WP_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: usdhc1, In Pin: wp 00 **GPIO_EMC_12_ALT3** — Selecting Pad: GPIO_EMC_12 for Mode: ALT3 10 **GPIO_AD_B1_00_ALT6** — Selecting Pad:GPIO_AD_B1_00 for Mode: ALT6 pin 19 01 **GPIO_EMC_36_ALT6** — Selecting Pad: GPIO_EMC_36for Mode: ALT6 pin 31 11 **GPIO_B1_13_ALT6** — Selecting Pad: GPIO_B1_13 for Mode: ALT6 pin 34 (T4.1)

11.6.371 USDHC2_CLK_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_CLK_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 5DCh offset = 401F_85DCh



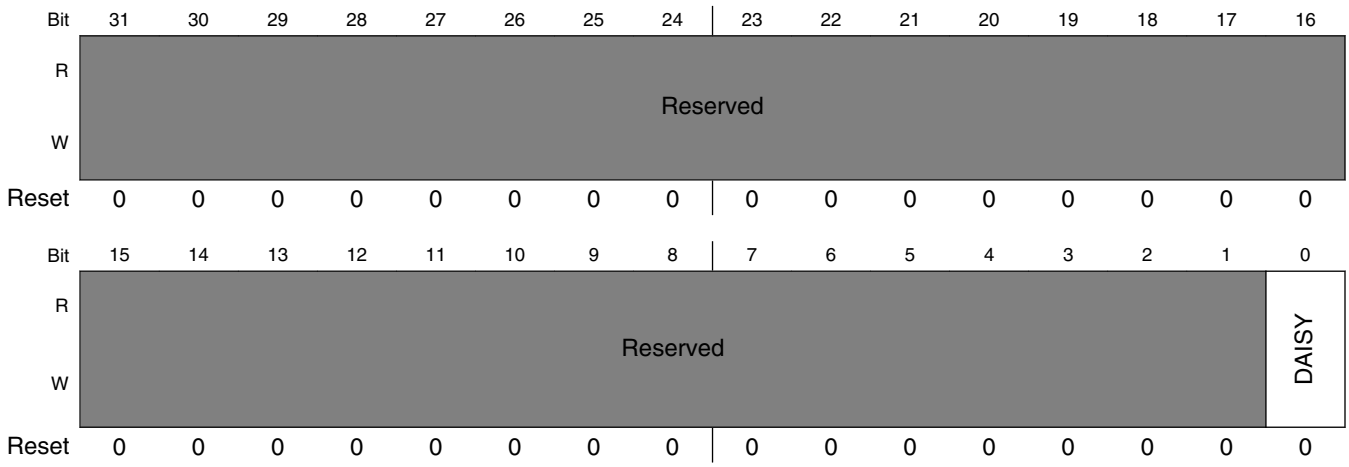
IOMUXC_USDHC2_CLK_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: usdhc2, In Pin: clk 0 **GPIO_SD_B1_04_ALT0** — Selecting Pad: GPIO_SD_B1_04 for Mode: ALT0 1 **GPIO_AD_B1_09_ALT6** — Selecting Pad: GPIO_AD_B1_09 for Mode: ALT6 pin 23

11.6.372 USDHC2_CD_B_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_CD_B_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 5E0h offset = 401F_85E0h



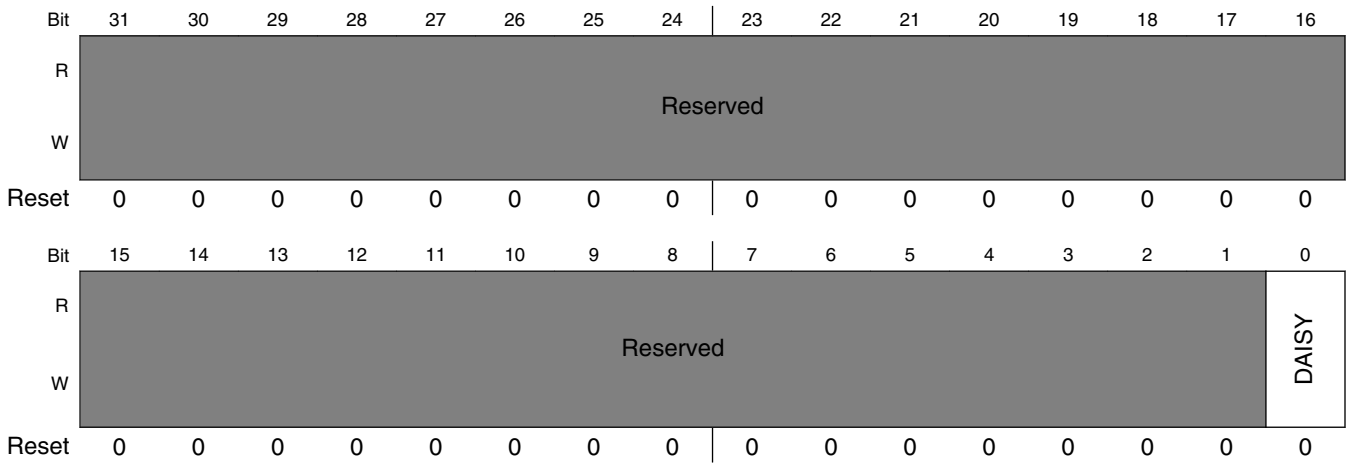
IOMUXC_USDHC2_CD_B_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: usdhc2, In Pin: det 0 **GPIO_AD_B1_03_ALT6** — Selecting Pad:GPIO_AD_B1_03 for Mode: ALT6 pin 15 1 **GPIO_EMC_39_ALT6** — Selecting Pad: GPIO_EMC_39 for Mode: ALT6

11.6.373 USDHC2_CMD_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_CMD_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 5E4h offset = 401F_85E4h



IOMUXC_USDHC2_CMD_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: usdhc2, In Pin: cmd 0 **GPIO_SD_B1_05_ALT0** — Selecting Pad: GPIO_SD_B1_05 for Mode: ALT0 1 **GPIO_AD_B1_08_ALT6** — Selecting Pad: GPIO_AD_B1_08 for Mode: ALT6 pin 22

11.6.374 USDHC2_DATA0_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_DATA0_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 5E8h offset = 401F_85E8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

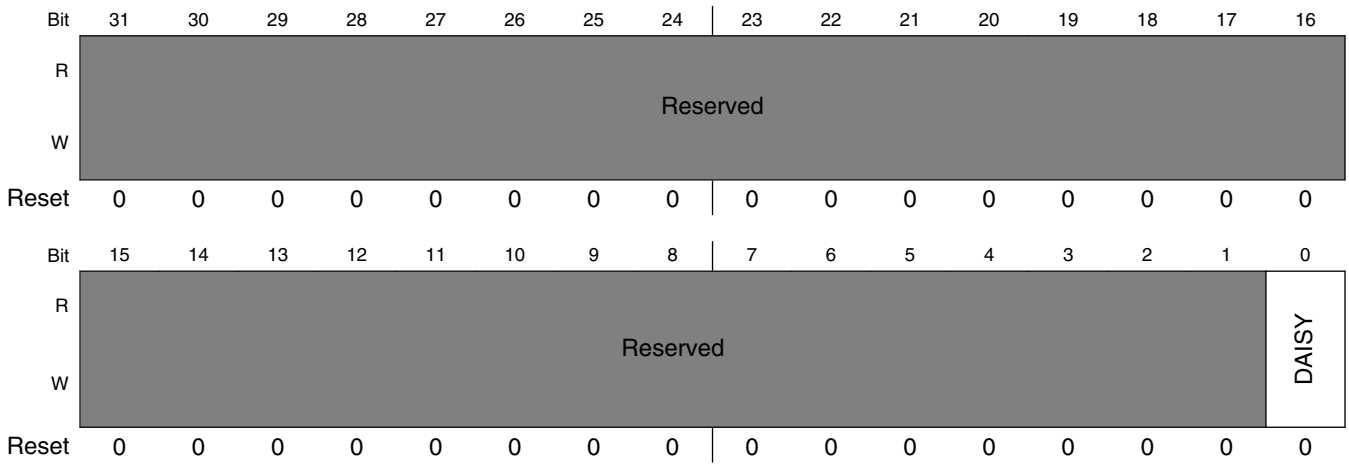
IOMUXC_USDHC2_DATA0_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: usdhc2, In Pin: data0 0 **GPIO_SD_B1_03_ALT0** — Selecting Pad: GPIO_SD_B1_03 for Mode: ALT0 1 **GPIO_AD_B1_04_ALT6** — Selecting Pad:GPIO_AD_B1_04 for Mode: ALT6 pin 40 (T4.1)

11.6.375 USDHC2_DATA1_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_DATA1_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 5ECh offset = 401F_85ECh



IOMUXC_USDHC2_DATA1_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: usdhc2, In Pin: data1 0 **GPIO_SD_B1_02_ALT0** — Selecting Pad: GPIO_SD_B1_02 for Mode: ALT0 1 **GPIO_AD_B1_05_ALT6** — Selecting Pad: GPIO_AD_B1_05 for Mode: ALT6 pin 41 (T4.1)

11.6.376 USDHC2_DATA2_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_DATA2_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 5F0h offset = 401F_85F0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

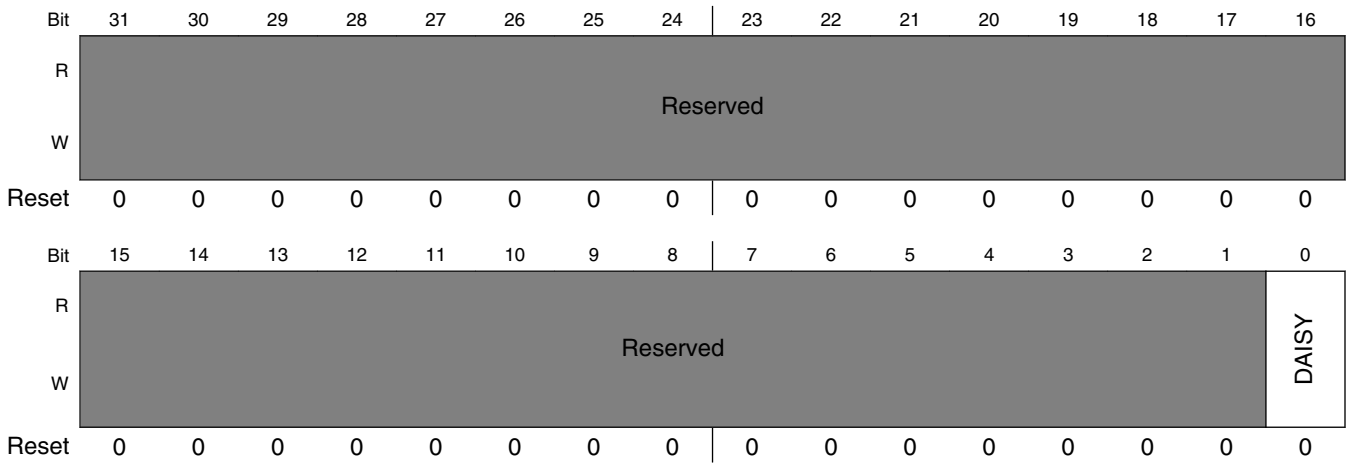
IOMUXC_USDHC2_DATA2_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: usdhc2, In Pin: data2 0 **GPIO_SD_B1_01_ALT0** — Selecting Pad: GPIO_SD_B1_01 for Mode: ALT0 1 **GPIO_AD_B1_06_ALT6** — Selecting Pad: GPIO_AD_B1_06 for Mode: ALT6 pin 17

11.6.377 USDHC2_DATA3_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_DATA3_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 5F4h offset = 401F_85F4h



IOMUXC_USDHC2_DATA3_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: usdhc2, In Pin: data3 0 **GPIO_SD_B1_00_ALT0** — Selecting Pad: GPIO_SD_B1_00 for Mode: ALT0 1 **GPIO_AD_B1_07_ALT6** — Selecting Pad: GPIO_AD_B1_07 for Mode: ALT6 pin 16

11.6.378 USDHC2_DATA4_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_DATA4_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 5F8h offset = 401F_85F8h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

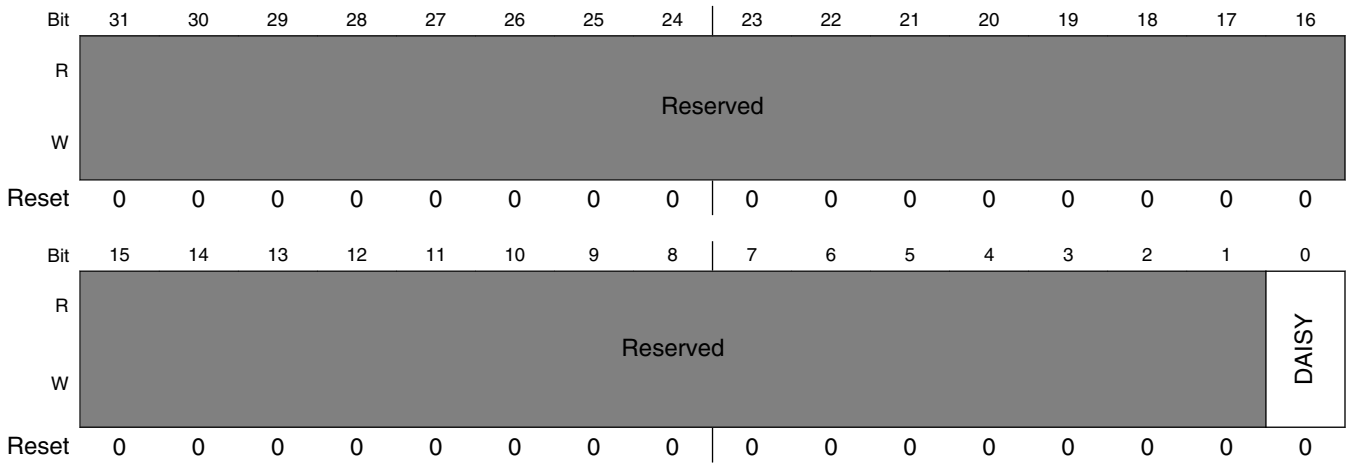
IOMUXC_USDHC2_DATA4_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: usdhc2, In Pin: data4 0 **GPIO_SD_B1_08_ALT0** — Selecting Pad: GPIO_SD_B1_08 for Mode: ALT0 1 **GPIO_AD_B1_12_ALT6** — Selecting Pad: GPIO_AD_B1_12 for Mode: ALT6 pin 38 (T4.1)

11.6.379 USDHC2_DATA5_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_DATA5_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 5FCh offset = 401F_85FCh



IOMUXC_USDHC2_DATA5_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: usdhc2, In Pin: data5 0 **GPIO_SD_B1_09_ALT0** — Selecting Pad: GPIO_SD_B1_09 for Mode: ALT0 1 **GPIO_AD_B1_13_ALT6** — Selecting Pad: GPIO_AD_B1_13 for Mode: ALT6 pin 39 (T4.1)

11.6.380 USDHC2_DATA6_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_DATA6_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 600h offset = 401F_8600h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

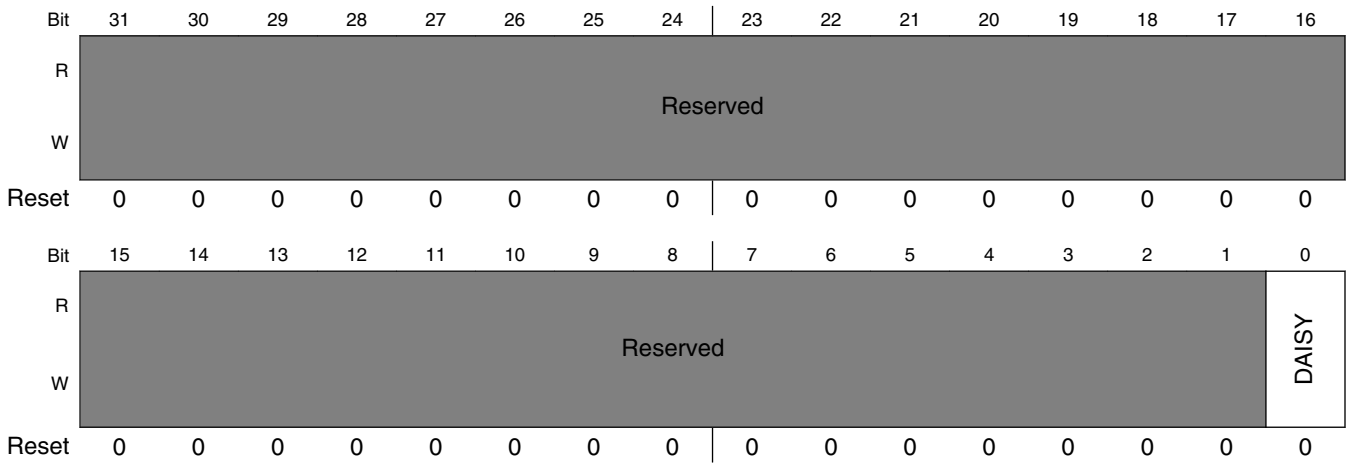
IOMUXC_USDHC2_DATA6_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: usdhc2, In Pin: data6 0 **GPIO_SD_B1_10_ALT0** — Selecting Pad: GPIO_SD_B1_10 for Mode: ALT0 1 **GPIO_AD_B1_14_ALT6** — Selecting Pad: GPIO_AD_B1_14 for Mode: ALT6 pin 26

11.6.381 USDHC2_DATA7_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_DATA7_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 604h offset = 401F_8604h



IOMUXC_USDHC2_DATA7_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: usdhc2, In Pin: data7 0 **GPIO_SD_B1_11_ALT0** — Selecting Pad: GPIO_SD_B1_11 for Mode: ALT0 1 **GPIO_AD_B1_15_ALT6** — Selecting Pad: GPIO_AD_B1_15 for Mode: ALT6 pin 27

11.6.382 USDHC2_WP_SELECT_INPUT DAISY Register (IOMUXC_USDHC2_WP_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 608h offset = 401F_8608h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															DAISY
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

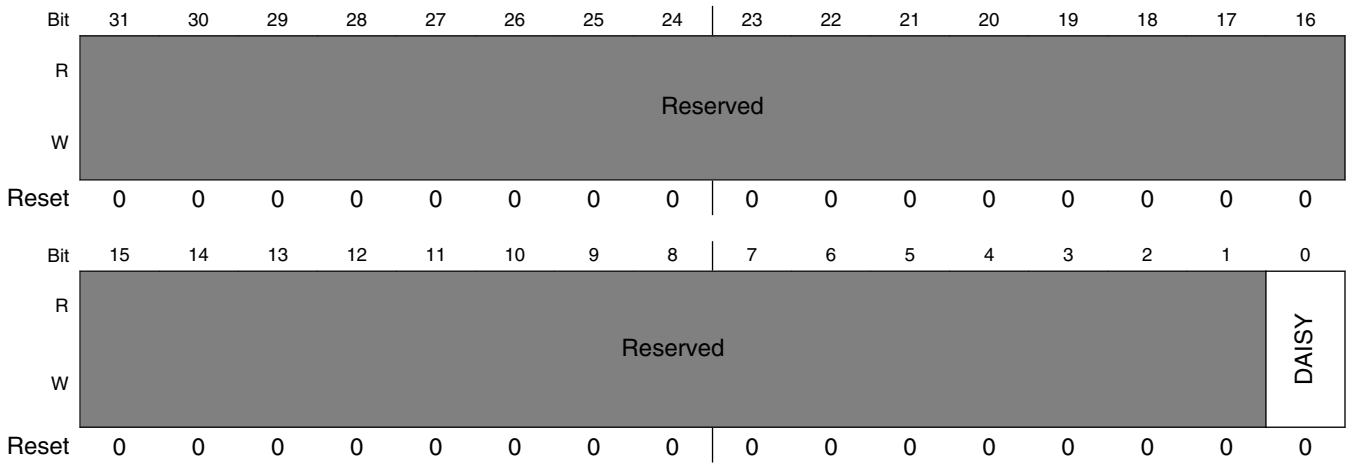
IOMUXC_USDHC2_WP_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: usdhc2, In Pin: wp 1 **GPIO_AD_B1_10_ALT6** — Selecting Pad: GPIO_AD_B1_10 for Mode: ALT6 pin 20 0 **GPIO_EMC_37_ALT6** — Selecting Pad: GPIO_EMC_37 for Mode: ALT6 pin 30

11.6.383 XBAR1_IN02_SELECT_INPUT DAISY Register (IOMUXC_XBAR1_IN02_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 60Ch offset = 401F_860Ch



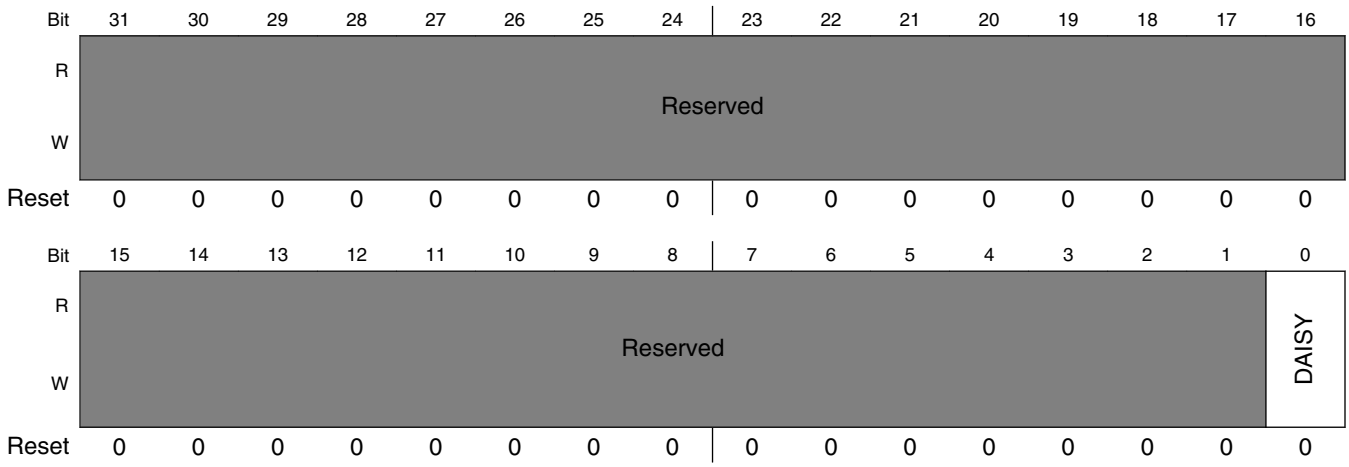
IOMUXC_XBAR1_IN02_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: xbar1, In Pin: xbar_in2 0 **GPIO_EMC_00_ALT3** — Selecting Pad: GPIO_EMC_00 for Mode: ALT3 1 **GPIO_B1_14_ALT3** — Selecting Pad: GPIO_B1_14 for Mode: ALT3

11.6.384 XBAR1_IN03_SELECT_INPUT DAISY Register (IOMUXC_XBAR1_IN03_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 610h offset = 401F_8610h



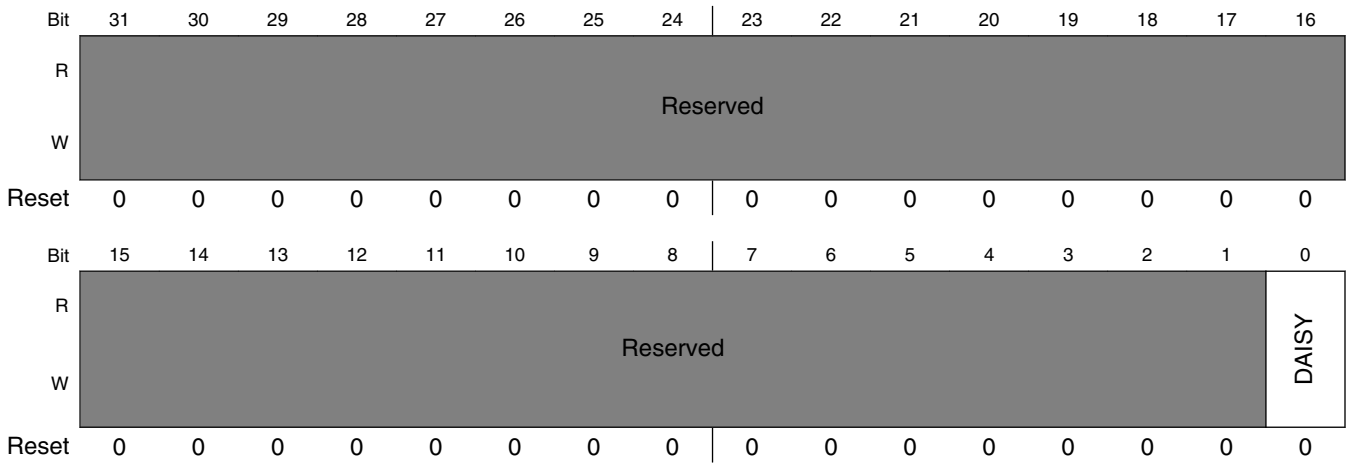
IOMUXC_XBAR1_IN03_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: xbar1, In Pin: xbar_in3 0 **GPIO_EMC_01_ALT3** — Selecting Pad: GPIO_EMC_01 for Mode: ALT3 1 **GPIO_B1_15_ALT3** — Selecting Pad: GPIO_B1_15 for Mode: ALT3

11.6.385 XBAR1_IN04_SELECT_INPUT DAISY Register (IOMUXC_XBAR1_IN04_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 614h offset = 401F_8614h



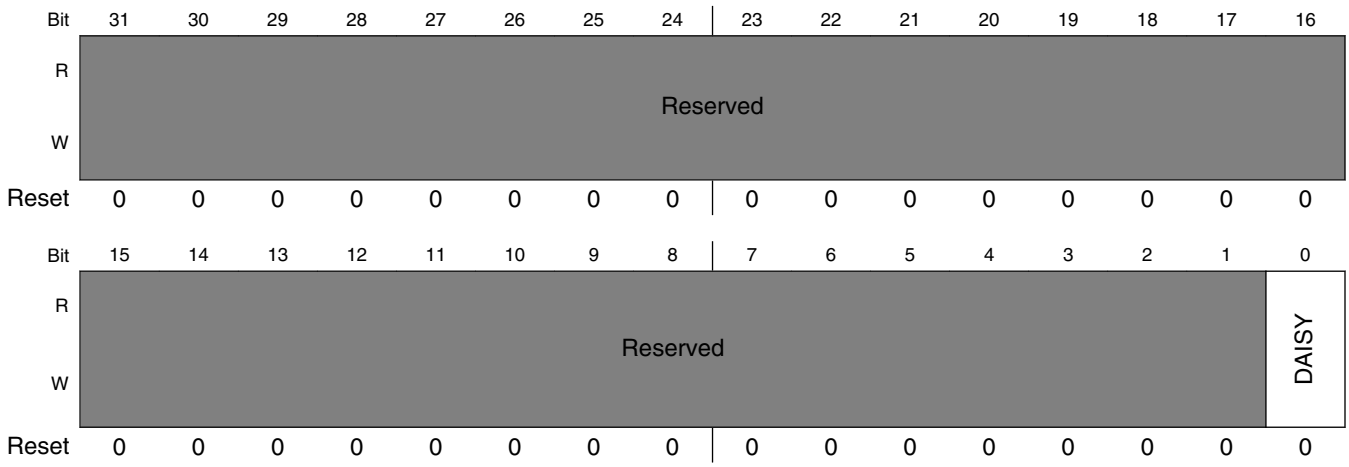
IOMUXC_XBAR1_IN04_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: xbar1, In Pin: xbar_in4 0 **GPIO_EMC_02_ALT3** — Selecting Pad: GPIO_EMC_02 for Mode: ALT3 1 **GPIO_SD_B0_00_ALT3** — Selecting Pad: GPIO_SD_B0_00 for Mode: ALT3

11.6.386 XBAR1_IN05_SELECT_INPUT DAISY Register (IOMUXC_XBAR1_IN05_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 618h offset = 401F_8618h



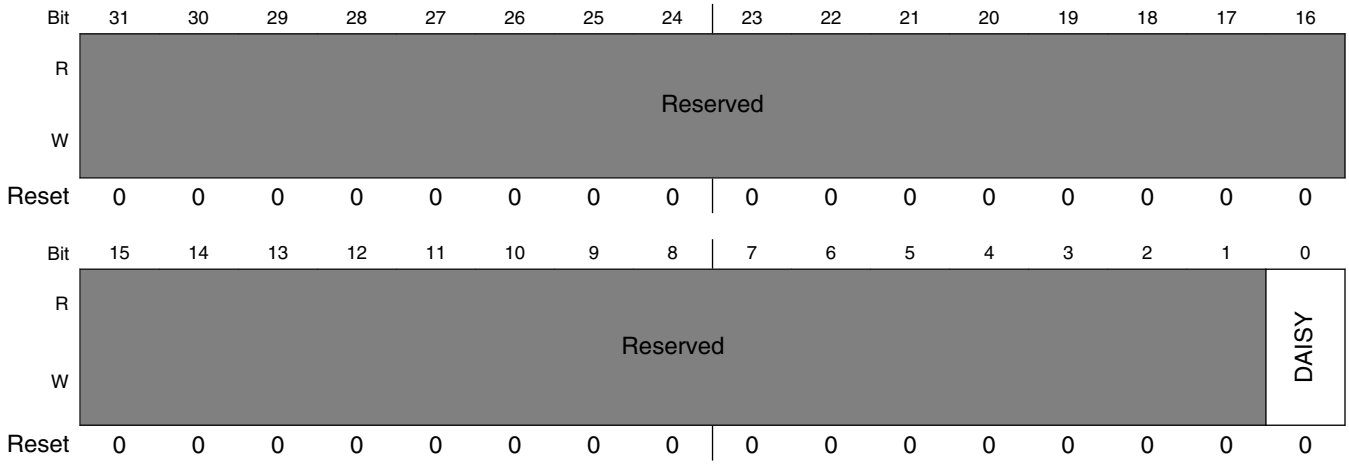
IOMUXC_XBAR1_IN05_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: xbar1, In Pin: xbar_in5 0 **GPIO_EMC_03_ALT3** — Selecting Pad: GPIO_EMC_03 for Mode: ALT3 1 **GPIO_SD_B0_01_ALT3** — Selecting Pad: GPIO_SD_B0_01 for Mode: ALT3

11.6.387 XBAR1_IN06_SELECT_INPUT DAISY Register (IOMUXC_XBAR1_IN06_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 61Ch offset = 401F_861Ch



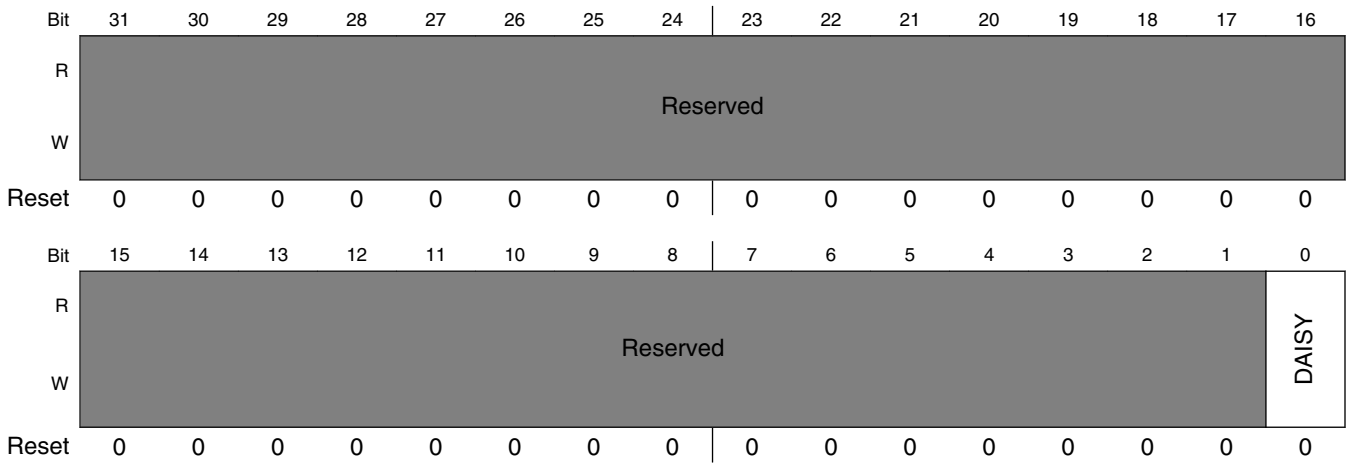
IOMUXC_XBAR1_IN06_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: xbar1, In Pin: xbar_in6 0 **GPIO_EMC_04_ALT3** — Selecting Pad: GPIO_EMC_04 for Mode: ALT3 pin 2 1 **GPIO_SD_B0_02_ALT3** — Selecting Pad: GPIO_SD_B0_02 for Mode: ALT3

11.6.388 XBAR1_IN07_SELECT_INPUT DAISY Register (IOMUXC_XBAR1_IN07_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 620h offset = 401F_8620h



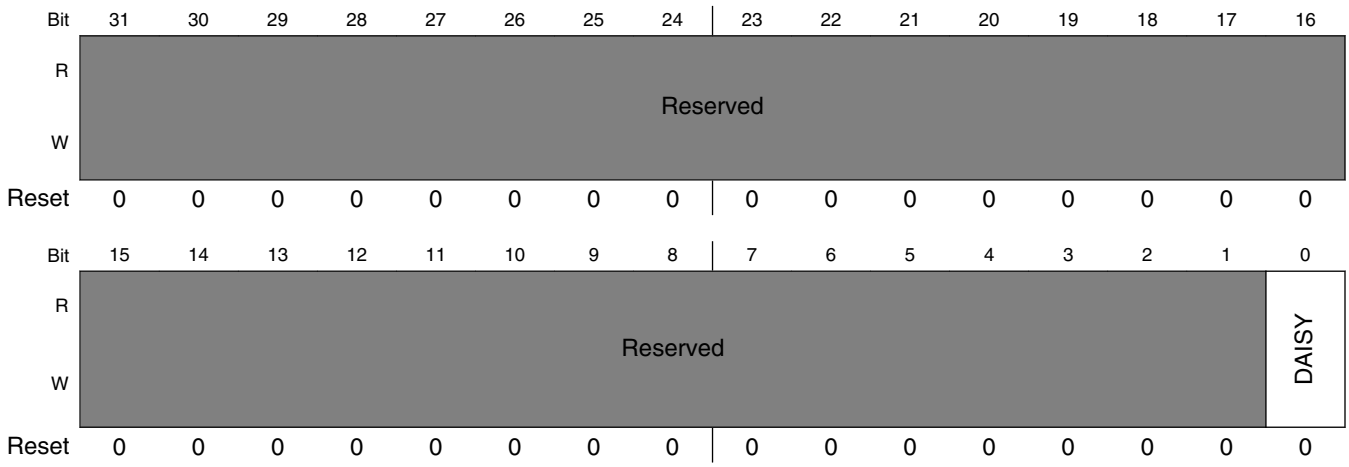
IOMUXC_XBAR1_IN07_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: xbar1, In Pin: xbar_in7 0 **GPIO_EMC_05_ALT3** — Selecting Pad: GPIO_EMC_05 for Mode: ALT3 pin 3 1 **GPIO_SD_B0_03_ALT3** — Selecting Pad: GPIO_SD_B0_03 for Mode: ALT3

11.6.389 XBAR1_IN08_SELECT_INPUT DAISY Register (IOMUXC_XBAR1_IN08_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 624h offset = 401F_8624h



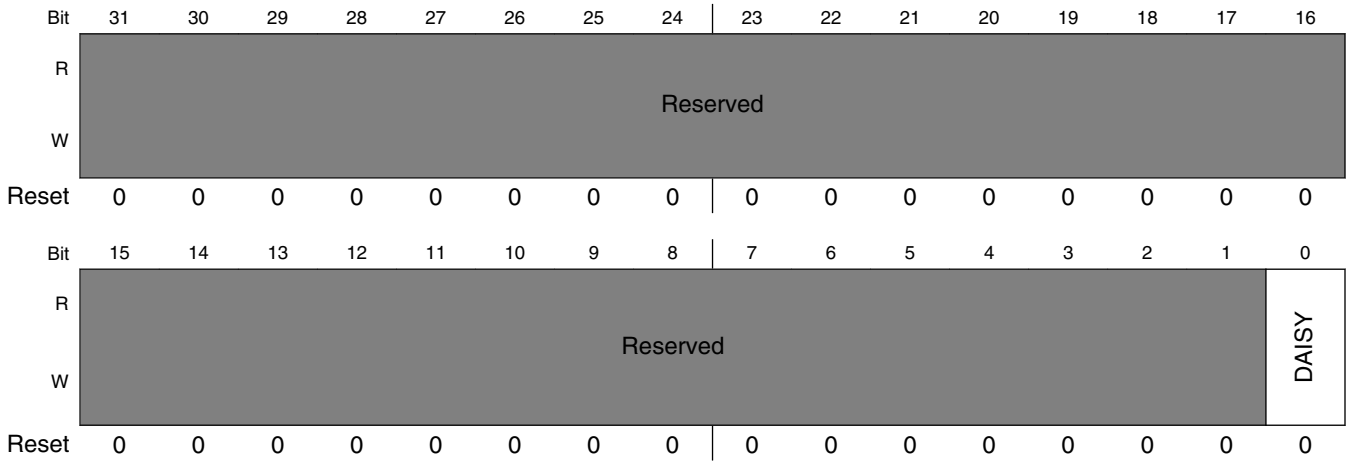
IOMUXC_XBAR1_IN08_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: xbar1, In Pin: xbar_in8 0 **GPIO_EMC_06_ALT3** — Selecting Pad: GPIO_EMC_06 for Mode: ALT3 pin 4 1 **GPIO_SD_B0_04_ALT3** — Selecting Pad: GPIO_SD_B0_04 for Mode: ALT3

11.6.390 XBAR1_IN09_SELECT_INPUT DAISY Register (IOMUXC_XBAR1_IN09_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 628h offset = 401F_8628h



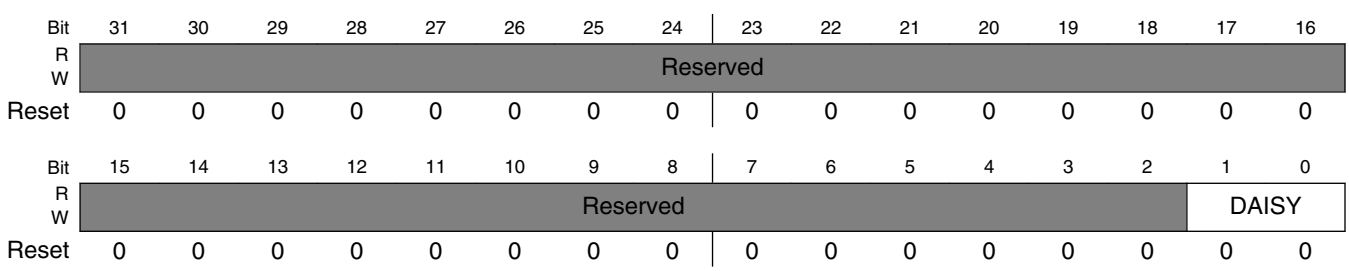
IOMUXC_XBAR1_IN09_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: xbar1, In Pin: xbar_in9 0 **GPIO_EMC_07_ALT3** — Selecting Pad: GPIO_EMC_07 for Mode: ALT3 pin 33 1 **GPIO_SD_B0_05_ALT3** — Selecting Pad: GPIO_SD_B0_05 for Mode: ALT3

11.6.391 XBAR1_IN17_SELECT_INPUT DAISY Register (IOMUXC_XBAR1_IN17_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 62Ch offset = 401F_862Ch



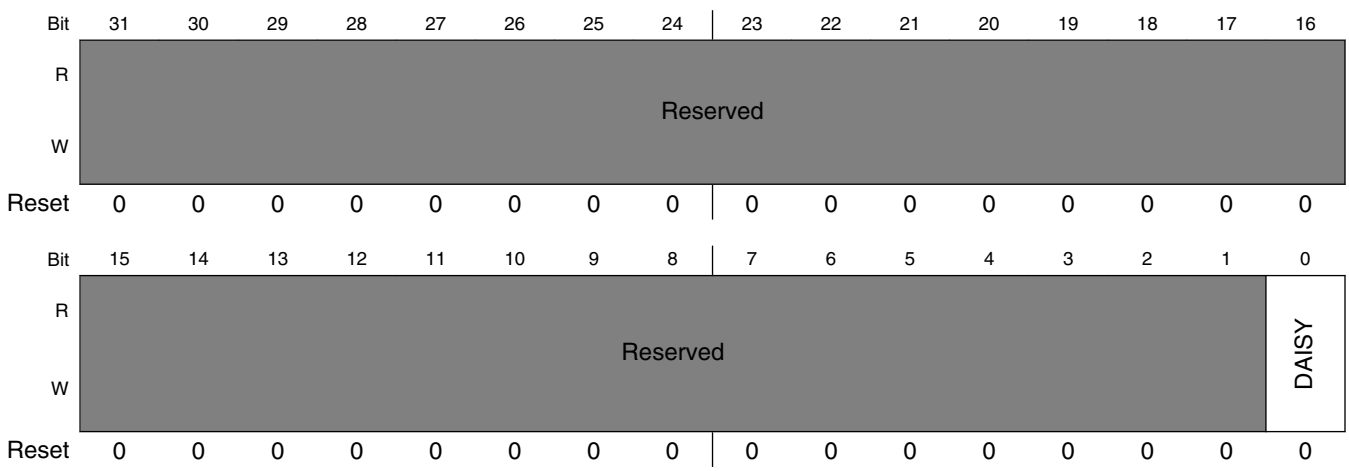
IOMUXC_XBAR1_IN17_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: xbar1, In Pin: xbar_in17 00 **GPIO_EMC_08_ALT3** — Selecting Pad: GPIO_EMC_08 for Mode: ALT3 pin 5 01 **GPIO_AD_B0_03_ALT1** — Selecting Pad: GPIO_AD_B0_03 for Mode: ALT1 pin 0 10 **GPIO_AD_B0_05_ALT6** — Selecting Pad: GPIO_AD_B0_05 for Mode: ALT6 11 **GPIO_B1_03_ALT1** — Selecting Pad: GPIO_B1_03 for Mode: ALT1 pin 37 (T4.1)

11.6.392 XBAR1_IN18_SELECT_INPUT DAISY Register (IOMUXC_XBAR1_IN18_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 630h offset = 401F_8630h



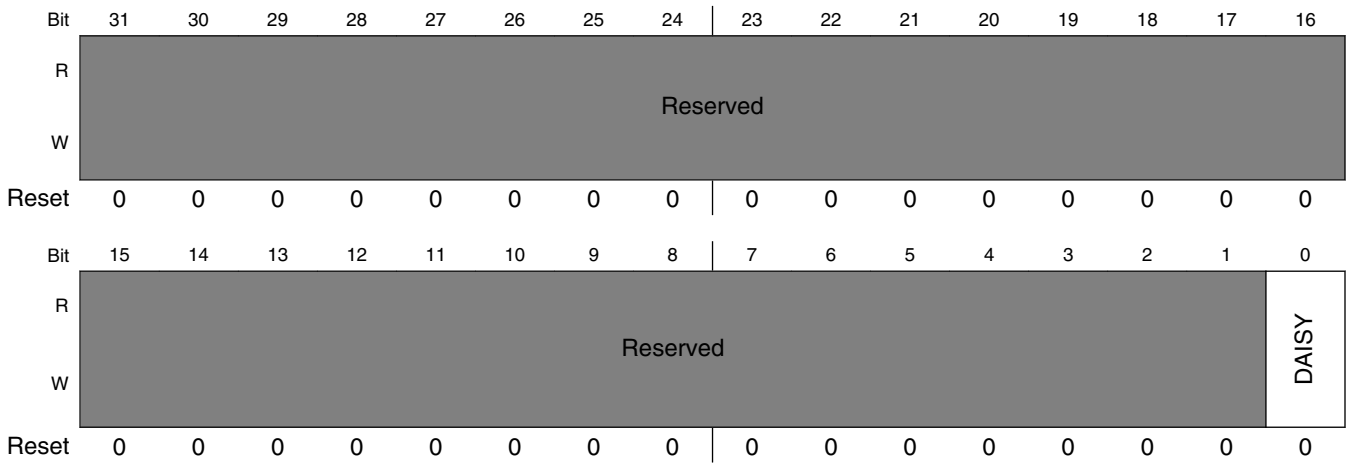
IOMUXC_XBAR1_IN18_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: xbar1, In Pin: xbar_in18 0 **GPIO_EMC_35_ALT1** — Selecting Pad: GPIO_EMC_35 for Mode: ALT1 1 **GPIO_AD_B0_06_ALT6** — Selecting Pad: GPIO_AD_B0_06 for Mode: ALT6

11.6.393 XBAR1_IN20_SELECT_INPUT DAISY Register (IOMUXC_XBAR1_IN20_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 634h offset = 401F_8634h



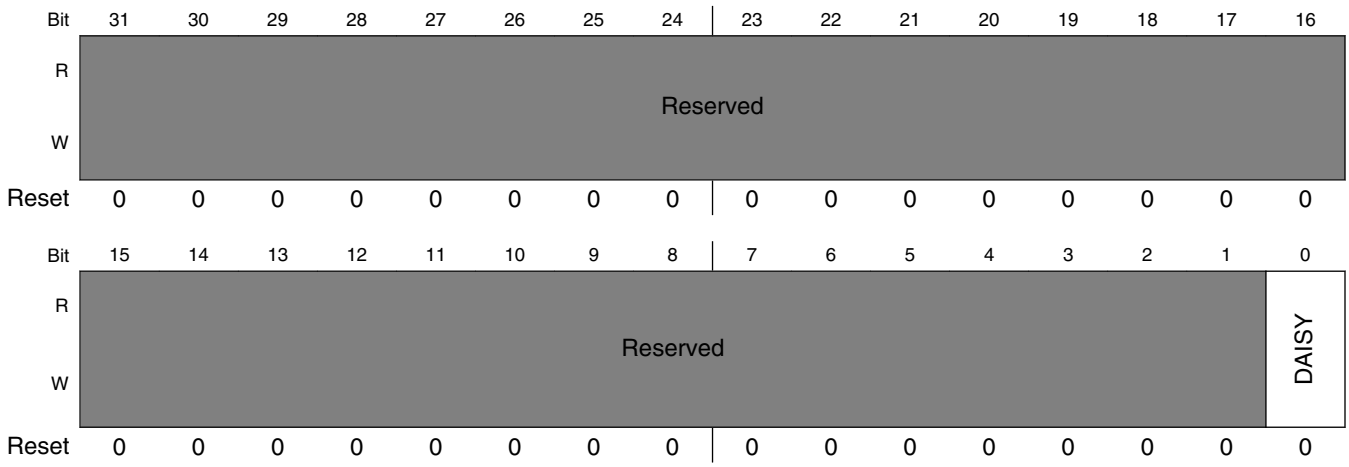
IOMUXC_XBAR1_IN20_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: xbar1, In Pin: xbar_in20 0 **GPIO_EMC_15_ALT1** — Selecting Pad: GPIO_EMC_15 for Mode: ALT1 1 **GPIO_AD_B0_08_ALT6** — Selecting Pad: GPIO_AD_B0_08 for Mode: ALT6

11.6.394 XBAR1_IN22_SELECT_INPUT DAISY Register (IOMUXC_XBAR1_IN22_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 638h offset = 401F_8638h



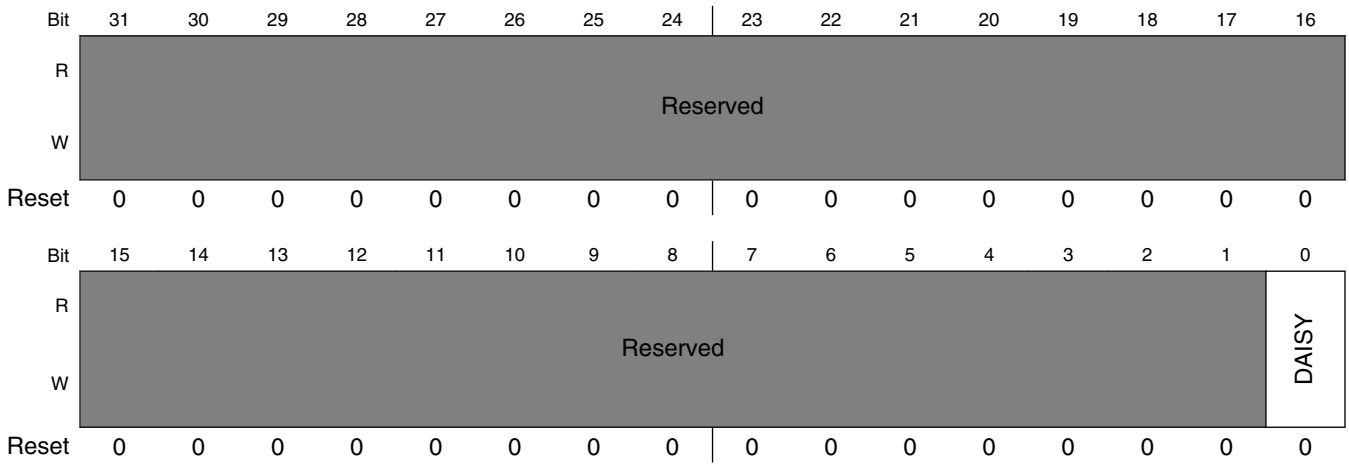
IOMUXC_XBAR1_IN22_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: xbar1, In Pin: xbar_in22 0 **GPIO_EMC_36_ALT1** — Selecting Pad: GPIO_EMC_36 for Mode: ALT1 pin 31 1 **GPIO_AD_B0_10_ALT6** — Selecting Pad: GPIO_AD_B0_10 for Mode: ALT6

11.6.395 XBAR1_IN23_SELECT_INPUT DAISY Register (IOMUXC_XBAR1_IN23_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 63Ch offset = 401F_863Ch



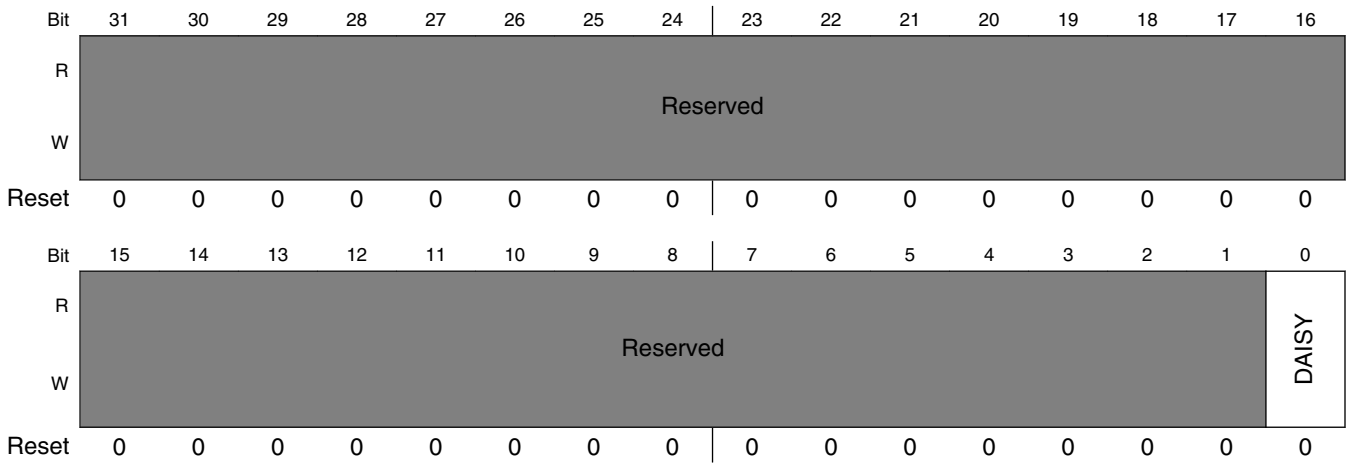
IOMUXC_XBAR1_IN23_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: xbar1, In Pin: xbar_in23 0 **GPIO_EMC_37_ALT1** — Selecting Pad: GPIO_EMC_37 for Mode: ALT1 pin 30 1 **GPIO_AD_B0_11_ALT6** — Selecting Pad: GPIO_AD_B0_11 for Mode: ALT6

11.6.396 XBAR1_IN24_SELECT_INPUT DAISY Register (IOMUXC_XBAR1_IN24_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 640h offset = 401F_8640h



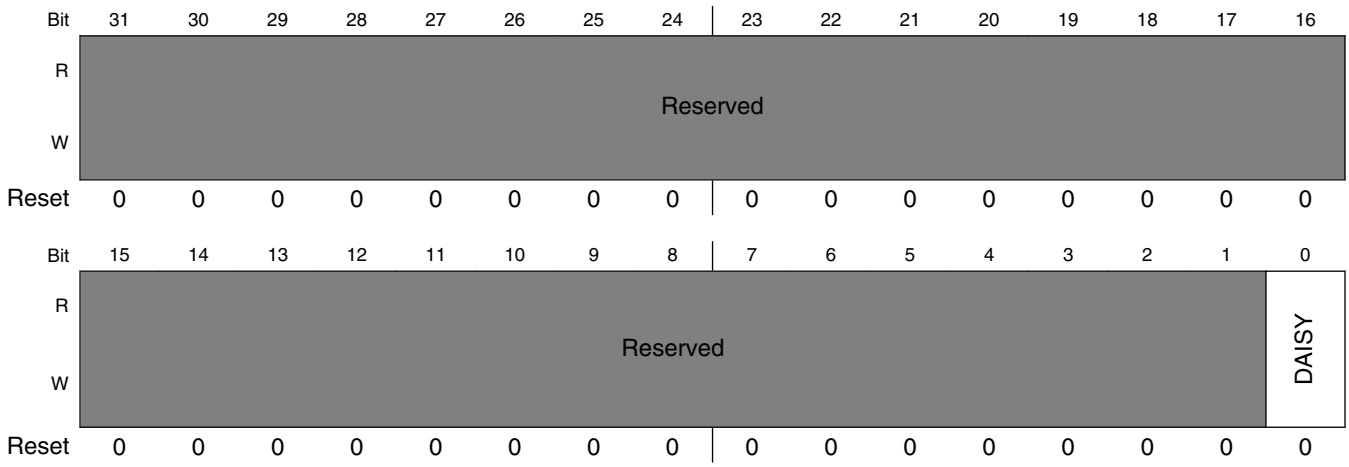
IOMUXC_XBAR1_IN24_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: xbar1, In Pin: xbar_in24 0 **GPIO_EMC_12_ALT1** — Selecting Pad: GPIO_EMC_12 for Mode: ALT1 1 **GPIO_AD_B0_14_ALT1** — Selecting Pad: GPIO_AD_B0_14 for Mode: ALT1

11.6.397 XBAR1_IN14_SELECT_INPUT DAISY Register (IOMUXC_XBAR1_IN14_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 644h offset = 401F_8644h



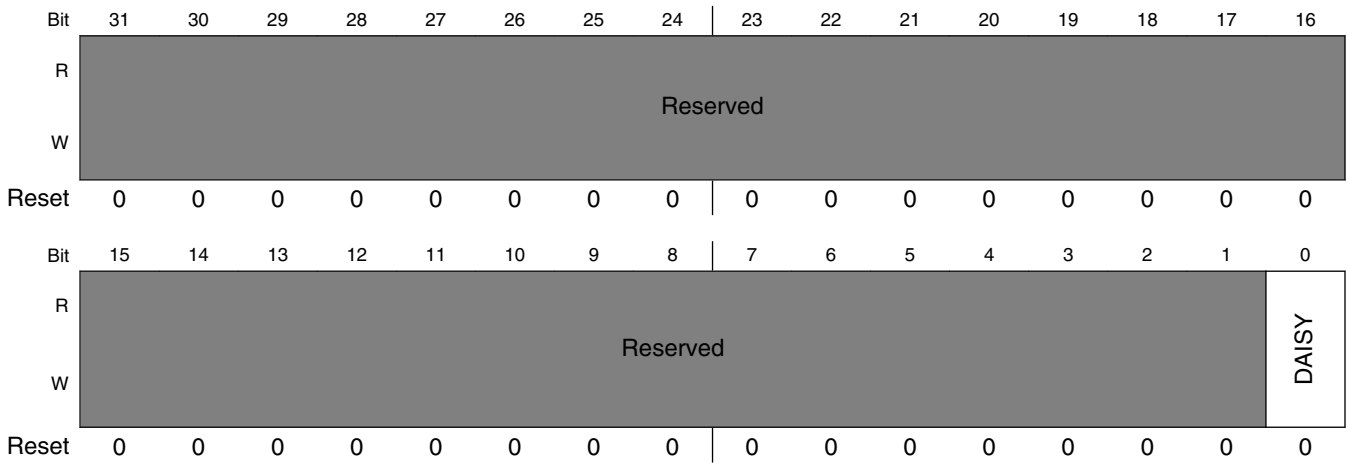
IOMUXC_XBAR1_IN14_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: xbar1, In Pin: xbar_in14 0 **GPIO_AD_B0_00_ALT1** — Selecting Pad: GPIO_AD_B0_00 for Mode: ALT1 1 **GPIO_B1_00_ALT1** — Selecting Pad:GPIO_B1_00 for Mode: ALT1 pin 8

11.6.398 XBAR1_IN15_SELECT_INPUT DAISY Register (IOMUXC_XBAR1_IN15_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 648h offset = 401F_8648h



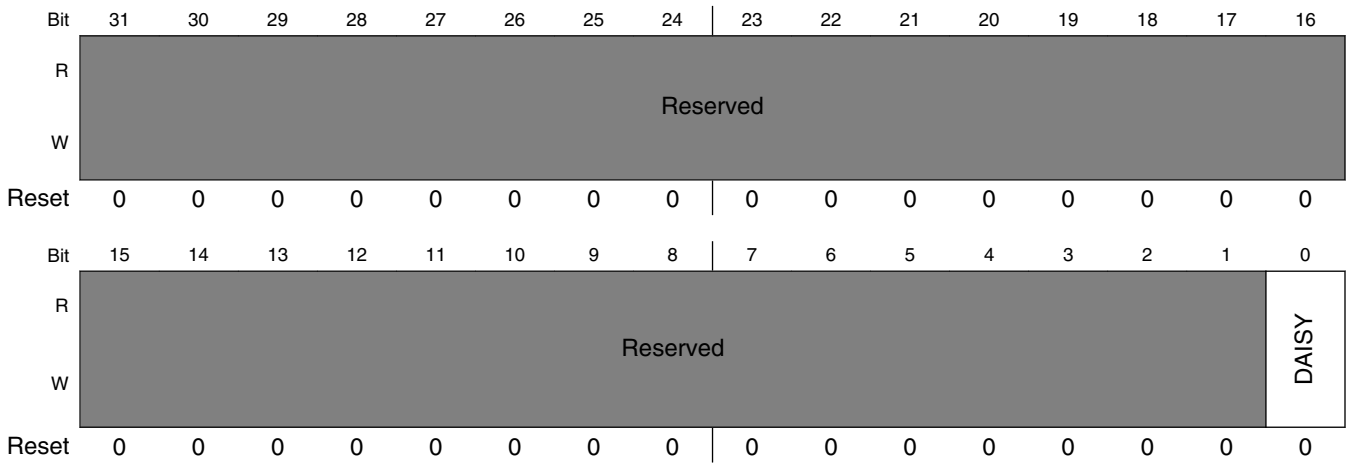
IOMUXC_XBAR1_IN15_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: xbar1, In Pin: xbar_in15 0 **GPIO_AD_B0_01_ALT1** — Selecting Pad: GPIO_AD_B0_01 for Mode: ALT1 1 **GPIO_B1_01_ALT1** — Selecting Pad: GPIO_B1_01 for Mode: ALT1 pin 7

11.6.399 XBAR1_IN16_SELECT_INPUT DAISY Register (IOMUXC_XBAR1_IN16_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 64Ch offset = 401F_864Ch



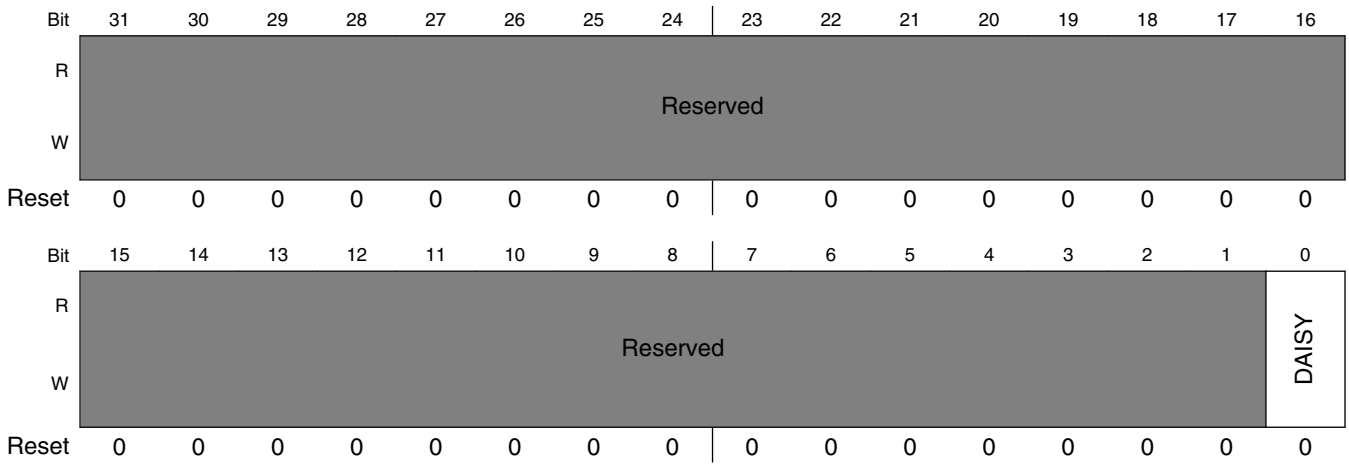
IOMUXC_XBAR1_IN16_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: xbar1, In Pin: xbar_in16 0 **GPIO_AD_B0_02_ALT1** — Selecting Pad: GPIO_AD_B0_02 for Mode: ALT1 pin 1 1 **GPIO_B1_02_ALT1** — Selecting Pad: GPIO_B1_02 for Mode: ALT1 pin 36 (T4.1)

11.6.400 XBAR1_IN25_SELECT_INPUT DAISY Register (IOMUXC_XBAR1_IN25_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 650h offset = 401F_8650h



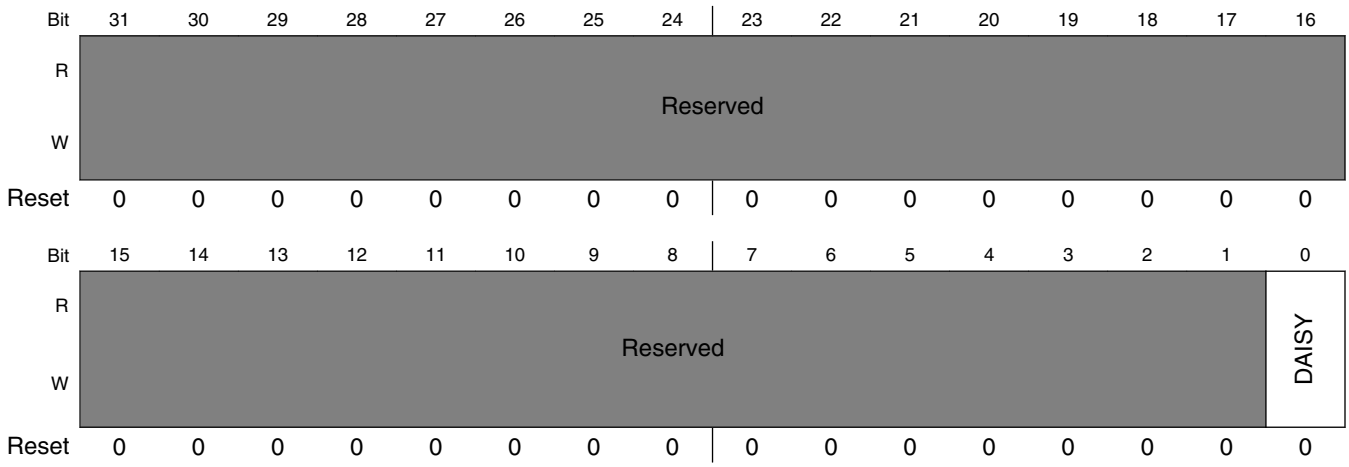
IOMUXC_XBAR1_IN25_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: xbar1, In Pin: xbar_in25 0 **GPIO_AD_B0_15_ALT1** — Selecting Pad: GPIO_AD_B0_15 for Mode: ALT1 1 **GPIO_EMC_13_ALT1** — Selecting Pad: GPIO_EMC_13 for Mode: ALT1

11.6.401 XBAR1_IN19_SELECT_INPUT DAISY Register (IOMUXC_XBAR1_IN19_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 654h offset = 401F_8654h



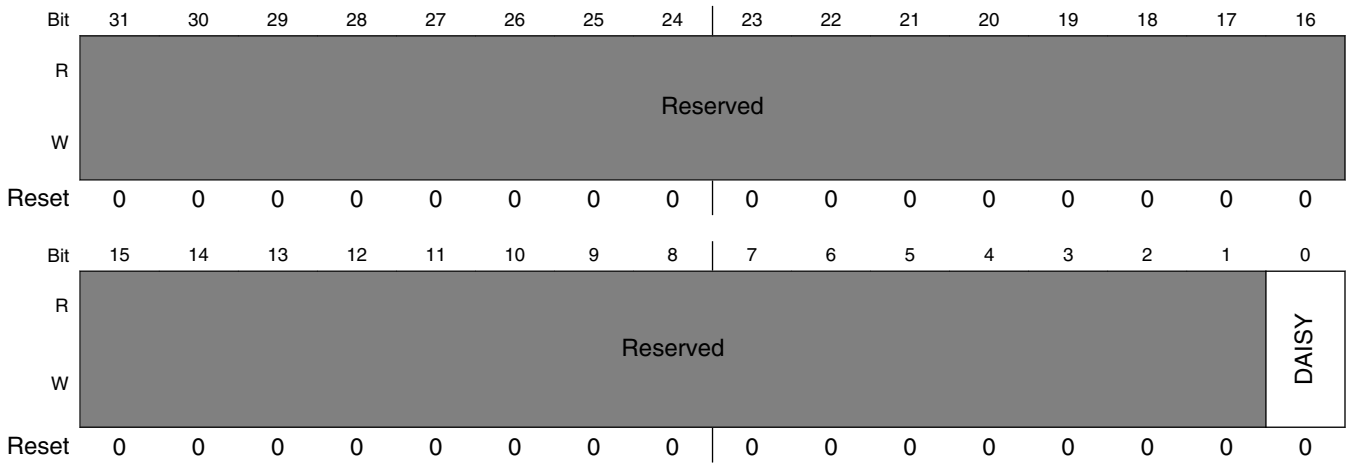
IOMUXC_XBAR1_IN19_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: xbar1, In Pin: xbar_in19 0 **GPIO_EMC_14_ALT1** — Selecting Pad: GPIO_EMC_14 for Mode: ALT1 1 **GPIO_AD_B0_07_ALT6** — Selecting Pad: GPIO_AD_B0_07 for Mode: ALT6

11.6.402 XBAR1_IN23_SELECT_INPUT DAISY Register (IOMUXC_XBAR1_IN21_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 658h offset = 401F_8658h



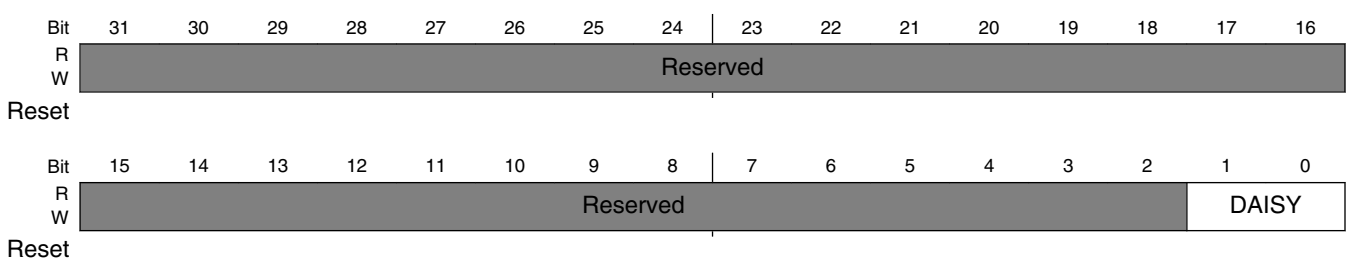
IOMUXC_XBAR1_IN21_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: xbar1, In Pin: xbar_in21 0 **GPIO_EMC_16_ALT1** — Selecting Pad: GPIO_EMC_16 for Mode: ALT1 1 **GPIO_AD_B0_09_ALT6** — Selecting Pad: GPIO_AD_B0_09 for Mode: ALT6

11.6.403 ENET2_IPG_CLK_RMII_SELECT_INPUT DAISY Register (IOMUXC_ENET2_IPG_CLK_RMII_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 70Ch offset = 401F_870Ch



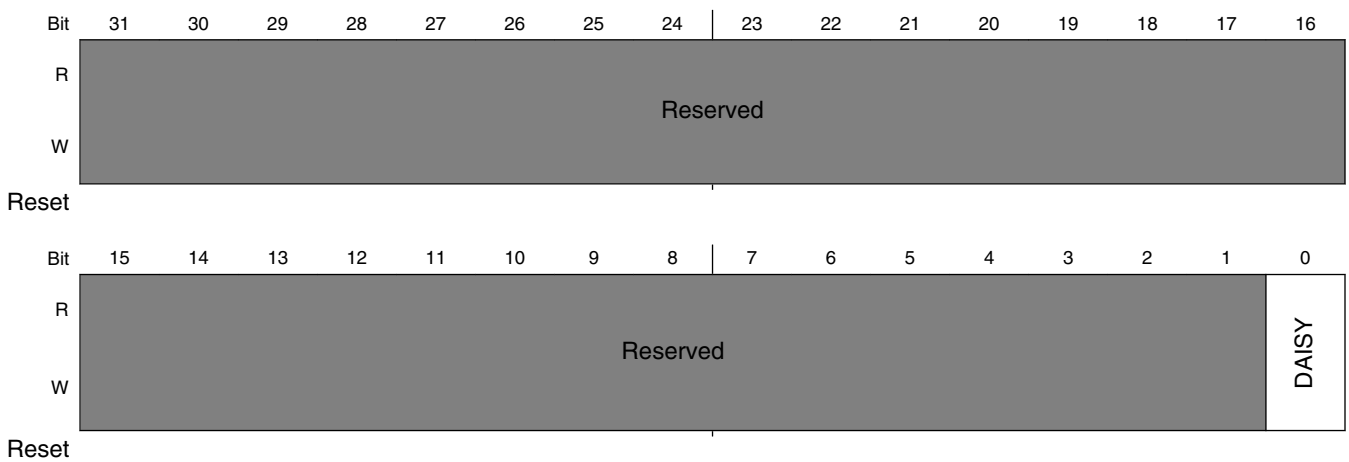
IOMUXC_ENET2_IPG_CLK_RMII_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: enet2, In Pin: ipg_clk_rmii 00 **GPIO_EMC_33_ALT9** — Selecting Pad: GPIO_EMC_33 for Mode: ALT9 01 **GPIO_SD_B0_01_ALT9** — Selecting Pad: GPIO_SD_B0_01 for Mode: ALT9 10 **GPIO_B0_15_ALT9** — Selecting Pad: GPIO_B0_15 for Mode: ALT9

11.6.404 ENET2_IPP_IND_MAC0_MDIO_SELECT_INPUT DAISY Register (IOMUXC_ENET2_IPP_IND_MAC0_MDIO_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 710h offset = 401F_8710h



IOMUXC_ENET2_IPP_IND_MAC0_MDIO_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: enet2, In Pin: ipp_ind_mac0_mdio 0 **GPIO_EMC_39_ALT8** — Selecting Pad: GPIO_EMC_39 for Mode: ALT8 1 **GPIO_B0_01_ALT8** — Selecting Pad: GPIO_B0_01 for Mode: ALT8 pin 12

11.6.405 ENET2_IPP_IND_MAC0_RXDATA_SELECT_INPUT_0 DAISY Register (IOMUXC_ENET2_IPP_IND_MAC0_RXDATA_SELECT_INPUT_0)

DAISY Register

Address: 401F_8000h base + 714h offset = 401F_8714h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset																

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset																

IOMUXC_ENET2_IPP_IND_MAC0_RXDATA_SELECT_INPUT_0 field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: enet2, In Pin: ipp_ind_mac0_rxdata[0] 00 **GPIO_EMC_35_ALT8** — Selecting Pad: GPIO_EMC_35 for Mode: ALT8 01 **GPIO_SD_B0_03_ALT8** — Selecting Pad: GPIO_SD_B0_03 for Mode: ALT8 10 **GPIO_B1_01_ALT8** — Selecting Pad: GPIO_B1_01 for Mode: ALT8 pin 7

11.6.406 ENET2_IPP_IND_MAC0_RXDATA_SELECT_INPUT_1 DAISY Register (IOMUXC_ENET2_IPP_IND_MAC0_RXDATA_SELECT_INPUT_1)

DAISY Register

Address: 401F_8000h base + 718h offset = 401F_8718h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset																

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset																

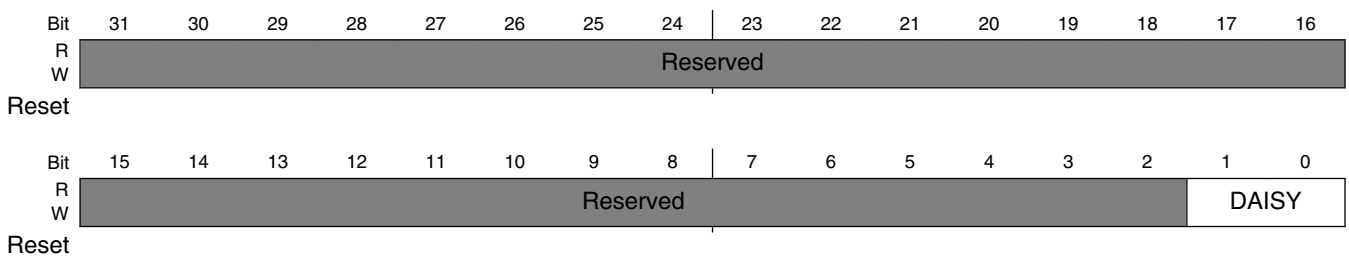
IOMUXC_ENET2_IPP_IND_MAC0_RXDATA_SELECT_INPUT_1 field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: enet2, In Pin: ipp_ind_mac0_rxdata[1] 00 **GPIO_EMC_36_ALT8** — Selecting Pad: GPIO_EMC_36 for Mode: ALT8 pin 31 01 **GPIO_SD_B0_04_ALT8** — Selecting Pad: GPIO_SD_B0_04 for Mode: ALT8 10 **GPIO_B1_02_ALT8** — Selecting Pad: GPIO_B1_02 for Mode: ALT8 pin 36 (T4.1)

11.6.407 ENET2_IPP_IND_MAC0_RXEN_SELECT_INPUT DAISY Register (IOMUXC_ENET2_IPP_IND_MAC0_RXEN_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 71Ch offset = 401F_871Ch



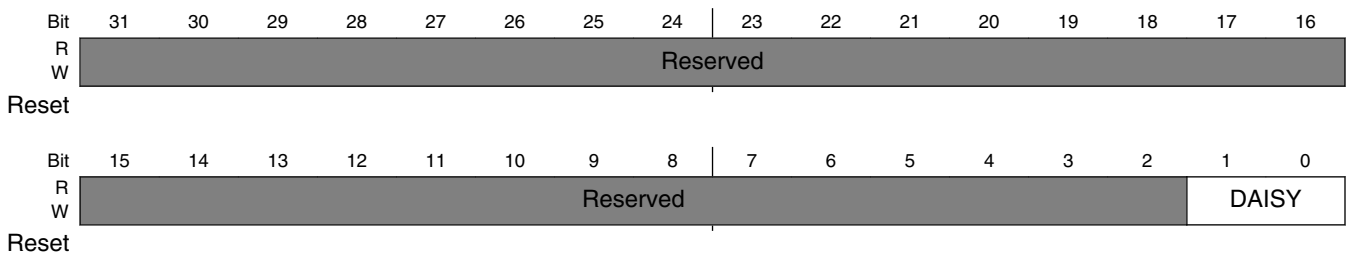
IOMUXC_ENET2_IPP_IND_MAC0_RXEN_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: enet2, In Pin: ipp_ind_mac0_rxen 00 **GPIO_EMC_37_ALT8** — Selecting Pad: GPIO_EMC_37 for Mode: ALT8 pin 30 01 **GPIO_SD_B0_05_ALT8** — Selecting Pad: GPIO_SD_B0_05 for Mode: ALT8 10 **GPIO_B1_03_ALT8** — Selecting Pad: GPIO_B1_03 for Mode: ALT8 pin 37 (T4.1)

11.6.408 ENET2_IPP_IND_MAC0_RXERR_SELECT_INPUT DAISY Register (IOMUXC_ENET2_IPP_IND_MAC0_RXERR_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 720h offset = 401F_8720h



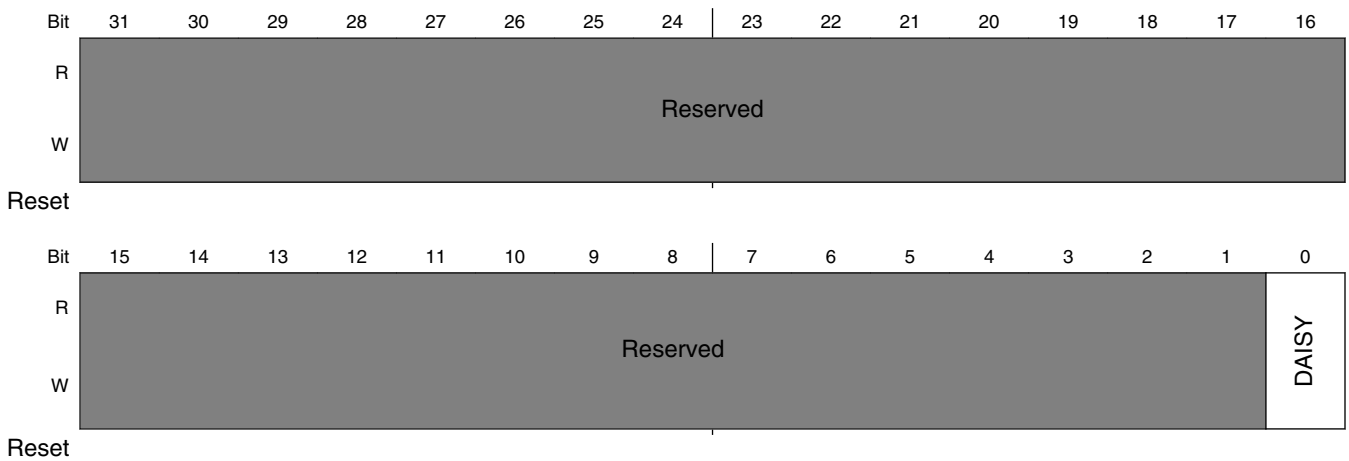
IOMUXC_ENET2_IPP_IND_MAC0_RXERR_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: enet2, In Pin: ipp_ind_mac0_rxerr 00 **GPIO_EMC_34_ALT8** — Selecting Pad: GPIO_EMC_34 for Mode: ALT8 01 **GPIO_SD_B0_02_ALT8** — Selecting Pad: GPIO_SD_B0_02 for Mode: ALT8 10 **GPIO_B1_00_ALT8** — Selecting Pad: GPIO_B1_00 for Mode: ALT8 pin 8

11.6.409 ENET2_IPP_IND_MAC0_TIMER_SELECT_INPUT_0 DAISY Register (IOMUXC_ENET2_IPP_IND_MAC0_TIMER_SELECT_INPUT_0)

DAISY Register

Address: 401F_8000h base + 724h offset = 401F_8724h



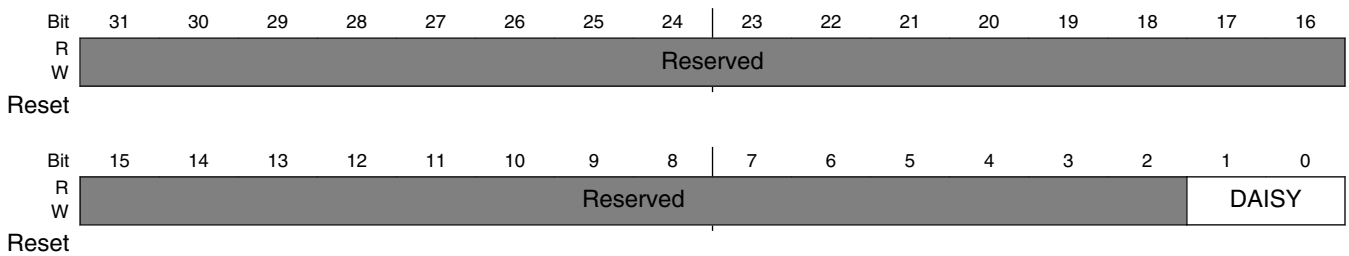
IOMUXC_ENET2_IPP_IND_MAC0_TIMER_SELECT_INPUT_0 field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: enet2, In Pin: ipp_ind_mac0_timer[0] 0 **GPIO_AD_B1_01_ALT8** — Selecting Pad: GPIO_AD_B1_01 for Mode: ALT8 pin 18 1 **GPIO_B0_03_ALT8** — Selecting Pad: GPIO_B0_03 for Mode: ALT8 pin 13

11.6.410 ENET2_IPP_IND_MAC0_TXCLK_SELECT_INPUT DAISY Register (IOMUXC_ENET2_IPP_IND_MAC0_TXCLK_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 728h offset = 401F_8728h



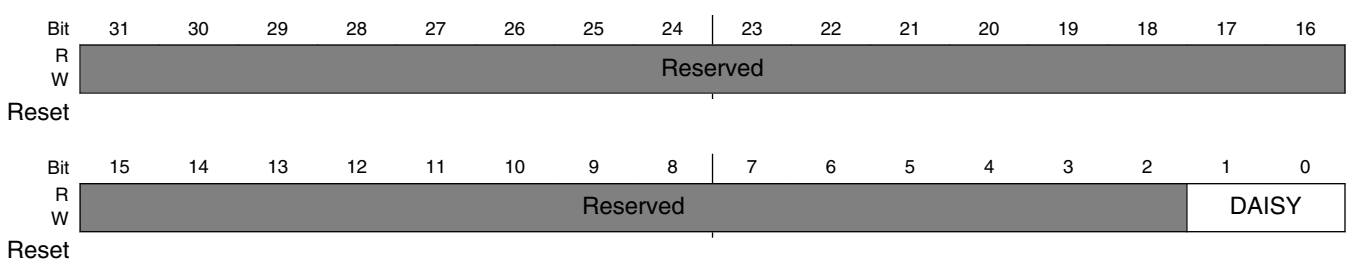
IOMUXC_ENET2_IPP_IND_MAC0_TXCLK_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: enet2, In Pin: ipp_ind_mac0_txclk 00 **GPIO_EMC_33_ALT8** — Selecting Pad: GPIO_EMC_33 for Mode: ALT8 01 **GPIO_SD_B0_01_ALT8** — Selecting Pad: GPIO_SD_B0_01 for Mode: ALT8 10 **GPIO_B0_15_ALT8** — Selecting Pad: GPIO_B0_15 for Mode: ALT8

11.6.411 FLEXSPI2_IPP_IND_DQS_FA_SELECT_INPUT DAISY Register (IOMUXC_FLEXSPI2_IPP_IND_DQS_FA_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 72Ch offset = 401F_872Ch



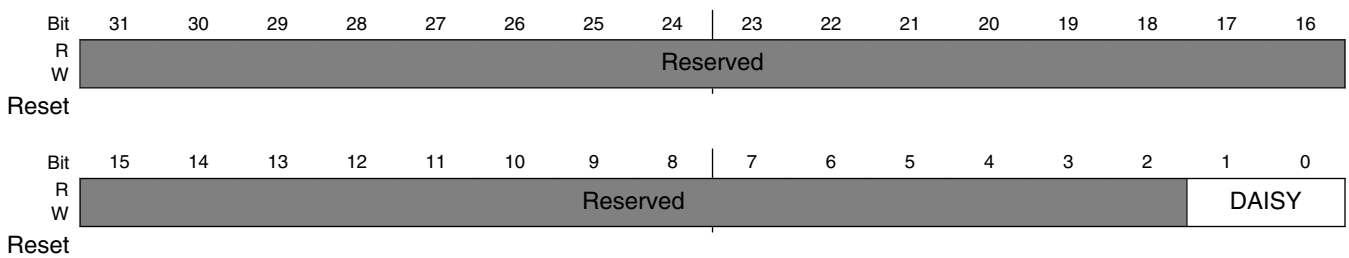
IOMUXC_FLEXSPI2_IPP_IND_DQS_FA_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: flexspi2, In Pin: ipp_ind_dqs_fa 00 **GPIO_SPI_B1_00_ALT0** — Selecting Pad: GPIO_SPI_B1_00 for Mode: ALT0 01 **GPIO_EMC_23_ALT8** — Selecting Pad: GPIO_EMC_23 for Mode: ALT8 10 **GPIO_SPI_B0_09_ALT0** — Selecting Pad: GPIO_SPI_B0_09 for Mode: ALT0

11.6.412 FLEXSPI2_IPP_IND_IO_FA_BIT0_SELECT_INPUT DAISY Register (IOMUXC_FLEXSPI2_IPP_IND_IO_FA_BIT0_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 730h offset = 401F_8730h



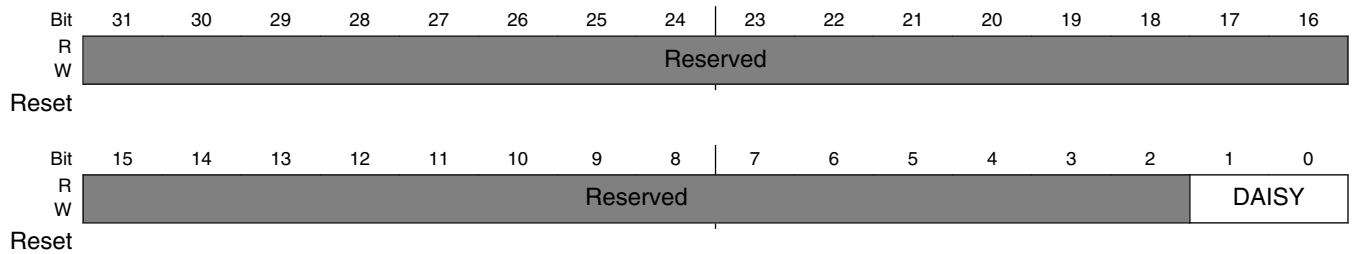
IOMUXC_FLEXSPI2_IPP_IND_IO_FA_BIT0_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: flexspi2, In Pin: ipp_ind_io_fa_bit0 00 **GPIO_SPI_B1_04_ALT0** — Selecting Pad: GPIO_SPI_B1_04 for Mode: ALT0 01 **GPIO_EMC_26_ALT8** — Selecting Pad: GPIO_EMC_26 for Mode: ALT8 10 **GPIO_SPI_B0_02_ALT0** — Selecting Pad: GPIO_SPI_B0_02 for Mode: ALT0

11.6.413 FLEXSPI2_IPP_IND_IO_FA_BIT1_SELECT_INPUT DAISY Register (IOMUXC_FLEXSPI2_IPP_IND_IO_FA_BIT1_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 734h offset = 401F_8734h



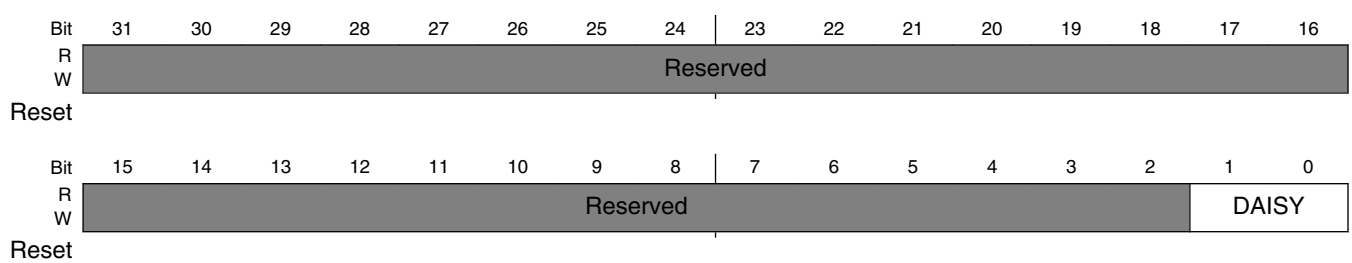
IOMUXC_FLEXSPI2_IPP_IND_IO_FA_BIT1_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: flexspi2, In Pin: ipp_ind_io_fa_bit1 00 **GPIO_SPI_B1_03_ALT0** — Selecting Pad: GPIO_SPI_B1_03 for Mode: ALT0 01 **GPIO_EMC_27_ALT8** — Selecting Pad: GPIO_EMC_27 for Mode: ALT8 10 **GPIO_SPI_B0_12_ALT0** — Selecting Pad: GPIO_SPI_B0_12 for Mode: ALT0

11.6.414 FLEXSPI2_IPP_IND_IO_FA_BIT2_SELECT_INPUT DAISY Register (IOMUXC_FLEXSPI2_IPP_IND_IO_FA_BIT2_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 738h offset = 401F_8738h



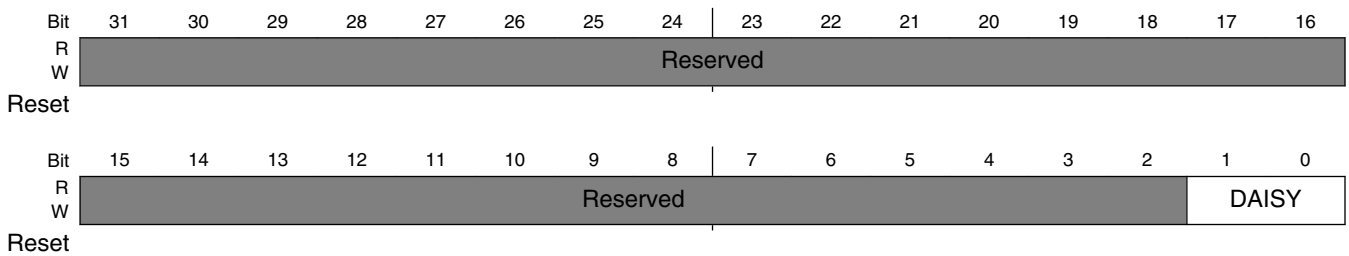
IOMUXC_FLEXSPI2_IPP_IND_IO_FA_BIT2_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: flexspi2, In Pin: ipp_ind_io_fa_bit2 00 **GPIO_SPI_B1_02_ALT0** — Selecting Pad: GPIO_SPI_B1_02 for Mode: ALT0 01 **GPIO_EMC_28_ALT8** — Selecting Pad: GPIO_EMC_28 for Mode: ALT8 10 **GPIO_SPI_B0_06_ALT0** — Selecting Pad: GPIO_SPI_B0_06 for Mode: ALT0

11.6.415 FLEXSPI2_IPP_IND_IO_FA_BIT3_SELECT_INPUT DAISY Register (IOMUXC_FLEXSPI2_IPP_IND_IO_FA_BIT3_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 73Ch offset = 401F_873Ch



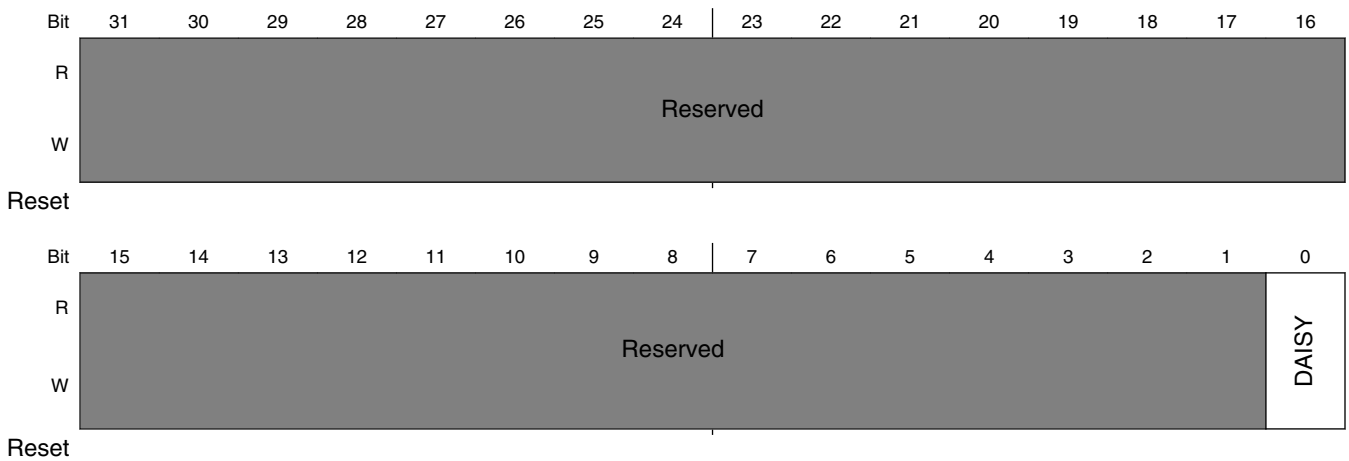
IOMUXC_FLEXSPI2_IPP_IND_IO_FA_BIT3_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: flexspi2, In Pin: ipp_ind_io_fa_bit3 00 **GPIO_SPI_B1_01_ALT0** — Selecting Pad: GPIO_SPI_B1_01 for Mode: ALT0 01 **GPIO_EMC_29_ALT8** — Selecting Pad: GPIO_EMC_29 for Mode: ALT8 10 **GPIO_SPI_B0_10_ALT0** — Selecting Pad: GPIO_SPI_B0_10 for Mode: ALT0

11.6.416 FLEXSPI2_IPP_IND_IO_FB_BIT0_SELECT_INPUT DAISY Register (IOMUXC_FLEXSPI2_IPP_IND_IO_FB_BIT0_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 740h offset = 401F_8740h



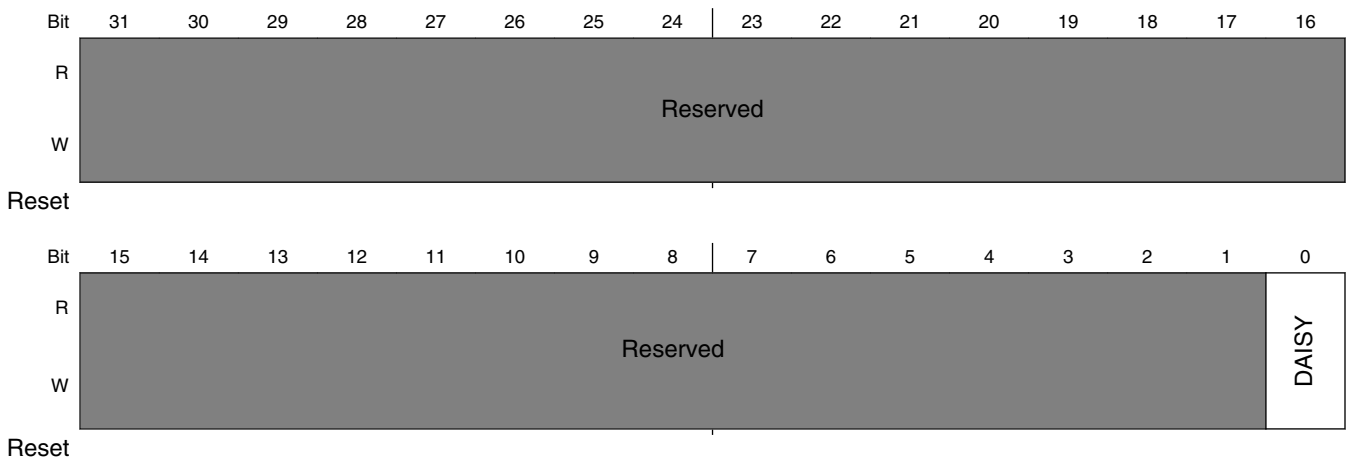
IOMUXC_FLEXSPI2_IPP_IND_IO_FB_BIT0_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: flexspi2, In Pin: ipp_ind_io_fb_bit0 0 **GPIO_EMC_13_ALT8** — Selecting Pad: GPIO_EMC_13 for Mode: ALT8 1 **GPIO_SPI_B0_11_ALT0** — Selecting Pad: GPIO_SPI_B0_11 for Mode: ALT0

11.6.417 FLEXSPI2_IPP_IND_IO_FB_BIT1_SELECT_INPUT DAISY Register (IOMUXC_FLEXSPI2_IPP_IND_IO_FB_BIT1_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 744h offset = 401F_8744h



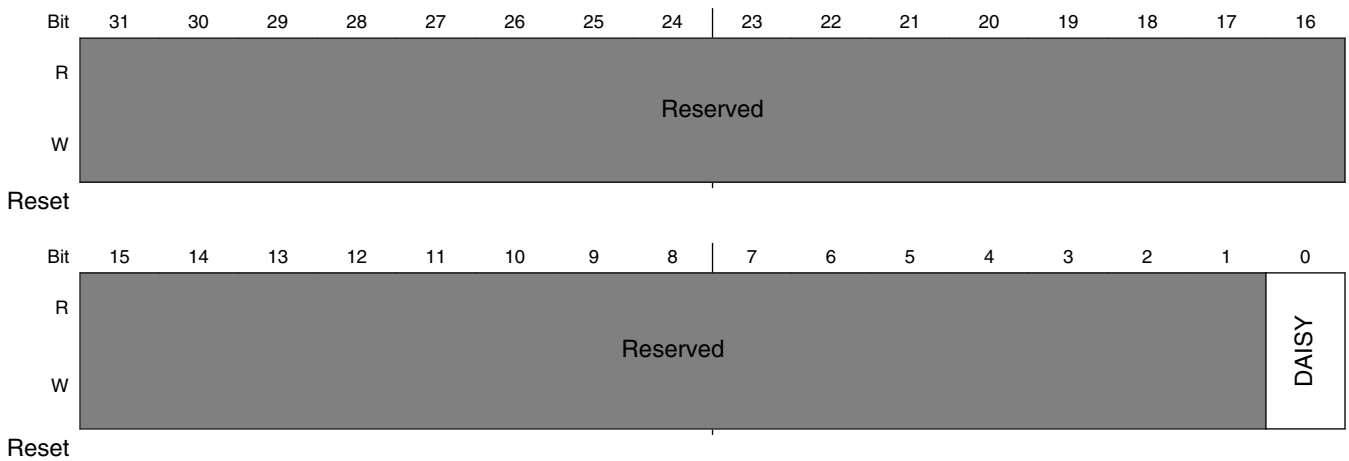
IOMUXC_FLEXSPI2_IPP_IND_IO_FB_BIT1_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: flexspi2, In Pin: ipp_ind_io_fb_bit1 0 **GPIO_EMC_14_ALT8** — Selecting Pad: GPIO_EMC_14 for Mode: ALT8 1 **GPIO_SPI_B0_07_ALT0** — Selecting Pad: GPIO_SPI_B0_07 for Mode: ALT0

11.6.418 FLEXSPI2_IPP_IND_IO_FB_BIT2_SELECT_INPUT DAISY Register (IOMUXC_FLEXSPI2_IPP_IND_IO_FB_BIT2_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 748h offset = 401F_8748h



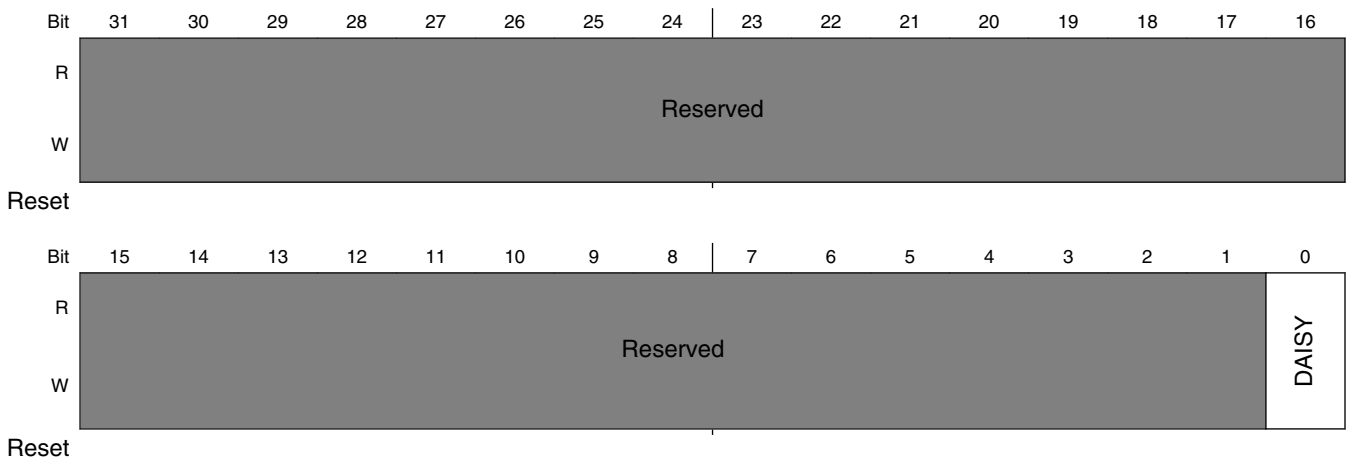
IOMUXC_FLEXSPI2_IPP_IND_IO_FB_BIT2_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: flexspi2, In Pin: ipp_ind_io_fb_bit2 0 **GPIO_EMC_15_ALT8** — Selecting Pad: GPIO_EMC_15 for Mode: ALT8 1 **GPIO_SPI_B0_03_ALT0** — Selecting Pad: GPIO_SPI_B0_03 for Mode: ALT0

11.6.419 FLEXSPI2_IPP_IND_IO_FB_BIT3_SELECT_INPUT DAISY Register (IOMUXC_FLEXSPI2_IPP_IND_IO_FB_BIT3_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 74Ch offset = 401F_874Ch



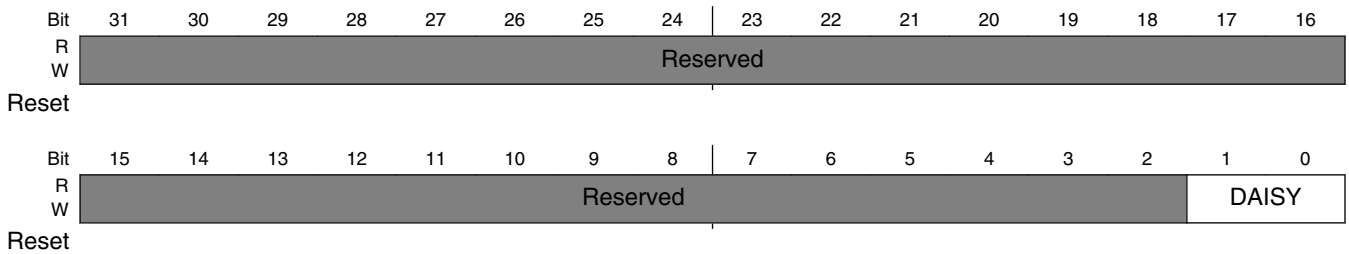
IOMUXC_FLEXSPI2_IPP_IND_IO_FB_BIT3_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: flexspi2, In Pin: ipp_ind_io_fb_bit3 0 **GPIO_EMC_16_ALT8** — Selecting Pad: GPIO_EMC_16 for Mode: ALT8 1 **GPIO_SPI_B0_04_ALT0** — Selecting Pad: GPIO_SPI_B0_04 for Mode: ALT0

11.6.420 FLEXSPI2_IPP_IND_SCK_FA_SELECT_INPUT DAISY Register (IOMUXC_FLEXSPI2_IPP_IND_SCK_FA_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 750h offset = 401F_8750h



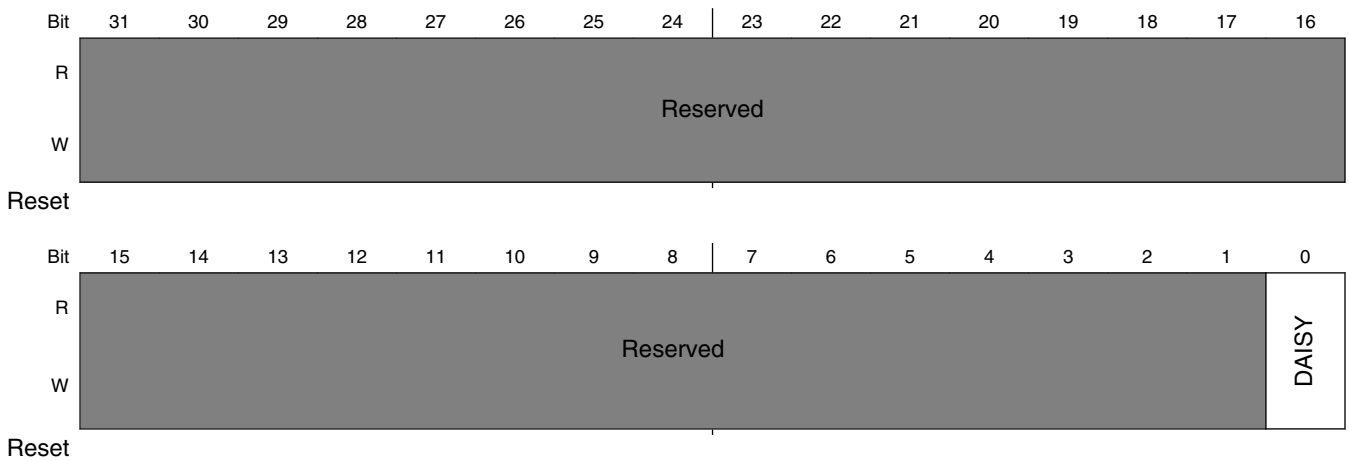
IOMUXC_FLEXSPI2_IPP_IND_SCK_FA_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: flexspi2, In Pin: ipp_ind_sck_fa 00 **GPIO_SPI_B1_05_ALT0** — Selecting Pad: GPIO_SPI_B1_05 for Mode: ALT0 01 **GPIO_EMC_25_ALT8** — Selecting Pad: GPIO_EMC_25 for Mode: ALT8 10 **GPIO_SPI_B0_08_ALT0** — Selecting Pad: GPIO_SPI_B0_08 for Mode: ALT0

11.6.421 FLEXSPI2_IPP_IND_SCK_FB_SELECT_INPUT DAISY Register (IOMUXC_FLEXSPI2_IPP_IND_SCK_FB_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 754h offset = 401F_8754h



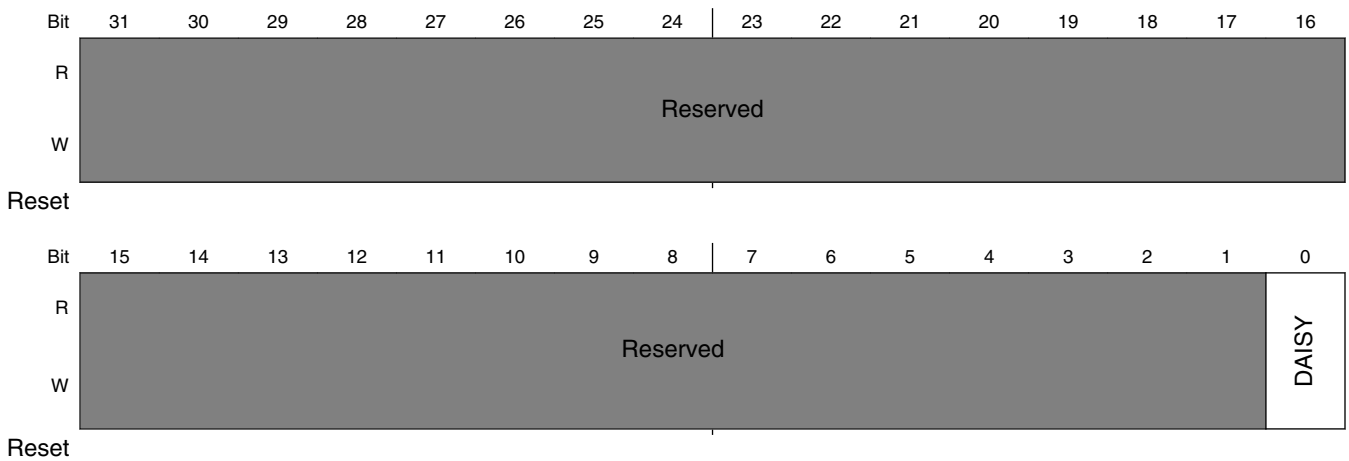
IOMUXC_FLEXSPI2_IPP_IND_SCK_FB_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: flexspi2, In Pin: ipp_ind_sck_fb 0 **GPIO_EMC_12_ALT8** — Selecting Pad: GPIO_EMC_12 for Mode: ALT8 1 **GPIO_SPI_B0_01_ALT0** — Selecting Pad: GPIO_SPI_B0_01 for Mode: ALT0

11.6.422 GPT1_IPP_IND_CAPIN1_SELECT_INPUT DAISY Register (IOMUXC_GPT1_IPP_IND_CAPIN1_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 758h offset = 401F_8758h



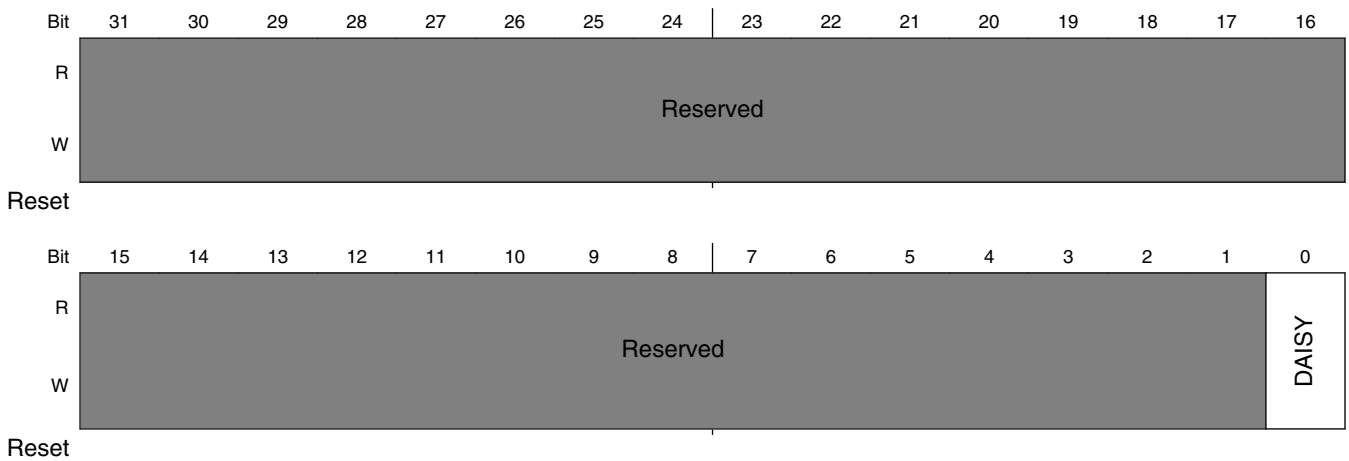
IOMUXC_GPT1_IPP_IND_CAPIN1_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: gpt1, In Pin: ipp_ind_capin1 0 **GPIO_EMC_24_ALT4** — Selecting Pad: GPIO_EMC_24 for Mode: ALT4 1 **GPIO_B1_05_ALT8** — Selecting Pad: GPIO_B1_05 for Mode: ALT8

11.6.423 GPT1_IPP_IND_CAPIN2_SELECT_INPUT DAISY Register (IOMUXC_GPT1_IPP_IND_CAPIN2_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 75Ch offset = 401F_875Ch



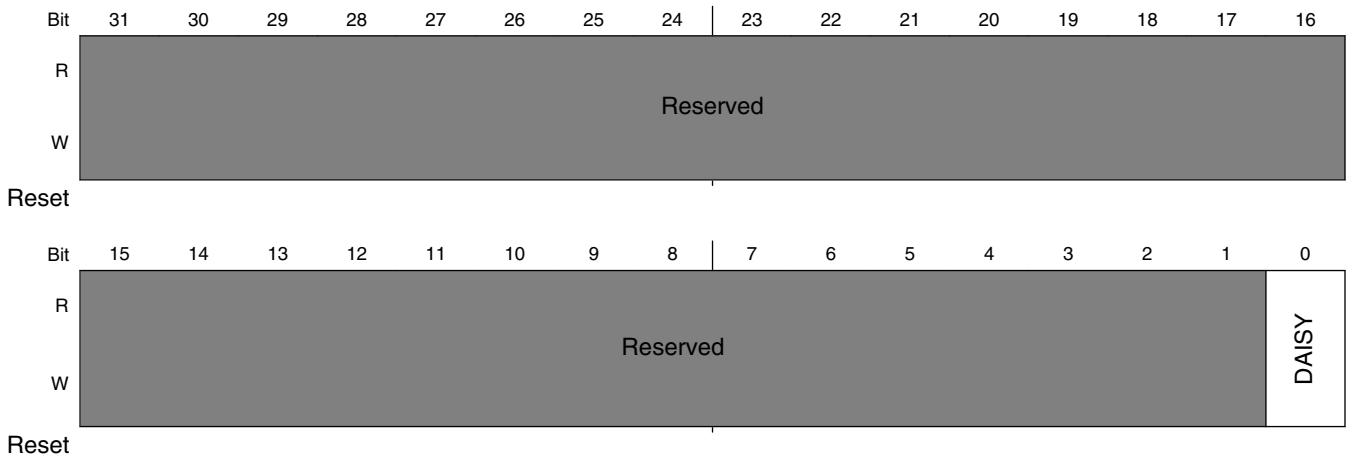
IOMUXC_GPT1_IPP_IND_CAPIN2_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: gpt1, In Pin: ipp_ind_capin2 0 **GPIO_EMC_23_ALT4** — Selecting Pad: GPIO_EMC_23 for Mode: ALT4 1 **GPIO_B1_06_ALT8** — Selecting Pad: GPIO_B1_06 for Mode: ALT8

11.6.424 GPT1_IPP_IND_CLKIN_SELECT_INPUT DAISY Register (IOMUXC_GPT1_IPP_IND_CLKIN_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 760h offset = 401F_8760h



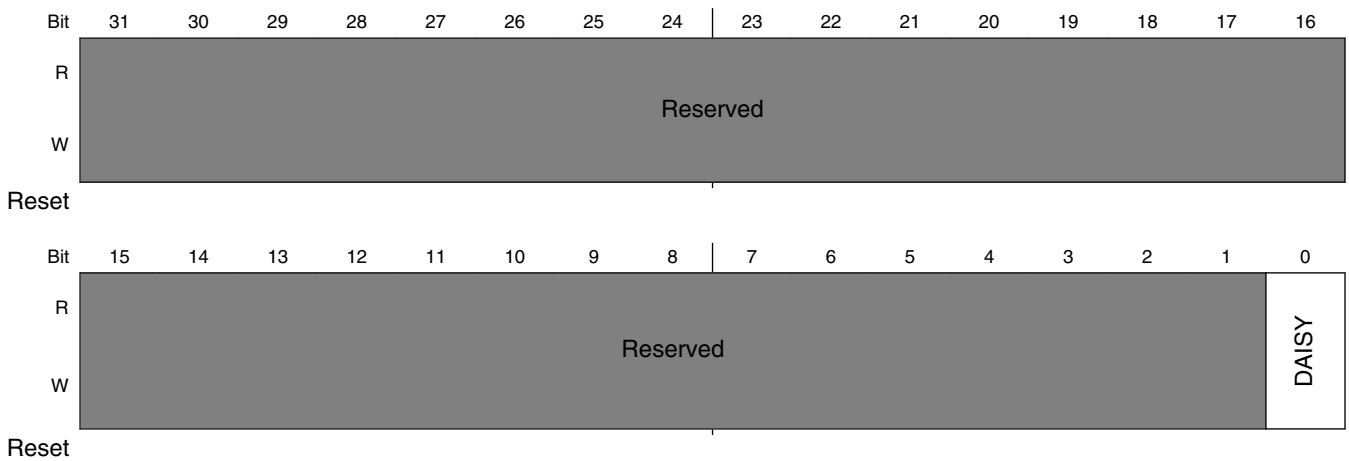
IOMUXC_GPT1_IPP_IND_CLKIN_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: gpt1, In Pin: ipp_ind_clkin 0 **GPIO_AD_B0_13_ALT1** — Selecting Pad: GPIO_AD_B0_13 for Mode: ALT1 pin 25 1 **GPIO_B1_04_ALT8** — Selecting Pad: GPIO_B1_04 for Mode: ALT8

11.6.425 GPT2_IPP_IND_CAPIN1_SELECT_INPUT DAISY Register (IOMUXC_GPT2_IPP_IND_CAPIN1_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 764h offset = 401F_8764h



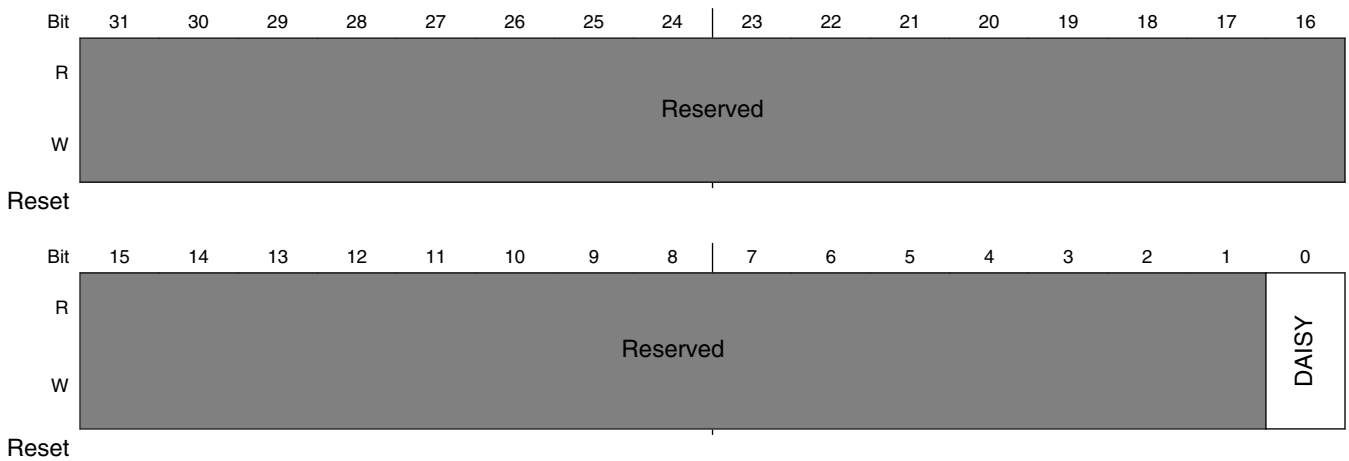
IOMUXC_GPT2_IPP_IND_CAPIN1_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: gpt2, In Pin: ipp_ind_capin1 0 **GPIO_EMC_41_ALT1** — Selecting Pad: GPIO_EMC_41 for Mode: ALT1 1 **GPIO_AD_B1_03_ALT8** — Selecting Pad: GPIO_AD_B1_03 for Mode: ALT8 pin 15

11.6.426 GPT2_IPP_IND_CAPIN2_SELECT_INPUT DAISY Register (IOMUXC_GPT2_IPP_IND_CAPIN2_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 768h offset = 401F_8768h



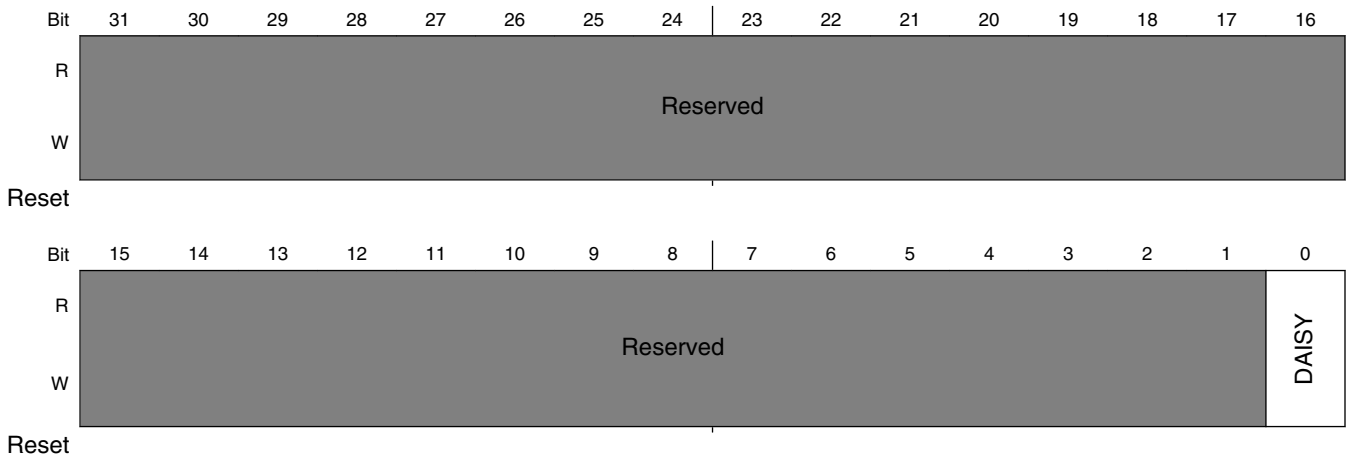
IOMUXC_GPT2_IPP_IND_CAPIN2_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: gpt2, In Pin: ipp_ind_capin2 0 **GPIO_EMC_40_ALT1** — Selecting Pad: GPIO_EMC_40 for Mode: ALT1 1 **GPIO_AD_B1_04_ALT8** — Selecting Pad: GPIO_AD_B1_04 for Mode: ALT8 pin 40 (T4.1)

11.6.427 GPT2_IPP_IND_CLKIN_SELECT_INPUT DAISY Register (IOMUXC_GPT2_IPP_IND_CLKIN_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 76Ch offset = 401F_876Ch



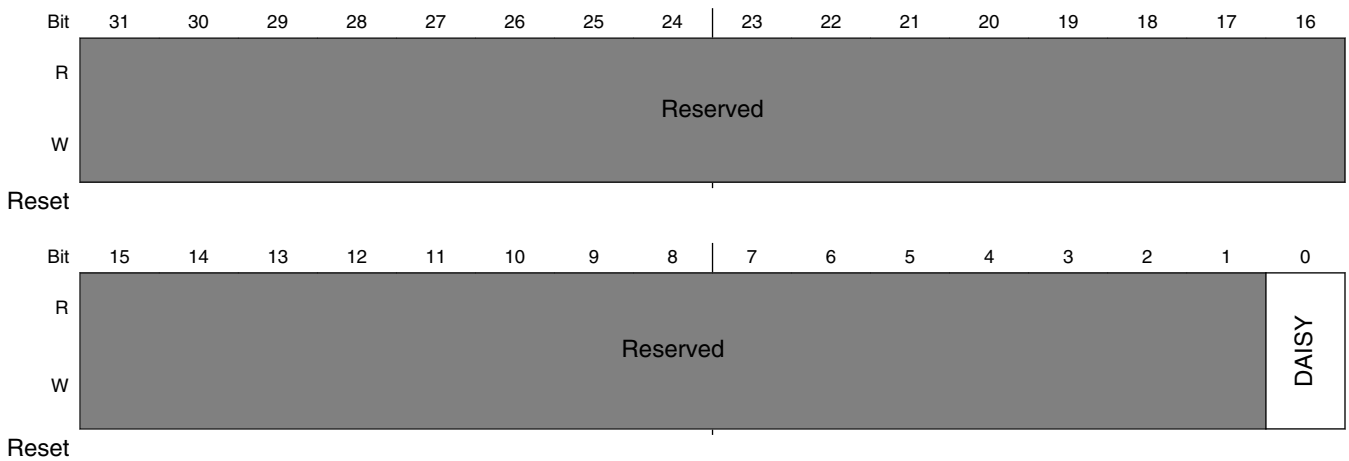
IOMUXC_GPT2_IPP_IND_CLKIN_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: gpt2, In Pin: ipp_ind_clkin 0 **GPIO_AD_B0_09_ALT7** — Selecting Pad: GPIO_AD_B0_09 for Mode: ALT7 1 **GPIO_AD_B1_02_ALT8** — Selecting Pad: GPIO_AD_B1_02 for Mode: ALT8 pin 14

11.6.428 SAI3_IPG_CLK_SAI_MCLK_SELECT_INPUT_2 DAISY Register (IOMUXC_SAI3_IPG_CLK_SAI_MCLK_SELECT_INPUT_2)

DAISY Register

Address: 401F_8000h base + 770h offset = 401F_8770h



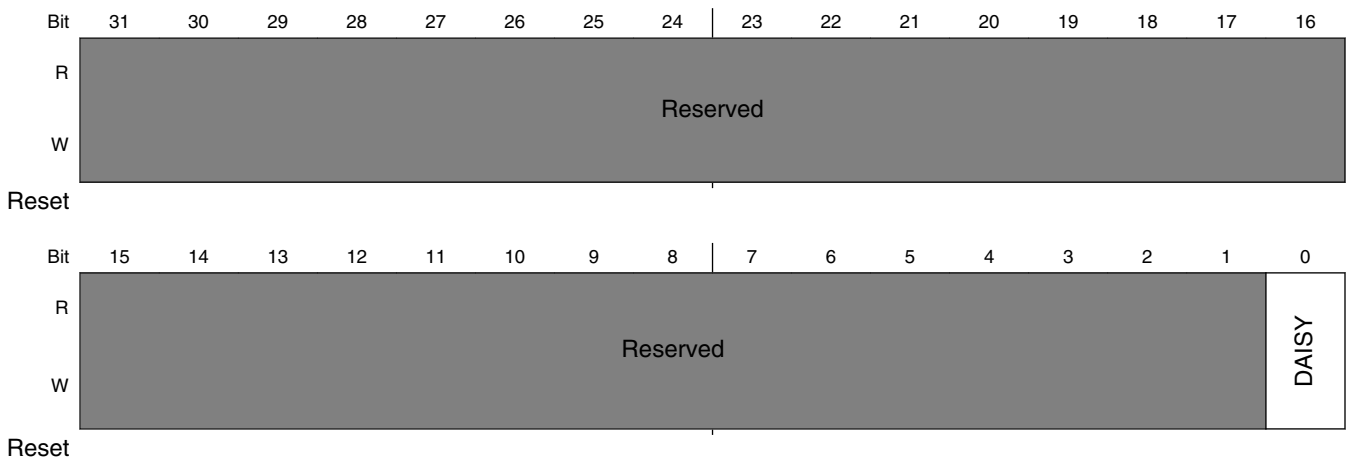
IOMUXC_SAI3_IPG_CLK_SAI_MCLK_SELECT_INPUT_2 field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: sai3, In Pin: ipg_clk_sai_mclk[2] 0 **GPIO_EMC_37_ALT3** — Selecting Pad: GPIO_EMC_37 for Mode: ALT3 pin 30 1 **GPIO_SD_B1_04_ALT8** — Selecting Pad: GPIO_SD_B1_04 for Mode: ALT8

11.6.429 SAI3_IPP_IND_SAI_RXBCLK_SELECT_INPUT DAISY Register (IOMUXC_SAI3_IPP_IND_SAI_RXBCLK_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 774h offset = 401F_8774h



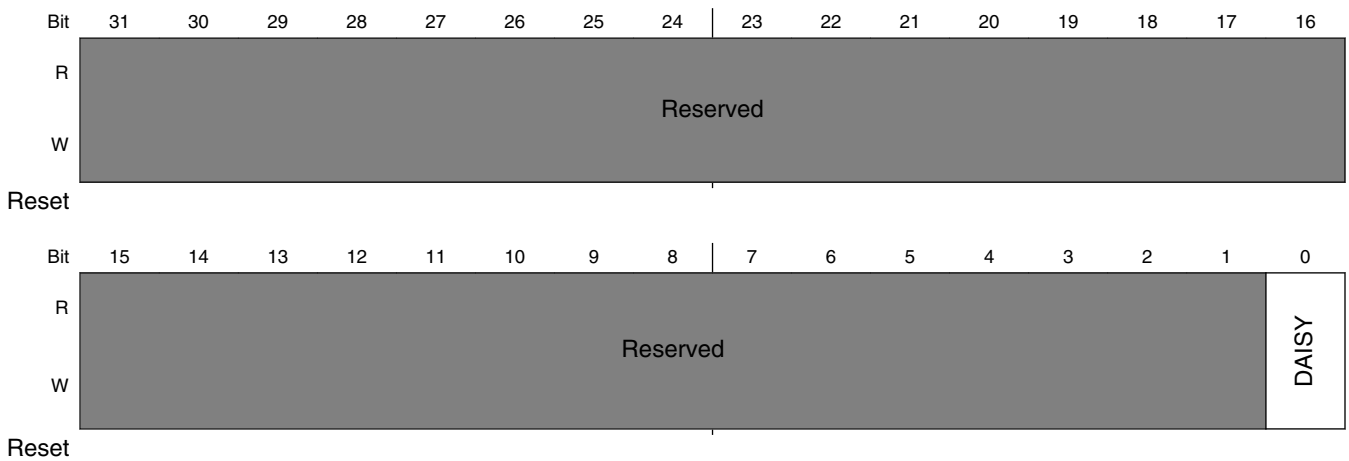
IOMUXC_SAI3_IPP_IND_SAI_RXBCLK_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: sai3, In Pin: ipp_ind_sai_rxbclk 0 **GPIO_EMC_35_ALT3** — Selecting Pad: GPIO_EMC_35 for Mode: ALT3 1 **GPIO_SD_B1_06_ALT8** — Selecting Pad: GPIO_SD_B1_06 for Mode: ALT8

11.6.430 SAI3_IPP_IND_SAI_RXDATA_SELECT_INPUT_0 DAISY Register (IOMUXC_SAI3_IPP_IND_SAI_RXDATA_SELECT_INPUT_0)

DAISY Register

Address: 401F_8000h base + 778h offset = 401F_8778h



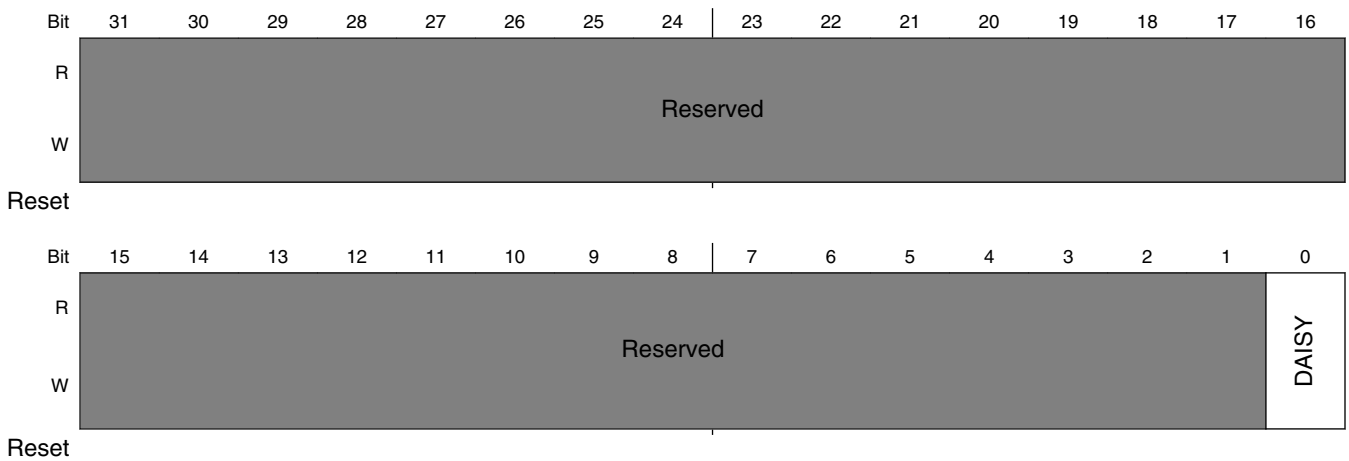
IOMUXC_SAI3_IPP_IND_SAI_RXDATA_SELECT_INPUT_0 field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: sai3, In Pin: ipp_ind_sai_rxdata[0] 0 **GPIO_EMC_33_ALT3** — Selecting Pad: GPIO_EMC_33 for Mode: ALT3 1 **GPIO_SD_B1_00_ALT8** — Selecting Pad: GPIO_SD_B1_00 for Mode: ALT8

11.6.431 SAI3_IPP_IND_SAI_RXSYNC_SELECT_INPUT DAISY Register (IOMUXC_SAI3_IPP_IND_SAI_RXSYNC_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 77Ch offset = 401F_877Ch



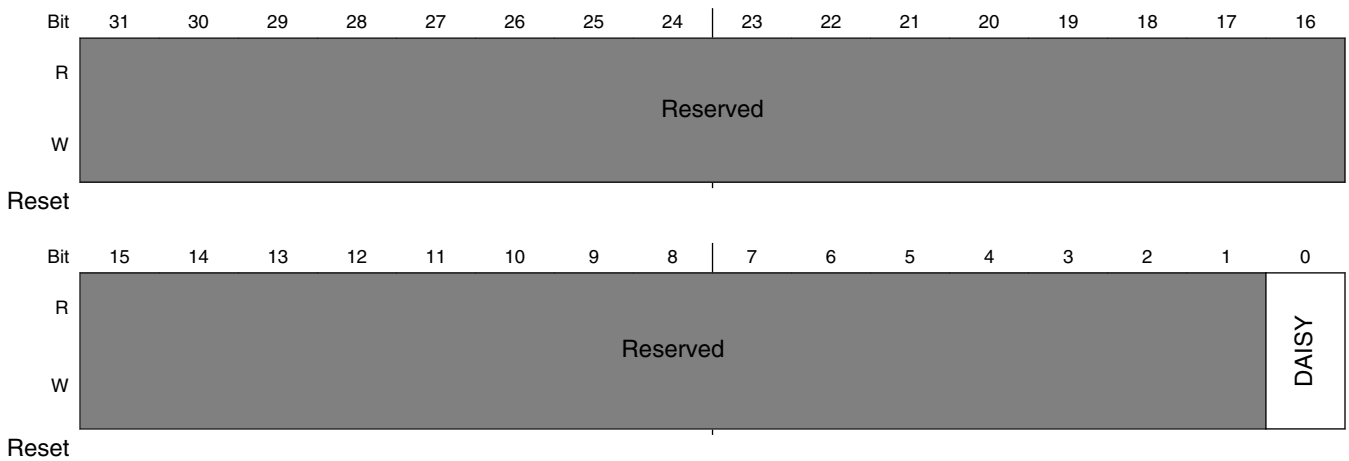
IOMUXC_SAI3_IPP_IND_SAI_RXSYNC_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: sai3, In Pin: ipp_ind_sai_rxsync 0 **GPIO_EMC_34_ALT3** — Selecting Pad: GPIO_EMC_34 for Mode: ALT3 1 **GPIO_SD_B1_05_ALT8** — Selecting Pad: GPIO_SD_B1_05 for Mode: ALT8

11.6.432 SAI3_IPP_IND_SAI_TXBCLK_SELECT_INPUT DAISY Register (IOMUXC_SAI3_IPP_IND_SAI_TXBCLK_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 780h offset = 401F_8780h



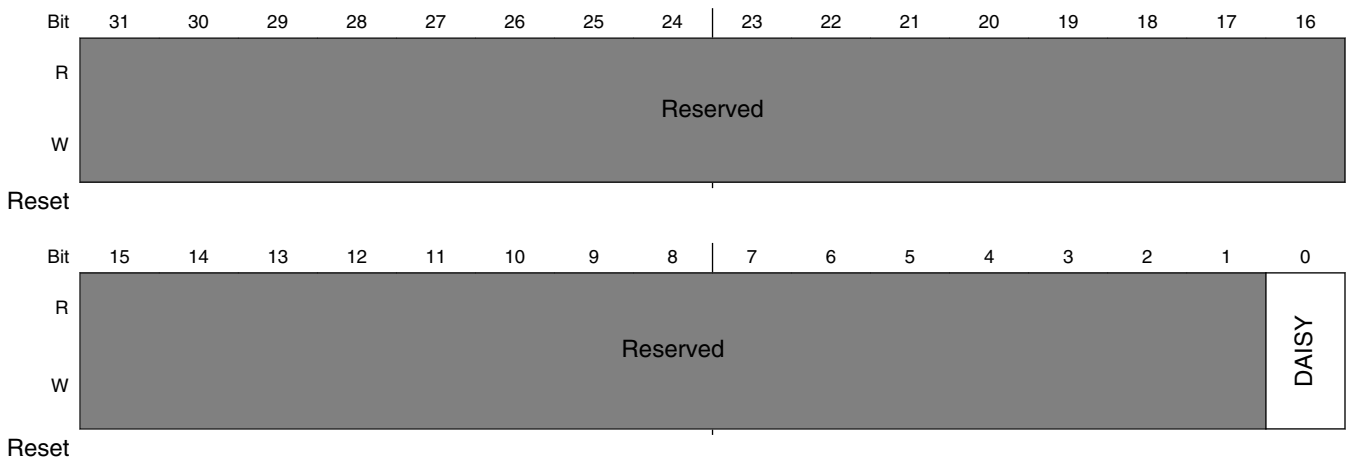
IOMUXC_SAI3_IPP_IND_SAI_TXBCLK_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: sai3, In Pin: ipp_ind_sai_txbclk 0 **GPIO_EMC_38_ALT3** — Selecting Pad: GPIO_EMC_38 for Mode: ALT3 1 **GPIO_SD_B1_03_ALT8** — Selecting Pad: GPIO_SD_B1_03 for Mode: ALT8

11.6.433 SAI3_IPP_IND_SAI_TXSYNC_SELECT_INPUT DAISY Register (IOMUXC_SAI3_IPP_IND_SAI_TXSYNC_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 784h offset = 401F_8784h



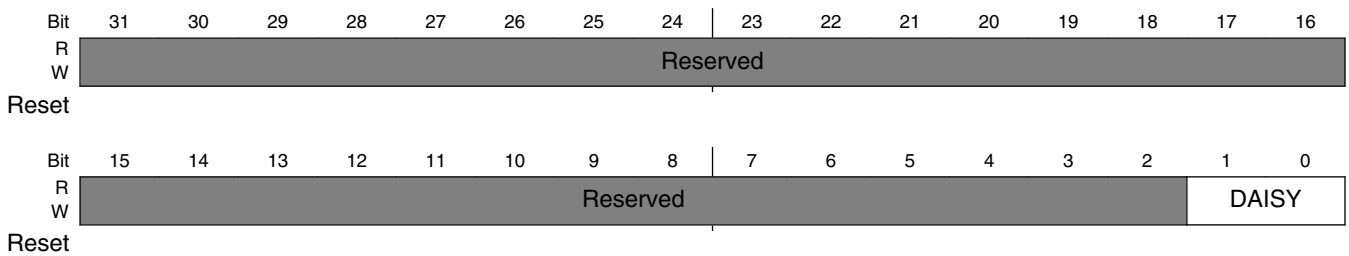
IOMUXC_SAI3_IPP_IND_SAI_TXSYNC_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: sai3, In Pin: ipp_ind_sai_txsync 0 **GPIO_EMC_39_ALT3** — Selecting Pad: GPIO_EMC_39 for Mode: ALT3 1 **GPIO_SD_B1_02_ALT8** — Selecting Pad: GPIO_SD_B1_02 for Mode: ALT8

11.6.434 SEMC_I_IPP_IND_DQS4_SELECT_INPUT DAISY Register (IOMUXC_SEMC_I_IPP_IND_DQS4_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 788h offset = 401F_8788h



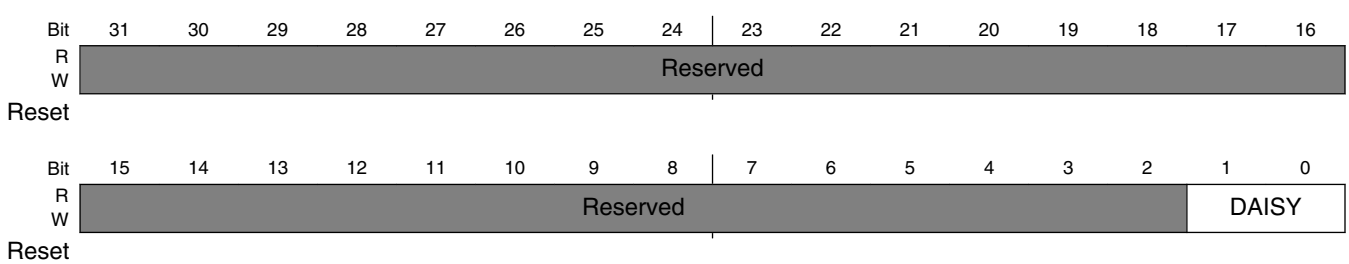
IOMUXC_SEMC_I_IPP_IND_DQS4_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: semc, In Pin: i_ipp_ind_dqs4 00 **GPIO_SD_B0_00_ALT9** — Selecting Pad: GPIO_SD_B0_00 for Mode: ALT9 01 **GPIO_EMC_39_ALT9** — Selecting Pad: GPIO_EMC_39 for Mode: ALT9 10 **GPIO_AD_B0_09_ALT9** — Selecting Pad: GPIO_AD_B0_09 for Mode: ALT9 11 **GPIO_B1_13_ALT8** — Selecting Pad: GPIO_B1_13 for Mode: ALT8 pin 34 (T4.1)

11.6.435 CANFD_IPP_IND_CANRX_SELECT_INPUT DAISY Register (IOMUXC_CANFD_IPP_IND_CANRX_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 78Ch offset = 401F_878Ch



IOMUXC_CANFD_IPP_IND_CANRX_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. Instance: flexcan3/canfd, In Pin: ipp_ind_canrx 00 **GPIO_EMC_37_ALT9** — Selecting Pad: GPIO_EMC_37 for Mode: ALT9 pin 30 01 **GPIO_AD_B0_15_ALT8** — Selecting Pad: GPIO_AD_B0_15 for Mode: ALT8 10 **GPIO_AD_B0_11_ALT8** — Selecting Pad: GPIO_AD_B0_11 for Mode: ALT8

Chapter 12
General Purpose Input/Output (GPIO)

12.1 Chip-specific GPIO information

Table 12-1. Reference links to related information

Topic	Related module(s)	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
Interrupts, DMA Events and XBAR Assignments	-	Interrupts, DMA Events and XBAR Assignments

High-speed GPIOs exist in this device.

- GPIO1-5 are standard-speed GPIOs that run off the IPG_CLK_ROOT, while GPIO6-9 are high-speed GPIOs that run at the AHB_CLK_ROOT frequency. See the table "System Clocks, Gating, and Override" in CCM chapter.
- Regular GPIO and high speed GPIO are paired (GPIO1 and GPIO6 share the same pins, GPIO2 and GPIO7 share, etc). The IOMUXC_GPR_GPR26-29 registers are used to determine if the regular or high-speed GPIO module is used for the GPIO pins on a given port.

Teensy startup code configures all pins to use fast GPIO6 - GPIO9. While this allows normal code to run much faster, the trade-off is no DMA access and all pins sharing a single pin change interrupt. For DMA access (as used by OctoWS2811 library) or special lower-latency interrupt response, individual pins can be configured to use of GPIO1 - GPIO4, but normal read / write access becomes much slower.

12.2 Overview

The General-Purpose Input/Output (GPIO) peripheral provides dedicated general-purpose pins that can be configured as either inputs or outputs.

When configured as an output, it is possible to write to an internal register to control the state driven on the output pin. When configured as an input, it is possible to detect the state of the input by reading the state of an internal register. In addition, the GPIO peripheral can produce CORE interrupts.

12.2.1 Block Diagram

The GPIO subsystem contains multiple GPIO blocks, which can generate and control up to 32 signals for general purpose.

The block diagram for the GPIO block is provided below.

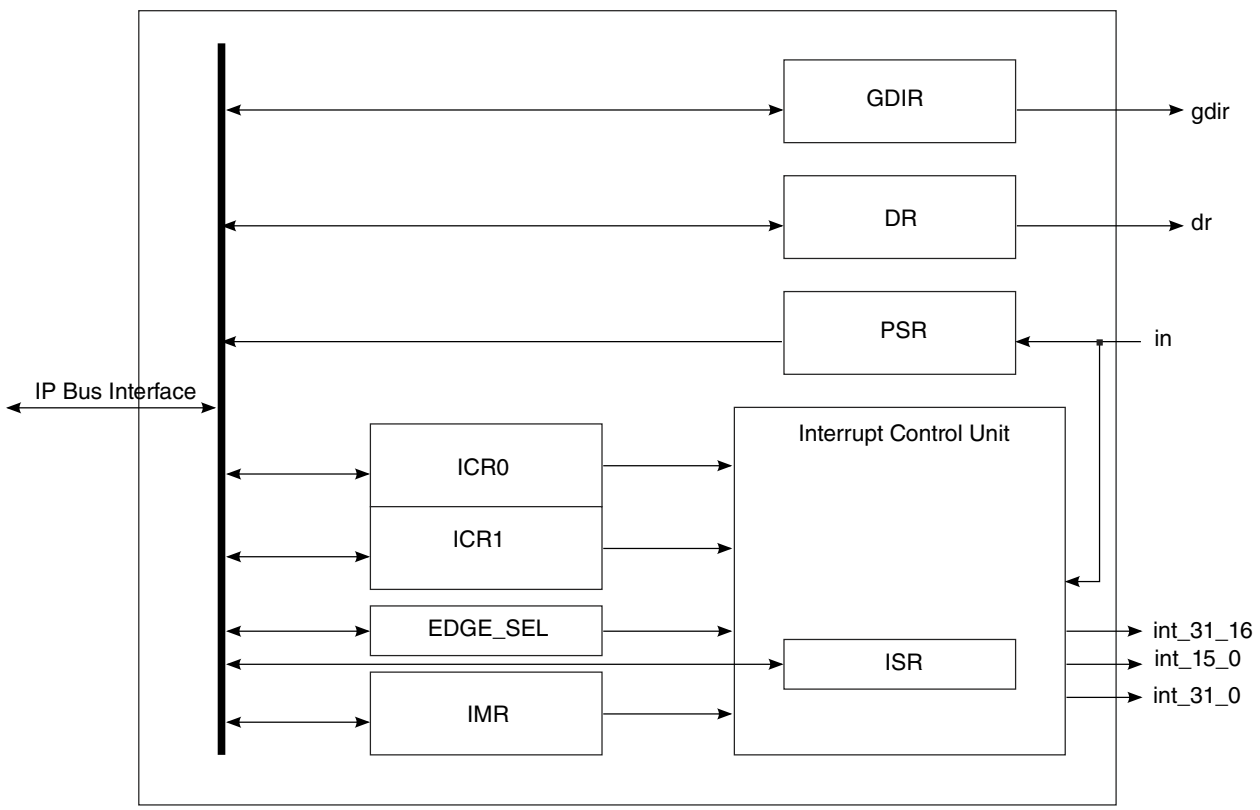


Figure 12-1. GPIO Block Diagram

12.2.2 Features

The GPIO includes the following features:

- General purpose input/output logic capabilities:
  - Drives specific data to output using the data register (DR)

    - Controls the direction of the signal using the GPIO direction register (GDIR)
    - Enables the core to sample the status of the corresponding inputs by reading the pad sample register (PSR).
- GPIO interrupt capabilities:
    - Supports 32 interrupts
    - Identifies interrupt edges
    - Generates three active-high interrupts to the SoC interrupt controller

12.3 Functional Description

The GPIO is one of the blocks controlling the IOMUX of the chip.

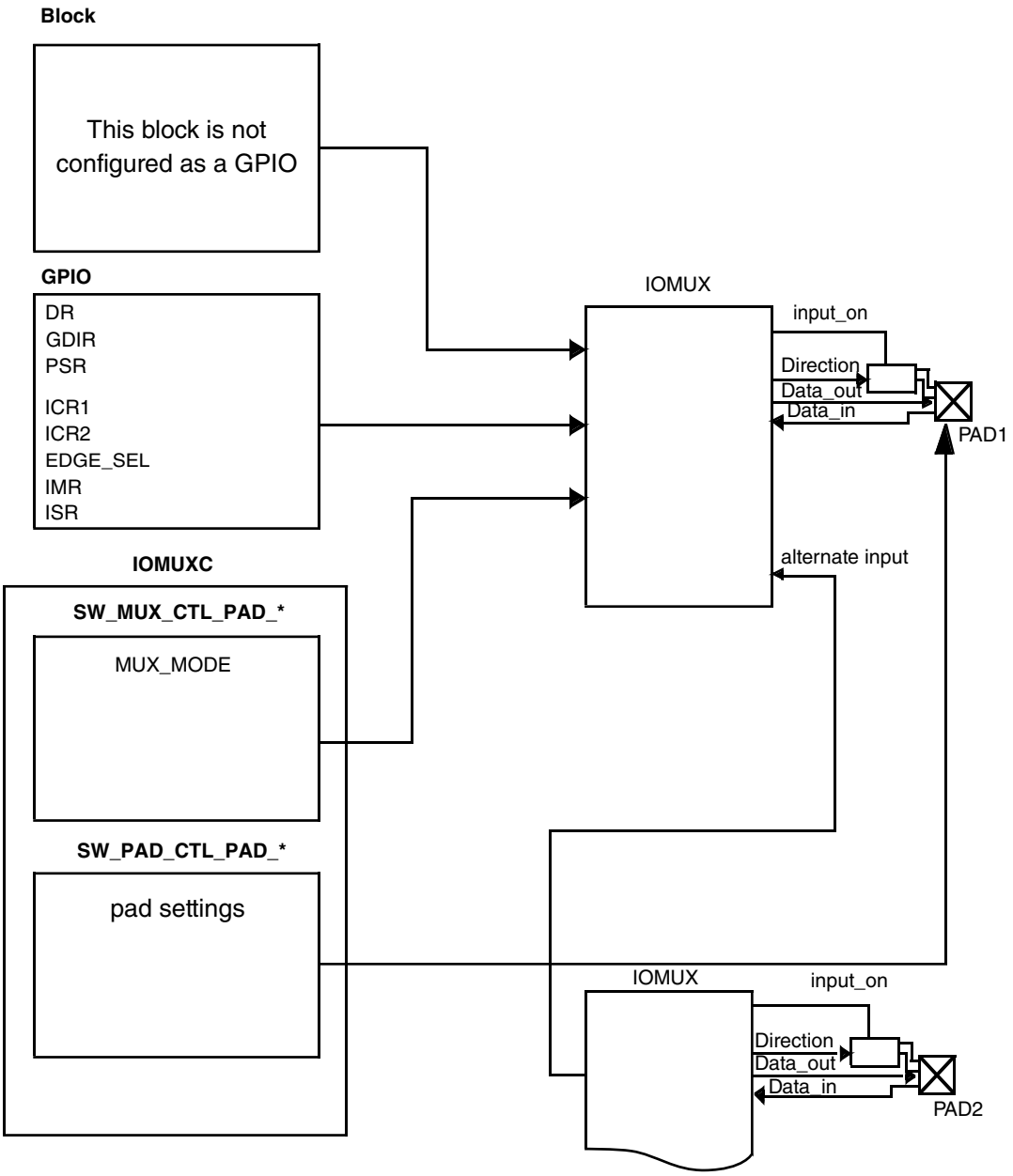


Figure 12-2. Chip IOMUX Scheme

A GPIO signal can operate as a general-purpose input/output when the IOMUX is set to GPIO mode. Each GPIO signal may be independently configured as either an input or an output using the GPIO direction register (GDIR).

When configured as an output (GDIR[GDIR] = 1), the value in the data bit in the GPIO data register (DR) is driven on the corresponding GPIO line. When a signal is configured as an input (GDIR[GDIR] = 0), the state of the input can be read from the corresponding PSR[PSR] bit.

The GPIO functionality is provided through eight registers, an edge-detect circuit, and interrupt generation logic.

The eight registers are:

- Data register (DR)
- GPIO direction register (GDIR)
- Pad sample register (PSR)
- Interrupt control registers (ICR1, ICR2)
- Edge select register (EDGE_SEL)
- Interrupt mask register (IMR)
- Interrupt status register (ISR)

The above registers are described in detail in the Register section.

Each GPIO input has a dedicated edge-detect circuit which can be configured through software to detect rising edges, falling edges, logic low-levels or logic high-levels on the input signals. The outputs of the edge detect circuits are optionally masked by setting the corresponding bit in the interrupt mask register (IMR). These qualified outputs are OR'ed together to generate two one-bit interrupt lines:

- Combined interrupt indication for GPIOx signals 0 - 15
- Combined interrupt indication for GPIOx signals 16 - 31

In addition, GPIO1 provides visibility to each of its 8 low order interrupt sources (i.e. GPIO1 interrupt n, for n = 0 – 7). However, individual interrupt indications from other GPIOx are not available.

The GPIO edge detection is described further in Interrupt Control Unit.

12.3.1 GPIO pad structure

The functional diagram for the GPIO pad is provided below.

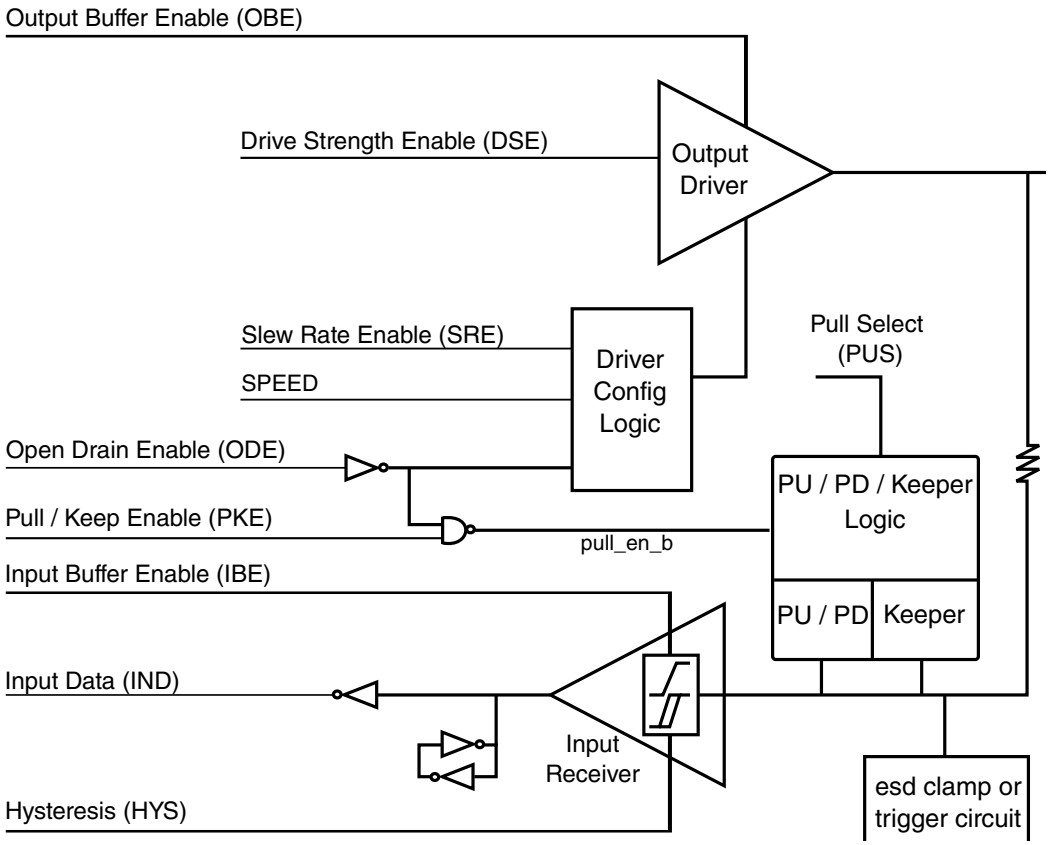


Figure 12-3. GPIO pad functional diagram

12.3.1.1 Input Driver

Input driver characteristics

- Selectable Schmitt trigger or CMOS input mode
- Keeper structure with buffer at the input receiver output to Core
- Receiver is tri-stated when I/O supply (OVDD) is powered down. The keeper at the receiver output keeps it's previous state.

12.3.1.1.1 Schmitt trigger

The anti-jamming functionality of the Schmitt trigger is illustrated below.

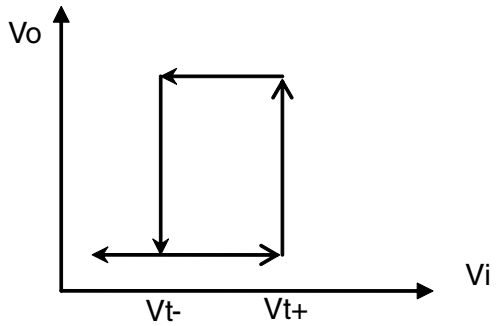


Figure 12-4. Schmitt trigger transfer characteristic

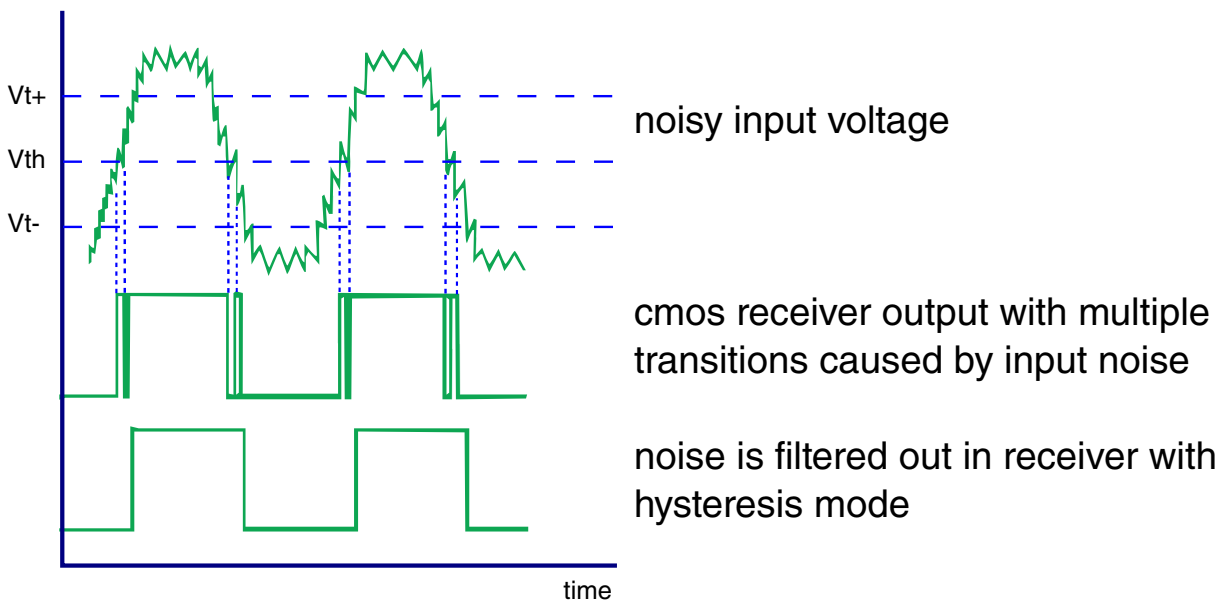


Figure 12-5. Receiver output in CMOS and hysteresis

12.3.1.1.2 Input keeper

A simple latch to hold the input value when OVDD is powered down, or the first inverter is tri-stated. Input buffer's keeper is always enabled for all the pads.

12.3.1.2 Output Driver

Output driver characteristics

- Selectable CMOS or open-drain output type

- Selectable pull-keeper enable signal to enable/disable the pull-up/down and output keeper
- Selectable pull-up resistors of 22K, 47K, 100K and a pull-down resistor of 100KOhm. Unsilicided P+ poly resistor is used to limit resistance variation to within +/- 20%.
- Pull-up, pull-down, and pad keeper are disabled in output mode.
- Seven drive strengths in each operating mode
- Additional 2-bit slew rate control to select between 50, 100, and 200 MHz IO cell operation range with reduced switching noise

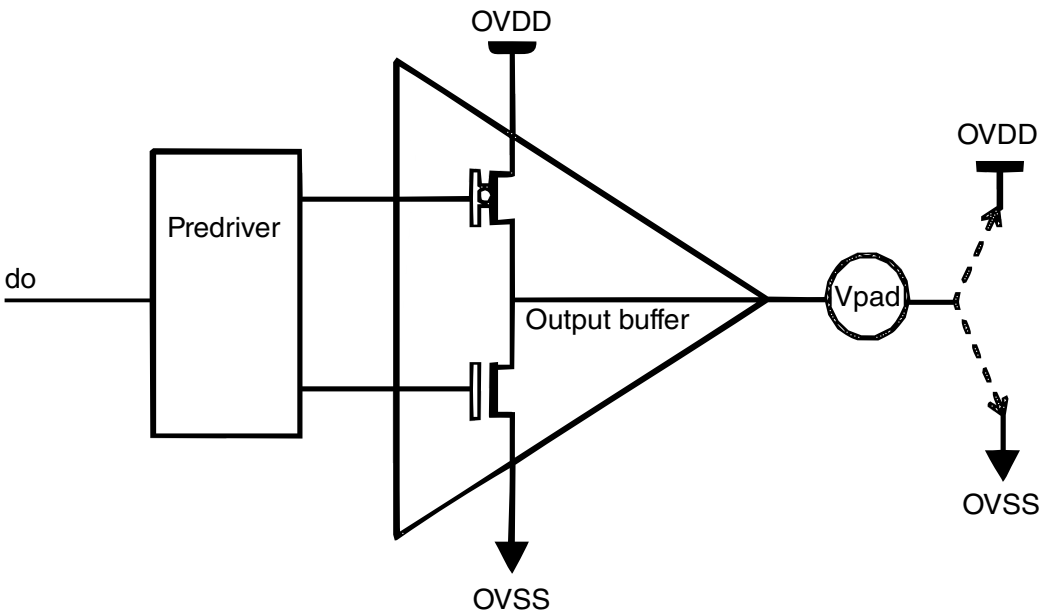


Figure 12-6. Output Driver Functional Diagram

12.3.1.2.1 Drive strength

Drive strength selection can be use to make the impedance matched and get better signal integrity.

12.3.1.2.2 Output keeper

A simple latch to hold the input value. The input value here refers to DO (data out), not data from PAD.

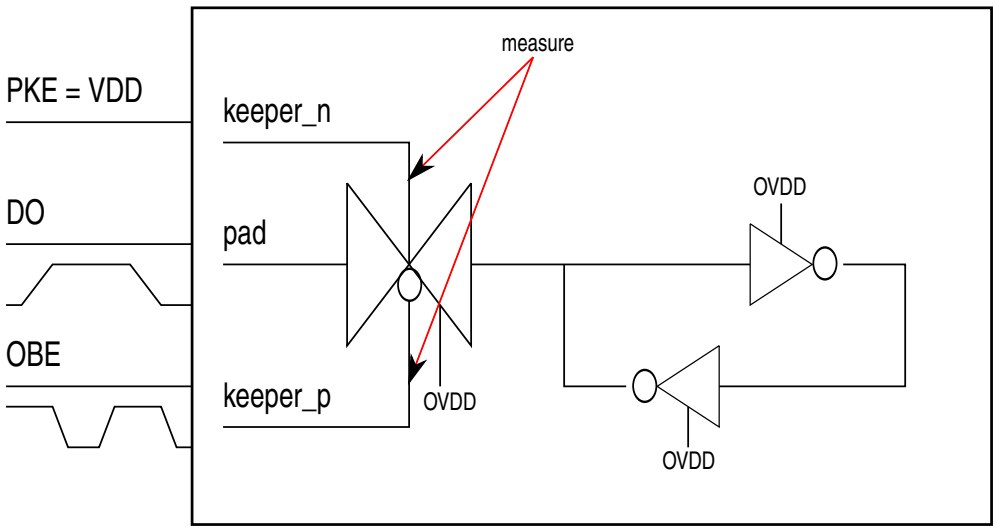


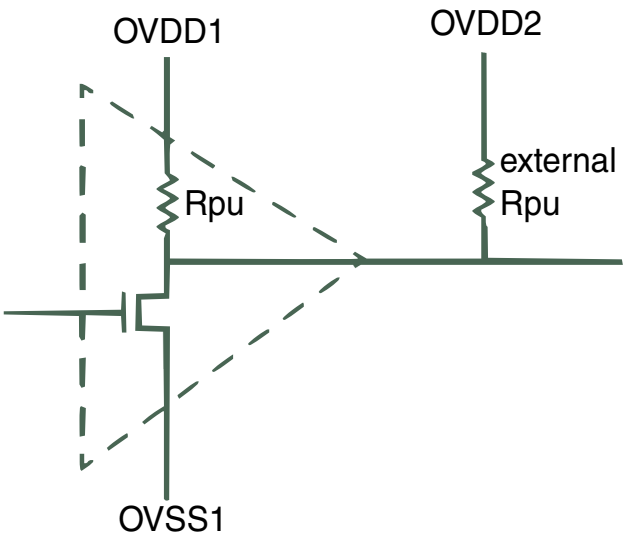
Figure 12-7. Keeper functional diagram

12.3.1.2.3 Pull-up (PU) / Pull-down (PD) / Keeper Logic

When Keeper is enabled, the pull-up and pull-down are disabled, and the output value of the pad depends on the Keeper. The output keeper is powered by OVDD. When the core VDD is powered down or the first inverter is tri-stated, the pad's state can be kept. Keeper and Pull can't be enabled together.

12.3.1.2.4 Open drain

Open drain is a circuit technique which allows multiple devices to communication over a single wire bi-directionally. Open drain drivers usually operate with an external or internal pull-up resistor that holds the signal line high until a device sinks enough current to pull the line low, usually used for a bus with multiple devices.



If internal pull-up resistor (Rpu) is used, output level will depend on OVDD1

If external Rpu is used, output level will depend on OVDD2

Figure 12-8. Output buffer in open drain mode

12.3.1.3 Operating Frequency

Table 12-2. IO Operating Frequency

DSE Setting	SRE / SPEED Setting	R_fixture (Ohm) OVDD = 1.8V nominal	R_fixture (Ohm) OVDD = 3.3V nominal	Operating Frequency (MHz)
1	0	400	220	50
	1			100
	10			100
	11			100
	100			50
	101			100
	110			100
	111			150
10	0	200	110	50
	1			100
	10			100
	11			100
	100			50
	101			100
	110			100
	111			150
11	0	132	73	50
	1			100

Table continues on the next page...

Table 12-2. IO Operating Frequency (continued)

DSE Setting	SRE / SPEED Setting	R_fixture (Ohm) OVDD = 1.8V nominal	R_fixture (Ohm) OVDD = 3.3V nominal	Operating Frequency (MHz)
	10			100
	11			100
	100			50
	101			100
	110			100
	111			150
100	0	104	58	50
	1			100
	10			100
	11			100
	100			50
	101			100
	110			100
	111			150
101	0	83	46	50
	1			100
	10			100
	11			100
	100			50
	101			100 (OVDD=1.8V nom) 150 (OVDD=3.3V nom)
	110			100 (OVDD=1.8V nom) 150 (OVDD=3.3V nom)
	111			150 (OVDD=1.8V nom) 200 (OVDD=3.3V nom)
110	0	70	38	50
	1			100
	10			100
	11			100
	100			50
	101			150
	110			150
	111			200
111	0	55	32	50
	1			100
	10			100
	11			100
	100			50

Table continues on the next page...

Table 12-2. IO Operating Frequency (continued)

DSE Setting	SRE / SPEED Setting	R_fixture (Ohm) OVDD = 1.8V nominal	R_fixture (Ohm) OVDD = 3.3V nominal	Operating Frequency (MHz)
	101			150
	110			150
	111			200

12.3.2 Interrupt Control Unit

In addition to the general-purpose input/output function, the edge-detect logic in the GPIO peripheral reflects whether a transition has occurred on a given GPIO signal that is configured as an input (GDIR[GDIR] = 0). The interrupt control registers (ICR1 and ICR2) may be used to independently configure the interrupt condition of each input signal (low-to-high transition, high-to-low transition, low, or high). For information about ICR1 and ICR2 settings, see the Register section.

The interrupt control unit is built of 32 interrupt control subunits, where each subunit handles a single interrupt line.

12.3.3 Clocks

The following table describes the clock sources for GPIO. Please see the CCM for clock setting, configuration and gating information.

Table 12-3. GPIO Clocks

Clock name	Description
ipg_clk_s	Peripheral access clock

12.4 External Signals

The following table describes the external signals of GPIO.

Table 12-4. GPIO External Signals

Signal	Description
IOn	GPIO Signal

12.5 Application Information

12.5.1 GPIO Read Mode

The programming sequence for reading input signals should be as follows:

1. Configure IOMUX to select GPIO mode (via IOMUX Controller (IOMUXC)).
2. Configure GPIO direction register to input (GDIR[GDIR] = 0b).
3. Read value from data register/pad status register.

A pseudocode description to read [input3:input0] values is as follows:

```
// SET INPUTS TO GPIO MODE.
write sw_mux_ctl_<input0>_<input1>_<input2>_<input3>, 32'h00000000
// SET GDIR TO INPUT.
write GDIR[31:4,input3_bit, input2_bit, input1_bit, input0_bit,] 32'hxxxxxxx0
// READ INPUT VALUE FROM DR.
read DR
// READ INPUT VALUE FROM PSR.
read PSR
```

NOTE

While the GPIO direction is set to input (GDIR[GDIR] = 0), a read access to DR does not return DR data. Instead, it returns the PSR data, which is the corresponding input signal value.

12.5.2 GPIO Write Mode

The programming sequence for driving output signals should be as follows:

1. Configure IOMUX to select GPIO mode (via IOMUXC). If needed, enable SION to read the loopback pad value through PSR.
2. Configure GPIO direction register to output (GDIR[GDIR] = 1b).
3. Write value to data register (DR).

A pseudocode description to drive 4'b0101 on [output3:output0] is as follows:

```
// SET PADS TO GPIO MODE VIA IOMUX.
write sw_mux_ctl_pad_<output[0-3]>.mux_mode, <GPIO_MUX_MODE>
// Enable loopback to capture pad value into PSR in output mode
write sw_mux_ctl_pad_<output[0-3]>.sion, 1
// SET GDIR=1 TO OUTPUT BITS.
write GDIR[31:4,output3_bit,output2_bit, output1_bit, output0_bit,] 32'hxxxxxxxF
// WRITE OUTPUT VALUE=4'b0101 TO DR.
write DR, 32'hxxxxxxx5
// READ OUTPUT VALUE FROM PSR ONLY.
read_cmp PSR, 32'hxxxxxxx5
```

12.6 Memory Map/Register Definition

This section includes the memory map and descriptions of all registers.

12.6.1 GPIO register descriptions

There are eight 32-bit GPIO registers. All registers are accessible from the IP interface. Only 32-bit access is supported.

The GPIO memory map is shown in the following table.

12.6.1.1 GPIO memory map

GPIO1 base address: 401B_8000h

GPIO2 base address: 401B_C000h

GPIO3 base address: 401C_0000h

GPIO4 base address: 401C_4000h

GPIO5 base address: 400C_0000h

GPIO6 base address: 4200_0000h

GPIO7 base address: 4200_4000h

GPIO8 base address: 4200_8000h

GPIO9 base address: 4200_C000h

Offset	Register	Width (In bits)	Access	Reset value
0h	GPIO data register (DR)	32	RW	0000_0000h
4h	GPIO direction register (GDIR)	32	RW	0000_0000h
8h	GPIO pad status register (PSR)	32	RO	0000_0000h
Ch	GPIO interrupt configuration register1 (ICR1)	32	RW	0000_0000h
10h	GPIO interrupt configuration register2 (ICR2)	32	RW	0000_0000h
14h	GPIO interrupt mask register (IMR)	32	RW	0000_0000h
18h	GPIO interrupt status register (ISR)	32	W1C	0000_0000h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
1Ch	GPIO edge select register (EDGE_SEL)	32	RW	0000_0000h
84h	GPIO data register SET (DR_SET)	32	WO	0000_0000h
88h	GPIO data register CLEAR (DR_CLEAR)	32	WO	0000_0000h
8Ch	GPIO data register TOGGLE (DR_TOGGLE)	32	WO	0000_0000h

12.6.1.2 GPIO data register (DR)

12.6.1.2.1 Offset

Register	Offset
DR	0h

12.6.1.2.2 Function

The 32-bit DR register stores data that is ready to be driven to the output lines. If the IOMUXC is in GPIO mode and a given GPIO direction bit is set, then the corresponding DR bit is driven to the output. If a given GPIO direction bit is cleared, then a read of DR register reflects the value of the corresponding signal. Two wait states are required in read access for synchronization.

The results of a read of a DR bit depends on the IOMUXC input mode settings and the corresponding GDIR bit as follows:

- If GDIR[*n*] is set and IOMUXC input mode is GPIO, then reading DR[*n*] returns the contents of DR[*n*].
- If GDIR[*n*] is cleared and IOMUXC input mode is GPIO, then reading DR[*n*] returns the corresponding input signal's value.
- If GDIR[*n*] is set and IOMUXC input mode is not GPIO, then reading DR[*n*] returns the contents of DR[*n*].
- If GDIR[*n*] is cleared and IOMUXC input mode is not GPIO, then reading DR[*n*] always returns zero.

12.6.1.2.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

12.6.1.2.4 Fields

Field	Function
31-0 DR	DR data bits This register defines the value of the GPIO output when the signal is configured as an output (GDIR[n]=1). Writes to this register are stored in a register. Reading DR register returns the value stored in the register if the signal is configured as an output (GDIR[n]=1), or the input signal's value if configured as an input (GDIR[n]=0). **NOTE:** The IOMUXC must be configured to GPIO mode for the DR register value to connect with the signal. Reading the data register with the input path disabled always returns a zero value.

12.6.1.3 GPIO direction register (GDIR)

12.6.1.3.1 Offset

Register	Offset
GDIR	4h

12.6.1.3.2 Function

The GDIR register functions as direction control when the IOMUXC is in GPIO mode. Each bit specifies the direction of a one-bit signal. The mapping of each DIR bit to a corresponding SoC signal is determined by the SoC's pin assignment and the IOMUX table. For more details consult the IOMUXC chapter.

12.6.1.3.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	GDIR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GDIR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

12.6.1.3.4 Fields

Field	Function
31-0 GDIR	GPIO direction bits Bit n of this register defines the direction of the GPIO[n] signal. **NOTE:** The GDIR register affects only the direction of the I/O signal when the corresponding bit in the IOMUXC is configured for GPIO. 0b - GPIO is configured as input. 1b - GPIO is configured as output.

12.6.1.4 GPIO pad status register (PSR)

12.6.1.4.1 Offset

Register	Offset
PSR	8h

12.6.1.4.2 Function

The PSR register is a read-only register. Each bit stores the value of the corresponding input signal (as configured in the IOMUX). This register is clocked with the ipg_clk_s clock, meaning that the input signal is sampled only when accessing this location. Two wait states are required any time this register is accessed for synchronization.

12.6.1.4.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PSR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PSR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

12.6.1.4.4 Fields

Field	Function
31-0 PSR	GPIO pad status bits Reading the PSR register returns the state of the corresponding input signal. **NOTE:** The IOMUXC must be configured to GPIO mode for the PSR register to reflect the state of the corresponding signal.

12.6.1.5 GPIO interrupt configuration register1 (ICR1)

12.6.1.5.1 Offset

Register	Offset
ICR1	Ch

12.6.1.5.2 Function

The ICR1 register contains 16 two-bit fields, where each field specifies the interrupt configuration for a different input signal.

12.6.1.5.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ICR15		ICR14		ICR13		ICR12		ICR11		ICR10		ICR9		ICR8	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ICR7		ICR6		ICR5		ICR4		ICR3		ICR2		ICR1		ICR0	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

12.6.1.5.4 Fields

Field	Function
31-30 ICR15	Interrupt configuration field for GPIO interrupt 15 This bit field controls the active condition of the interrupt function for GPIO interrupt 15. 00b - Interrupt 15 is low-level sensitive. 01b - Interrupt 15 is high-level sensitive. 10b - Interrupt 15 is rising-edge sensitive. 11b - Interrupt 15 is falling-edge sensitive.
29-28 ICR14	Interrupt configuration field for GPIO interrupt 14 This bit field controls the active condition of the interrupt function for GPIO interrupt 14. 00b - Interrupt 14 is low-level sensitive. 01b - Interrupt 14 is high-level sensitive. 10b - Interrupt 14 is rising-edge sensitive. 11b - Interrupt 14 is falling-edge sensitive.
27-26 ICR13	Interrupt configuration field for GPIO interrupt 13 This bit field controls the active condition of the interrupt function for GPIO interrupt 13. 00b - Interrupt 13 is low-level sensitive. 01b - Interrupt 13 is high-level sensitive. 10b - Interrupt 13 is rising-edge sensitive. 11b - Interrupt 13 is falling-edge sensitive.
25-24 ICR12	Interrupt configuration field for GPIO interrupt 12 This bit field controls the active condition of the interrupt function for GPIO interrupt 12. 00b - Interrupt 12 is low-level sensitive. 01b - Interrupt 12 is high-level sensitive. 10b - Interrupt 12 is rising-edge sensitive. 11b - Interrupt 12 is falling-edge sensitive.
23-22 ICR11	Interrupt configuration field for GPIO interrupt 11 This bit field controls the active condition of the interrupt function for GPIO interrupt 11. 00b - Interrupt 11 is low-level sensitive. 01b - Interrupt 11 is high-level sensitive. 10b - Interrupt 11 is rising-edge sensitive. 11b - Interrupt 11 is falling-edge sensitive.

Table continues on the next page...

Field	Function
21-20	Interrupt configuration field for GPIO interrupt 10
ICR10	This bit field controls the active condition of the interrupt function for GPIO interrupt 10. 00b - Interrupt 10 is low-level sensitive. 01b - Interrupt 10 is high-level sensitive. 10b - Interrupt 10 is rising-edge sensitive. 11b - Interrupt 10 is falling-edge sensitive.
19-18	Interrupt configuration field for GPIO interrupt 9
ICR9	This bit field controls the active condition of the interrupt function for GPIO interrupt 9. 00b - Interrupt 9 is low-level sensitive. 01b - Interrupt 9 is high-level sensitive. 10b - Interrupt 9 is rising-edge sensitive. 11b - Interrupt 9 is falling-edge sensitive.
17-16	Interrupt configuration field for GPIO interrupt 8
ICR8	This bit field controls the active condition of the interrupt function for GPIO interrupt 8. 00b - Interrupt 8 is low-level sensitive. 01b - Interrupt 8 is high-level sensitive. 10b - Interrupt 8 is rising-edge sensitive. 11b - Interrupt 8 is falling-edge sensitive.
15-14	Interrupt configuration field for GPIO interrupt 7
ICR7	This bit field controls the active condition of the interrupt function for GPIO interrupt 7. 00b - Interrupt 7 is low-level sensitive. 01b - Interrupt 7 is high-level sensitive. 10b - Interrupt 7 is rising-edge sensitive. 11b - Interrupt 7 is falling-edge sensitive.
13-12	Interrupt configuration field for GPIO interrupt 6
ICR6	This bit field controls the active condition of the interrupt function for GPIO interrupt 6. 00b - Interrupt 6 is low-level sensitive. 01b - Interrupt 6 is high-level sensitive. 10b - Interrupt 6 is rising-edge sensitive. 11b - Interrupt 6 is falling-edge sensitive.
11-10	Interrupt configuration field for GPIO interrupt 5
ICR5	This bit field controls the active condition of the interrupt function for GPIO interrupt 5. 00b - Interrupt 5 is low-level sensitive. 01b - Interrupt 5 is high-level sensitive. 10b - Interrupt 5 is rising-edge sensitive. 11b - Interrupt 5 is falling-edge sensitive.
9-8	Interrupt configuration field for GPIO interrupt 4
ICR4	This bit field controls the active condition of the interrupt function for GPIO interrupt 4. 00b - Interrupt 4 is low-level sensitive. 01b - Interrupt 4 is high-level sensitive. 10b - Interrupt 4 is rising-edge sensitive. 11b - Interrupt 4 is falling-edge sensitive.
7-6	Interrupt configuration field for GPIO interrupt 3
ICR3	This bit field controls the active condition of the interrupt function for GPIO interrupt 3. 00b - Interrupt 3 is low-level sensitive. 01b - Interrupt 3 is high-level sensitive.

Table continues on the next page...

Field	Function
	10b - Interrupt 3 is rising-edge sensitive. 11b - Interrupt 3 is falling-edge sensitive.
5-4 ICR2	Interrupt configuration field for GPIO interrupt 2 This bit field controls the active condition of the interrupt function for GPIO interrupt 2. 00b - Interrupt 2 is low-level sensitive. 01b - Interrupt 2 is high-level sensitive. 10b - Interrupt 2 is rising-edge sensitive. 11b - Interrupt 2 is falling-edge sensitive.
3-2 ICR1	Interrupt configuration field for GPIO interrupt 1 This bit field controls the active condition of the interrupt function for GPIO interrupt 1. 00b - Interrupt 1 is low-level sensitive. 01b - Interrupt 1 is high-level sensitive. 10b - Interrupt 1 is rising-edge sensitive. 11b - Interrupt 1 is falling-edge sensitive.
1-0 ICR0	Interrupt configuration field for GPIO interrupt 0 This bit field controls the active condition of the interrupt function for GPIO interrupt 0. 00b - Interrupt 0 is low-level sensitive. 01b - Interrupt 0 is high-level sensitive. 10b - Interrupt 0 is rising-edge sensitive. 11b - Interrupt 0 is falling-edge sensitive.

12.6.1.6 GPIO interrupt configuration register2 (ICR2)

12.6.1.6.1 Offset

Register	Offset
ICR2	10h

12.6.1.6.2 Function

The ICR2 register contains 16 two-bit fields, where each field specifies the interrupt configuration for a different input signal.

12.6.1.6.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ICR31		ICR30		ICR29		ICR28		ICR27		ICR26		ICR25		ICR24	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ICR23		ICR22		ICR21		ICR20		ICR19		ICR18		ICR17		ICR16	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

12.6.1.6.4 Fields

Field	Function
31-30 ICR31	Interrupt configuration field for GPIO interrupt 31 This bit field controls the active condition of the interrupt function for GPIO interrupt 31. 00b - Interrupt 31 is low-level sensitive. 01b - Interrupt 31 is high-level sensitive. 10b - Interrupt 31 is rising-edge sensitive. 11b - Interrupt 31 is falling-edge sensitive.
29-28 ICR30	Interrupt configuration field for GPIO interrupt 30 This bit field controls the active condition of the interrupt function for GPIO interrupt 30. 00b - Interrupt 30 is low-level sensitive. 01b - Interrupt 30 is high-level sensitive. 10b - Interrupt 30 is rising-edge sensitive. 11b - Interrupt 30 is falling-edge sensitive.
27-26 ICR29	Interrupt configuration field for GPIO interrupt 29 This bit field controls the active condition of the interrupt function for GPIO interrupt 29. 00b - Interrupt 29 is low-level sensitive. 01b - Interrupt 29 is high-level sensitive. 10b - Interrupt 29 is rising-edge sensitive. 11b - Interrupt 29 is falling-edge sensitive.
25-24 ICR28	Interrupt configuration field for GPIO interrupt 28 This bit field controls the active condition of the interrupt function for GPIO interrupt 28. 00b - Interrupt 28 is low-level sensitive. 01b - Interrupt 28 is high-level sensitive. 10b - Interrupt 28 is rising-edge sensitive. 11b - Interrupt 28 is falling-edge sensitive.
23-22 ICR27	Interrupt configuration field for GPIO interrupt 27 This bit field controls the active condition of the interrupt function for GPIO interrupt 27. 00b - Interrupt 27 is low-level sensitive. 01b - Interrupt 27 is high-level sensitive. 10b - Interrupt 27 is rising-edge sensitive. 11b - Interrupt 27 is falling-edge sensitive.

Table continues on the next page...

Field	Function
21-20	Interrupt configuration field for GPIO interrupt 26
ICR26	This bit field controls the active condition of the interrupt function for GPIO interrupt 26. 00b - Interrupt 26 is low-level sensitive. 01b - Interrupt 26 is high-level sensitive. 10b - Interrupt 26 is rising-edge sensitive. 11b - Interrupt 26 is falling-edge sensitive.
19-18	Interrupt configuration field for GPIO interrupt 25
ICR25	This bit field controls the active condition of the interrupt function for GPIO interrupt 25. 00b - Interrupt 25 is low-level sensitive. 01b - Interrupt 25 is high-level sensitive. 10b - Interrupt 25 is rising-edge sensitive. 11b - Interrupt 25 is falling-edge sensitive.
17-16	Interrupt configuration field for GPIO interrupt 24
ICR24	This bit field controls the active condition of the interrupt function for GPIO interrupt 24. 00b - Interrupt 24 is low-level sensitive. 01b - Interrupt 24 is high-level sensitive. 10b - Interrupt 24 is rising-edge sensitive. 11b - Interrupt 24 is falling-edge sensitive.
15-14	Interrupt configuration field for GPIO interrupt 23
ICR23	This bit field controls the active condition of the interrupt function for GPIO interrupt 23. 00b - Interrupt 23 is low-level sensitive. 01b - Interrupt 23 is high-level sensitive. 10b - Interrupt 23 is rising-edge sensitive. 11b - Interrupt 23 is falling-edge sensitive.
13-12	Interrupt configuration field for GPIO interrupt 22
ICR22	This bit field controls the active condition of the interrupt function for GPIO interrupt 22. 00b - Interrupt 22 is low-level sensitive. 01b - Interrupt 22 is high-level sensitive. 10b - Interrupt 22 is rising-edge sensitive. 11b - Interrupt 22 is falling-edge sensitive.
11-10	Interrupt configuration field for GPIO interrupt 21
ICR21	This bit field controls the active condition of the interrupt function for GPIO interrupt 21. 00b - Interrupt 21 is low-level sensitive. 01b - Interrupt 21 is high-level sensitive. 10b - Interrupt 21 is rising-edge sensitive. 11b - Interrupt 21 is falling-edge sensitive.
9-8	Interrupt configuration field for GPIO interrupt 20
ICR20	This bit field controls the active condition of the interrupt function for GPIO interrupt 20. 00b - Interrupt 20 is low-level sensitive. 01b - Interrupt 20 is high-level sensitive. 10b - Interrupt 20 is rising-edge sensitive. 11b - Interrupt 20 is falling-edge sensitive.
7-6	Interrupt configuration field for GPIO interrupt 19
ICR19	This bit field controls the active condition of the interrupt function for GPIO interrupt 19. 00b - Interrupt 19 is low-level sensitive. 01b - Interrupt 19 is high-level sensitive.

Table continues on the next page...

Field	Function
	10b - Interrupt 19 is rising-edge sensitive. 11b - Interrupt 19 is falling-edge sensitive.
5-4 ICR18	Interrupt configuration field for GPIO interrupt 18 This bit field controls the active condition of the interrupt function for GPIO interrupt 18. 00b - Interrupt 18 is low-level sensitive. 01b - Interrupt 18 is high-level sensitive. 10b - Interrupt 18 is rising-edge sensitive. 11b - Interrupt 18 is falling-edge sensitive.
3-2 ICR17	Interrupt configuration field for GPIO interrupt 17 This bit field controls the active condition of the interrupt function for GPIO interrupt 17. 00b - Interrupt 17 is low-level sensitive. 01b - Interrupt 17 is high-level sensitive. 10b - Interrupt 17 is rising-edge sensitive. 11b - Interrupt 17 is falling-edge sensitive.
1-0 ICR16	Interrupt configuration field for GPIO interrupt 16 This bit field controls the active condition of the interrupt function for GPIO interrupt 16. 00b - Interrupt 16 is low-level sensitive. 01b - Interrupt 16 is high-level sensitive. 10b - Interrupt 16 is rising-edge sensitive. 11b - Interrupt 16 is falling-edge sensitive.

12.6.1.7 GPIO interrupt mask register (IMR)

12.6.1.7.1 Offset

Register	Offset
IMR	14h

12.6.1.7.2 Function

The IMR register contains masking bits for each interrupt line.

12.6.1.7.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	IMR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IMR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

12.6.1.7.4 Fields

Field	Function
31-0 IMR	Interrupt Mask bits This register is used to enable or disable the interrupt function on each of the 32 GPIO signals. Bit IMR[n] (n=0...31) controls interrupt n as follows: 0b - Interrupt n is disabled. 1b - Interrupt n is enabled.

12.6.1.8 GPIO interrupt status register (ISR)

12.6.1.8.1 Offset

Register	Offset
ISR	18h

12.6.1.8.2 Function

The ISR register functions as an interrupt status indicator. Each bit indicates whether an interrupt condition has been met for the corresponding input signal. When an interrupt condition is met (as determined by the corresponding interrupt condition register field), the corresponding bit in this register is set. Two wait states are required in read access for synchronization. One wait state is required for reset.

12.6.1.8.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	ISR															
W	W1C															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ISR															
W	W1C															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

12.6.1.8.4 Fields

Field	Function
31-0	Interrupt status bits
ISR	Bit n of this register is asserted (active high) when the active condition (as determined by the corresponding ICR bit) is detected on the GPIO input and is waiting for service. The value of this register is independent of the value in IMR register. When the active condition has been detected, the corresponding bit remains set until cleared by software. Status flags are cleared by writing a 1 to the corresponding bit position.

12.6.1.9 GPIO edge select register (EDGE_SEL)

12.6.1.9.1 Offset

Register	Offset
EDGE_SEL	1Ch

12.6.1.9.2 Function

The EDGE_SEL register may be used to override the ICR registers' configuration. If the GPIO_EDGE_SEL bit is set, then a rising edge or falling edge in the corresponding signal generates an interrupt. This register provides backward compatibility. On reset all bits are cleared (ICR is not overridden).

12.6.1.9.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	GPIO_EDGE_SEL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GPIO_EDGE_SEL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

12.6.1.9.4 Fields

Field	Function
31-0 GPIO_EDGE_SEL	Edge select When EDGE_SEL[*n*] is set, the GPIO disregards the ICR[*n*] setting, and detects any edge on the corresponding input signal.

12.6.1.10 GPIO data register SET (DR_SET)

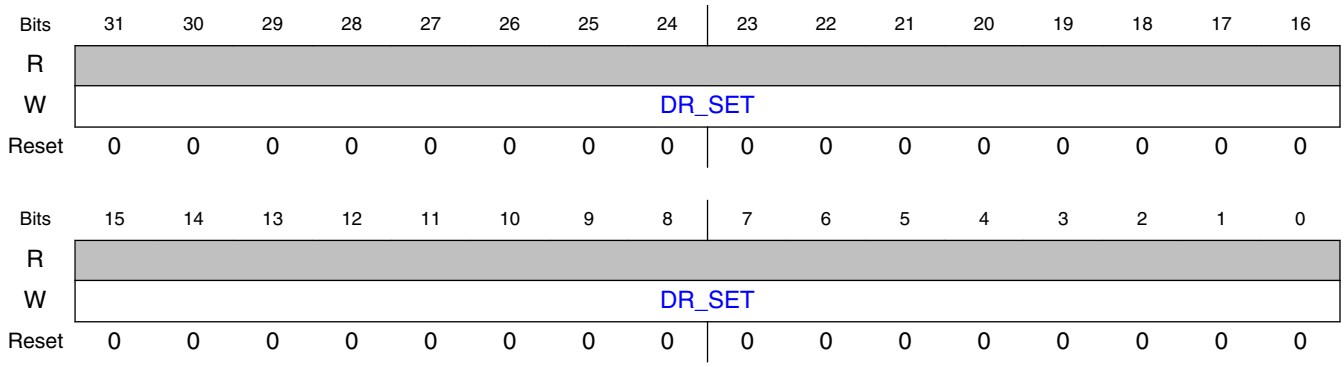
12.6.1.10.1 Offset

Register	Offset
DR_SET	84h

12.6.1.10.2 Function

The SET register of DR.

12.6.1.10.3 Diagram



12.6.1.10.4 Fields

Field	Function
31-0 DR_SET	Set Writing a 1 to a bit in this register sets the corresponding bit in DR

12.6.1.11 GPIO data register CLEAR (DR_CLEAR)

12.6.1.11.1 Offset

Register	Offset
DR_CLEAR	88h

12.6.1.11.2 Function

The CLEAR register of DR.

12.6.1.11.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	DR_CLEAR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	DR_CLEAR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

12.6.1.11.4 Fields

Field	Function
31-0	Clear
DR_CLEAR	Writing a 1 to a bit in this register clears the corresponding bit in DR

12.6.1.12 GPIO data register TOGGLE (DR_TOGGLE)

12.6.1.12.1 Offset

Register	Offset
DR_TOGGLE	8Ch

12.6.1.12.2 Function

The TOGGLE register of DR.

12.6.1.12.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	DR_TOGGLE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	DR_TOGGLE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

12.6.1.12.4 Fields

Field	Function
31-0 DR_TOGGLE	Toggle Writing a 1 to a bit in this register toggles the corresponding bit in DR

Chapter 13
Clock and Power Management

13.1 Introduction

This chapter describes the Clock and Power Management architecture of the SoC.

The chip targets applications where low power consumption, long battery life, always-on and instant-on capabilities are paramount, and where there is no need for active cooling. To achieve these capabilities, the primary focus of the chip's design is reducing current consumption as much as possible, while simultaneously enabling the maximum level of peak performance and a balanced level of sustained performance for target applications. To achieve this, the chip architecture uses a wide range of power-management techniques and their combinations for maximum system design flexibility.

This chapter contains information about:

- Structural components of the power and clock management systems of the chip
- Power, clock and thermal management techniques supported by the chip

All given numerical values are typical or examples. For accurate values one should refer to the datasheet.

13.2 Device Power Management Architecture Components

To provide a clean and versatile architecture supporting a wide range of power-management techniques, clocks, and power rails are managed resources.

For each rail, two levels of management are defined: the first level is centralized or SoC-level resource management, and the second is a local or "module level" resource management.

The high level architectural view of the clock, power and thermal management system of the chip is presented in the figure below.

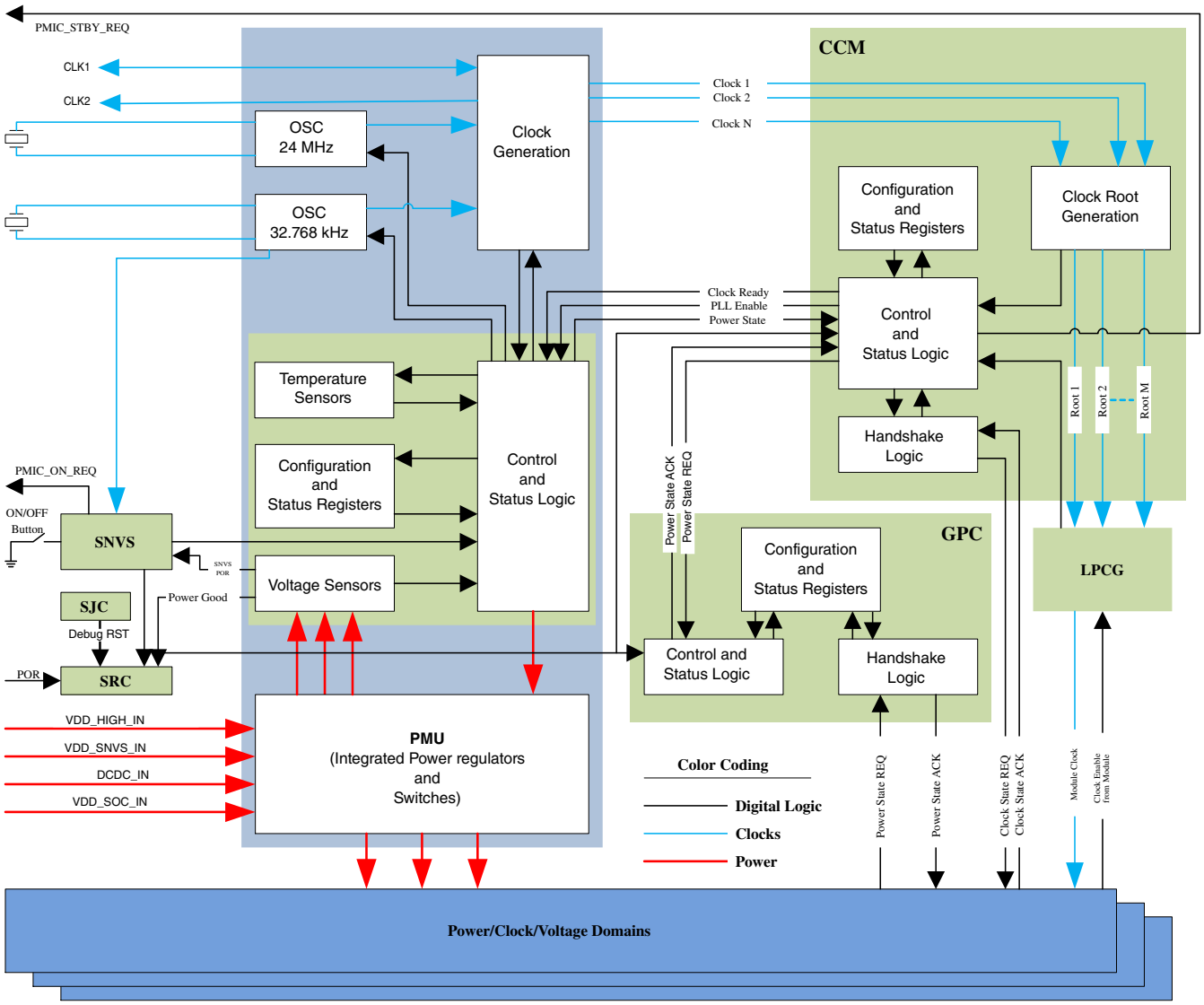


Figure 13-1. Power and clock management framework

13.2.1 Centralized components of clock generation and management

Centralized components of the clock generation and management sub-system are implemented in the following blocks:

- CCM (Clock Control Module): The CCM module provides control for primary (source level) and secondary (root level) clock generation, division, distribution, synchronization, and coarse-level gating. See Clock Controller Module (CCM)) for

information on the CCM architecture, functional description and programming model.
- LPCG (Low Power Clock Gating): This module distributes the clocks to all blocks in the SoC and handles block level software-controllable and automated clock gating. See Clock Controller Module (CCM) for information on the LPCG architecture and functional description.

13.2.2 Centralized components of power generation, distribution and management

Centralized components of the power generation, distribution and management sub-system are implemented in the following blocks:

- Power Management Unit (PMU). See Power Management Unit (PMU) for information on the PMU architecture, functional description and programming model.
- General Power Controller (GPC). See General Power Controller (GPC) for information on the GPC architecture, functional description and programming model.
- On Chip DC-DC regulator. See the DCDC chapter for more detailed information.

13.2.3 Reset generation and distribution system

Power and clock management are accompanied with an appropriate reset generation and distribution system, centralized functions of which are implemented by the System Reset Controller (SRC).

13.2.4 Power and clock management framework

Together, the modules listed above provide enhanced power-management features with centralized control for the clock, reset, and power-management signals on the SoC.

The centralized management framework defines the managed components of the power-management architecture. These components are called the clock, power, and voltage domains.

NOTE

A domain is a group of modules or functional blocks that share a common resource entity (for example, common clock root, common power source, or a common power switch). The software component managing shared resources should take

into account the joint constraints of all the modules belonging to that resource domain.

13.3 Clock Management

13.3.1 Centralized components of clock management system

The clock generation and management system is built around the CCM and LPCG blocks.

A high level block diagram of the clock management system in the SoC environment is shown in the following figure.

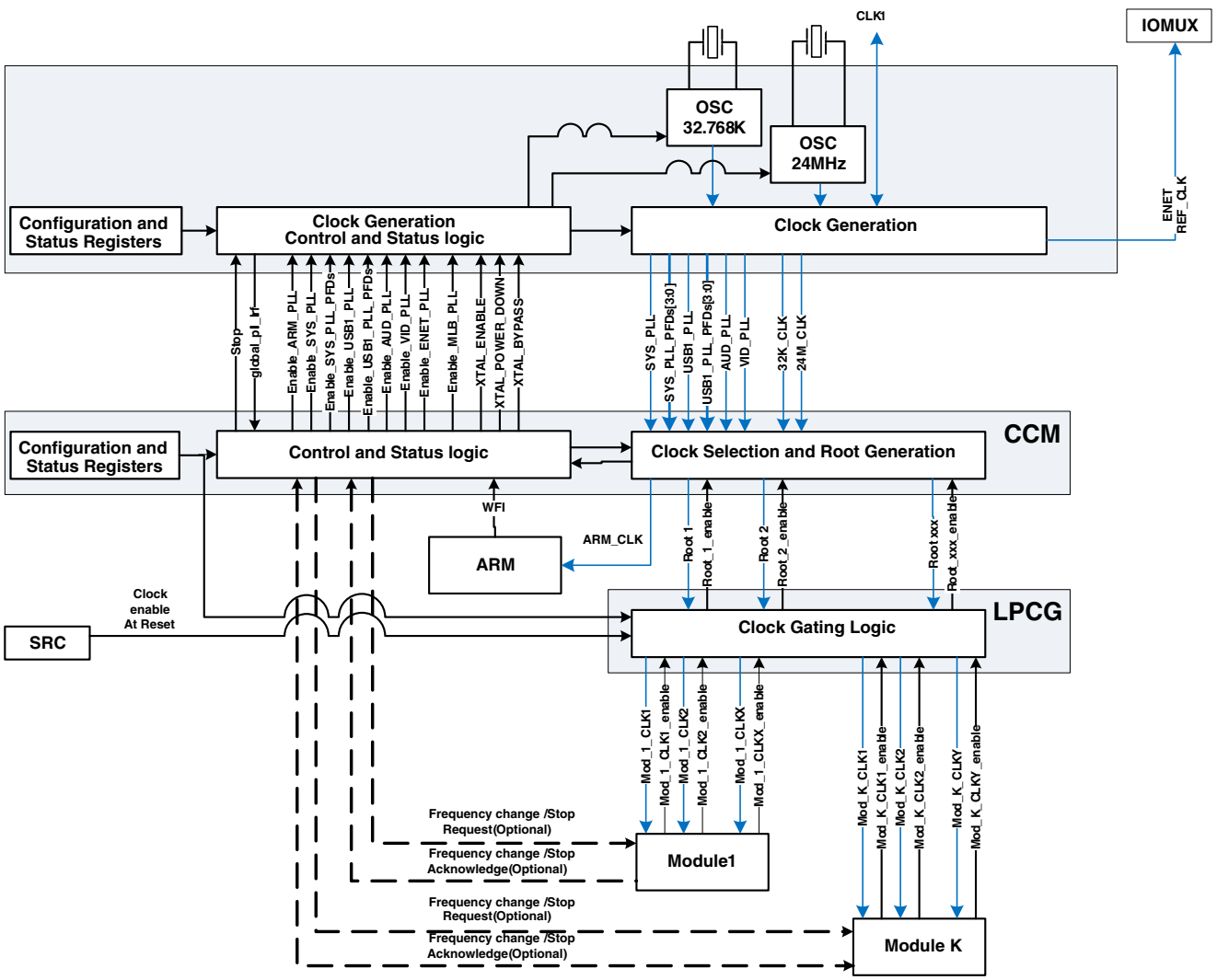


Figure 13-2. Clock Management System

13.3.2 Clock generation

The clock generation section includes the components detailed in the following sections.

13.3.2.1 Crystal Oscillator (XTALOSC)

The Crystal Oscillator block is comprised of both the high frequency oscillator (typical frequency is 24 MHz) and the low frequency real time clock oscillator (typical frequency of 32.768 KHz). Each of these oscillators is implemented as a biased amplifier that, when combined with a suitable external quartz crystal and external load capacitors, implements an oscillator. See Crystal Oscillator (XTALOSC) for details of the XTALOSC block.

NOTE

1. The 24 MHz XTALOSC can be the external clock source of SYSTICK. Hardware devides this down to 100KHz before it reaches SYSTICK.
2. If XTALOSC24M_LOWPWR_CTRLn[OSC_SEL] is set to ROSC, the chip's 24MHz clock will be from the 24MHz RCOSC. In this case, the RCOSC should be enabled properly beforehand.

13.3.2.2 PLLs

Seven PLLs are included in the clock generation section. Two of these PLLs are each equipped with four Phase Fractional Dividers (PFDs) in order to generate additional frequencies.

NOTE

Each PFD works independently by interpolating the VCO of the PLL to which it is connected. It effectively takes the PLL VCO frequency and produces 18/N x Fvco at its output where N ranges from 12 to 35. PFD is a completely digital design with no analog components or feedback loops. The frequency switch time is much faster than a PLL because keeping the base PLL locked and changing the integer N only changes the logical combination of the interpolated outputs of the VCO. Note that the PFD not only enables faster frequency changes than a PLL, but also allows the configuration to be safely changed "on-the-fly" without going through the output clock disabling/enabling process.

The seven PLLs are listed below:

- PLL1 (also referred to as ARM_PLL) - This is the PLL clocking the Arm core complex. It is a programmable integer frequency multiplier capable of output

frequency of up to 1.3 GHz. Note that this frequency is higher than the maximum chip supported frequency.

- PLL2 (also referred tp as System_PLL or 528_PLL) - PLL2 runs at a fixed multiplier of 22, producing 528 MHz output frequency with 24 MHz reference from XTALOSC. Besides the main output, this PLL drives four PFDs (PLL2_PFD0...PLL2_PFD3). The main PLL2 output and its PFD outputs are used as inputs for many clock roots. These do not require exact/constant frequency and can be changed as a part of dynamic frequency scaling procedure. Typically, this PLL or its PFDs are a clock source for internal system buses, internal processing logic, SDRAM interface, NAND/NOR interface modules, etc.
- PLL3 (also referred to as USB1_PLL) - PLL3 is used in conjunction with the first instance of USB PHY (USBPHY1, also known as OTG PHY). This PLL drives four PFDs (PLL3_PFD0...PLL3_PFD3) and runs at a fixed multiplier of 20. This results in a VCO frequency of 480 MHz with a 24 MHz oscillator. The main PLL3 output and its PFD outputs are used as inputs for many clock roots that require constant frequency, such as UART and other serial interfaces, audio interfaces, etc.
- PLL4 (also referred to as an Audio PLL) - This is a fractional multiplier PLL used for generating a low jitter and high precision audio clock with standardized audio frequencies. The PLLs oscillator frequency range is from 650 MHz to 1300 MHz, and the frequency resolution is better than 1 Hz. This clock is mainly used as a clock for serial audio interfaces and as a reference clock for external audio codecs. It is equipped with a divider on its output and can generate divided by 1, 2 or 4 from the PLL VCO frequency.
- PLL5 (also referred to as a Video PLL) - This is a fractional multiplier PLL used for generating a low jitter and high precision video clock with standardized video frequencies. The PLLs oscillator frequency range is from 650 MHz to 1300 MHz, and the frequency resolution is better than 1 Hz. This clock is mainly used as a clock for display and video interfaces. It is equipped with dividers on its output and can generate clock divided by 1, 2, 4, 8 or16 from the PLL VCO frequency
- PLL6 (also referred to as ENET_PLL) - This PLL implements a fixed 20+(5/6) multiplier. With a 24 MHz input, it has a VCO frequency of 500 MHz. This PLL is used to generate:
  - 50 or 25 MHz for the external ethernet interface.
  - 125 MHz for reduced gigabit ethernet interface.
  - 100 MHz for general purposes.
- PLL7 (also referred to as USB2_PLL) - This PLL provides clock exclusively to USB2 PHY (USBPHY2, also known as OTG PHY). It runs at a fixed multiplier of 20, resulting in a VCO frequency of 480 MHz with a 24 MHz oscillator.

13.3.2.2.1 General PLL Control and Status Functions

Each PLLs configuration and control functions are accessible individually through its PFDs and global configuration and status registers.

Reference input clock for any of the PLLs could be selected individually by the BYPASS_CLK_SRC field of the PLL control register. See CCM Analog Memory Map/ Register Definition for more information.

Each of the PLLs could be individually configured to "Bypass", "Output disabled" and "Power Down" modes.

When configured in "Bypass" PLL pass directly its input reference clocks to the PLL output. Bypassing the PLL is done by setting the BYPASS bit in the control register. For the PLL equipped with PFDs the input reference clock is also bypassed to all PFDs outputs.

When configured in output disabled mode (ENABLE=0), the PLL's output is completely gated and there is neither a bypass clock nor PLL generated clock that propagates to PLL output. Each PLL output has an individual "Output Enable" control bit. The PFDs are gated by the ENABLE bit of their associated PLL. Each PFD does have an associated clock gate bit that can be used to turn it off individually.

When configured in "Power Down mode" most of the PLL circuitry is switched off. Neither main PLL output nor PFD outputs are available in this mode.

When the related PLL is powered up from the power down state or made to go through a relock cycle due to PLL reprogramming, it is required that the related PFDx_CLKGATE bit in CCM_ANALOG_PFD_480n or CCM_ANALOG_PFD_528n, be cycled on and off (1 to 0) after PLL lock. The PFDs can be in the clock gated state during PLL relock but must be clock un-gated only after lock is achieved. See the engineering bulletin, Configuration of Phase Fractional Dividers (EB790) at www.nxp.com for procedure details.

Individual PLL status is reflected in "PLL Lock" bits of the PLL control registers. PLL enable logic which monitors the register value change is implemented to gate off the PLL outputs during the "lock in" period.

Outputs are generated to be sent out by monitoring the individual PLL lock flags and filtering out any random initial edges.

Individual PLL Lock ready flags are first "ORED" with "enables" and then "ANDED" together to generate the global PLL lock ready flag that reflects status of all PLLs enabled in certain moment.

CCM Memory Map/Register Definition and CCM Analog Memory Map/Register Definition contains detailed descriptions of the memory mapped registers and control functions of the clock generation sub-module.

13.3.2.3 CCM

CCM includes:

- Clock root generation logic - This sub-block provides the registers that control most of the secondary clock source programming, including both the primary clock source selection and the clock dividers. The clock roots are each individual clocks to the core, system buses (AXI, AHB, IPG) and all other SoC peripherals, among those are serial clocks, baud clocks, and special functional clocks. Most of clock roots are specific per module.
- CCM, in coordination with GPC, PMU and SRC, manages the Power modes, namely RUN, WAIT and STOP modes. The gating of the peripheral clocks is programmable in RUN and WAIT modes.

CCM manages the frequency scaling procedure for:

- Arm core clock - "on the fly" without clock interruption, by either shifting between PLL sources PLL clock change or by changing the divider ratio.
- Peripheral root clock - by using programmable divider. The division factor can change on the fly without loss of clocks.

NOTE

On-the-fly frequency changing for synchronous interfaces like serial audio interfaces, video and display interfaces, or general purpose serial interfaces (e.g. UART, CAN) may cause synchronization loss and should not be done.

13.3.2.4 Low Power Clock Gating unit (LPCG)

The LPCG block receives the root clocks from CCM and splits them to clock branches for each block. The clock branches are individually gated clocks.

The enables for those gates can come from three sources:

- Clock enable signal from CCM - This signal is generated depending on the power mode the system is in. For each power mode, it is defined in the software using the configuration of the CGR bits in CCM.

- Clock enable signal from the block - This signal is generated by the block based on its internal logic. Not every enable signal from the block is used. Each clock enable signal from the block can be overridden based on the programmable bit in CCM.
- Clock enable signal from the reset controller (SRC) - This signal will enable the clock during the reset procedure.

13.3.3 Peripheral components of clock management system

13.3.3.1 Interface and functional clock

Each block within the SoC has specific clock input characteristic requirements. Based on the characteristics of the clocks delivered to modules, the clocks are divided into two categories: bus interface clocks and functional clocks.

The bus interface clocks have the following characteristics:

- They ensure proper communication between any block/subsystem and the system buses.
- In most cases, they supply the system interface and configuration registers of the block.
- A typical block has one system bus clock, but blocks with multiple interface clocks may also exist (that is, when a block is connected to multiple buses).
- The bus interface clocks are always fed by the outputs of the CCM/LPCG.
- Clock management for this type of clock is always implemented at the system level because it requires coordinated clock management between the block and system buses.

Functional clocks have the following characteristics:

- They supply the functional part of a block or a subsystem.
- Typically, these clocks are completely asynchronous and independent from the bus interface clock of the same block.
- A block can have one or more functional clocks. Some functional clocks are mandatory, while others are optional for its functioning. A block needs its mandatory clock(s) to be operational. The optional clocks are used for specific features and can be shut down without stopping the block activity.
- The functional clocks are fed either by a CCM/LPCG block functional clock output, or by some other clock source, such as a clock output of another block or an external signal coming from IOMUX.

13.3.3.2 Block level clock management

Each block in the system may also have specific clock requirements. Certain module clocks must be active when operating in some specific modes, or may be gated in some others. Generally, the activation and gating of the module clocks are managed by LPCG. Hence, the LPCG block must be programmed properly and, in case of hardware controllable clock gating, peripheral module should provide signals indicating when to activate and when to gate the module clocks.

The LPCG block differentiates the clock-management behavior for device modules based on whether the block can initiate transactions on the device interconnect (called master module), or if it cannot initiate transactions and only responds to the transactions initiated by the master (called slave module). Thus, two hardware-based clock-management protocols are used:

- Master protocol - Clock-management protocol between the CCM/LPCG and blocks that can be bus master
- Slave protocol - Clock-management protocol between the CCM/LPCG and slave modules

13.3.3.2.1 Master clock protocol

This protocol is used to indicate that a master module is ready to initiate a transaction on the device interconnect and requests specific (both functional and interface) clocks. The CCM/LPCG block ensures that the required clocks are active when the master module requests that the CCM/LPCG enable them. The module is said to be functional after the required clocks are activated.

Similarly, when the master module no longer requires the clocks, it informs the LPCG/CCM block and the LPCG/CCM can then gate the clocks to the module and all the clock precedents that are not used by other blocks. The master module is then said to be in clock-gated or partially clock gated mode.

Examples of module supporting master clock protocol USDHC. Please see details in chapters describing these modules and in the CCM enable override register (CCM_CMEOR).

13.3.3.2.2 Slave clock protocol

This hardware protocol allows CCM to control the state of a slave module. CCM informs the slave module, through assertion of a stop/change request, when its clocks (both interface and functional) can be changed or gated. The slave acknowledges the request and CCM is then allowed to gate or change the clocks to the block.

Similarly, a clock-gated slave module may need to be woken up because of some event or a service request from a master module. In this situation, CCM enables the clocks to the module and then de-asserts the stop request to signal the module to wake up.

Examples of modules supporting slave clock protocol are CAN, and GPT. Please see details in chapters describing these modules and in the CCM Module Enable Overide Register (CCM_CMEOR). See CCM Memory Map/Register Definition for more details.

The protocol in both "master" and "slave" cases is completely hardware-controlled, but software should configure the clock management behavior for the module in two places: in the CCM registers associated with the block and in the block configuration registers.

13.3.3.3 Clock Domain(s)

A clock domain is a group of blocks fed by clock signals controlled by the same clock controls in CCM. By gating the clocks in a clock domain, the clocks to all the blocks belonging to that clock domain can be gated/activated, either by software control or by hardware control associated with block activity. Thus, a clock domain allows efficient control of the dynamic power consumption of the domain.

The device is partitioned into multiple clock domains and each clock domain is controlled by an associated group of clock gating cells within the LPCG block. This allows the CCM/LPCG to individually activate and gate each clock domain of the system.

13.3.3.4 Domain level clock management

The domain clock manager can automatically (based on hardware conditions) and manage the bus interface clocks within the clock domain. The functional clocks within the clock domain are managed through software settings.

13.3.3.5 Domain dependencies

A domain dependency is a hierarchical relationship between two clock domains. Clock domain "X" is said to depend on a clock domain "Y" when a block in clock domain "Y" provides services (or even just a clock) to a block in clock domain "X". As a result, clock domain "Y" must be active whenever clock domain "X" is active.

The dependency between two clock domains may also exist if one clock domain serves to ensure communication between two blocks (for example, the clock domain of the device interconnect).

13.4 Power management

13.4.1 Centralized Components of Power Management System

The power generation and management system is built around the PMU and GPC blocks.

A high level block diagram of the power management system in the SoC environment is shown in the figure below.

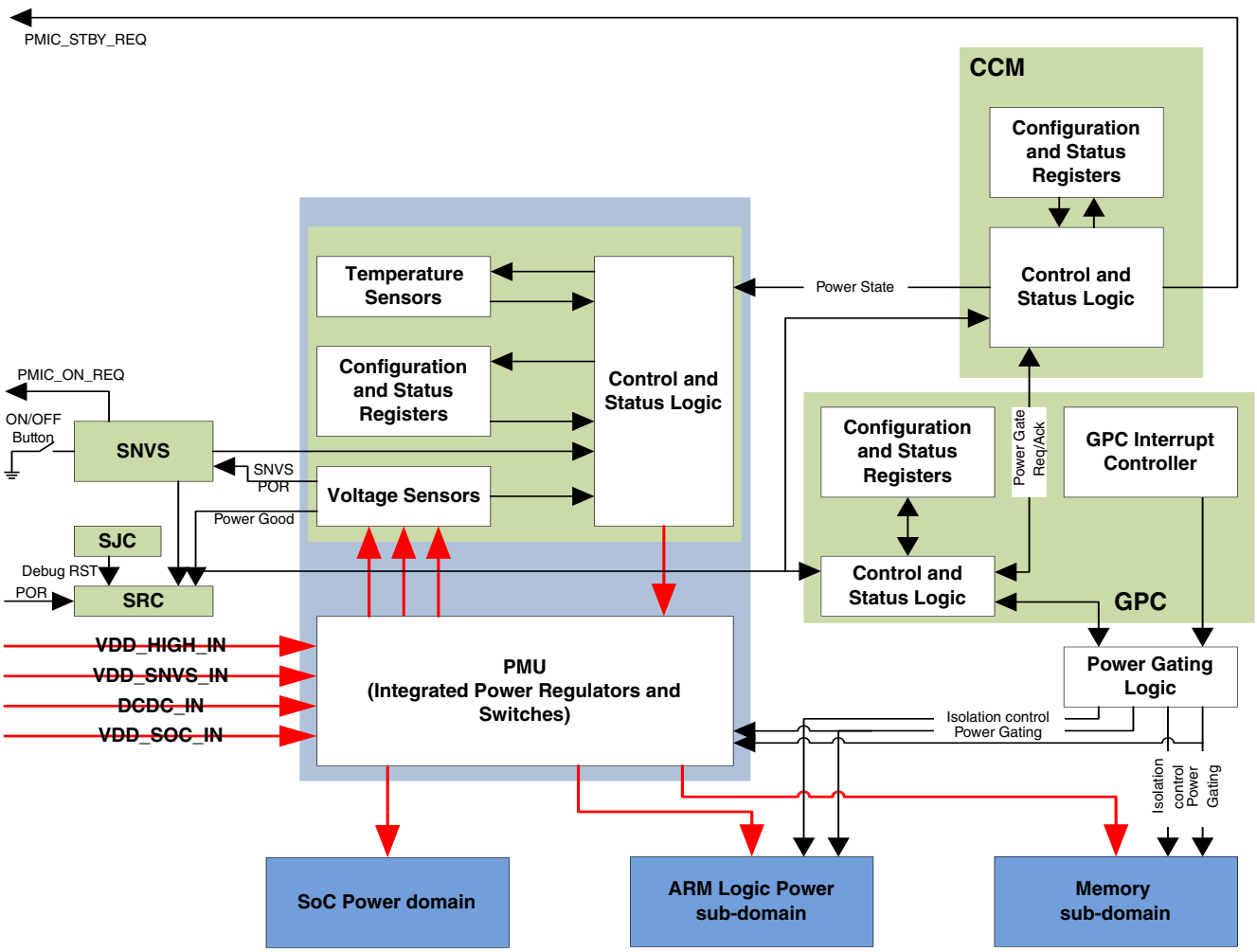


Figure 13-3. Power Management System

13.4.1.1 Integrated PMU

The first component of the power management system, referred to as the integrated PMU, is designed to simplify the external power interface.

It consists of a set of secondary power supplies that enable SoC operations from just two or three primary supplies. The high level block diagram of the power tree, utilizing the integrated PMU, is shown below.

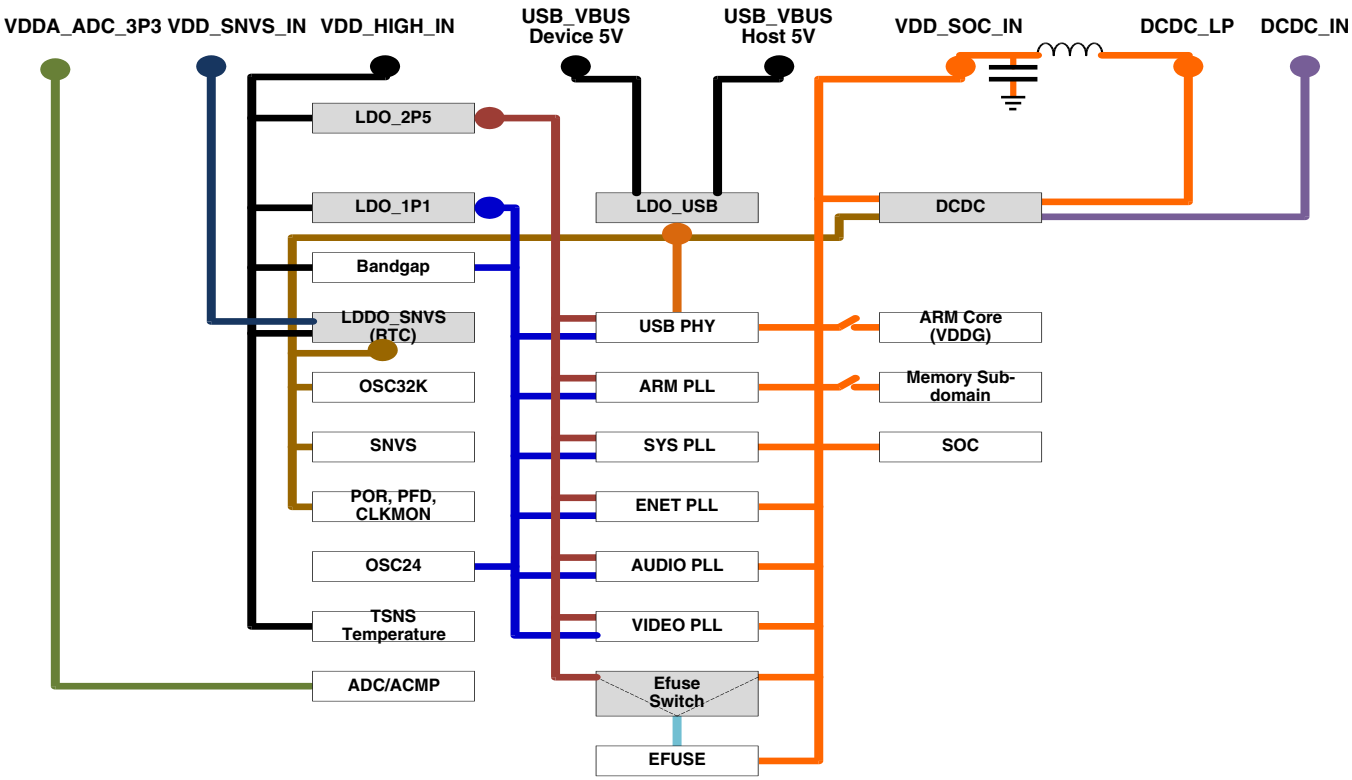


Figure 13-4. SOC Power Tree

The integrated PMU includes the following components:

- On-chip DCDC regulator
- Two Analog LDO regulators
- USB LDO
- SNVS regulator

See Power Management Unit (PMU) for further details on integrated PMU functional description and programmability.

13.4.1.1.1 DCDC Regulator

The DCDC regulator includes the following features:

- Adjustable (25 mV per step) high efficiency regulator
- Supports 3.3 V (±10%) input voltage
- Output ranged in 0.9 V~1.3 V
- Supports nominal run, low power standby and off modes
- Supports 0.9 V~1.3 V output in run mode

- Supports 0.9 V~1.0 V output in standby mode
- Over current and over voltage detection

These modes allow the regulator to implement voltage scaling and power gating, and allow bypass when an external high power efficient regulator is used as a direct source for some of the SoC loads.

The DVFS (Dynamic Voltage and Frequency Scaling) in a typical cost/complexity optimized application is considered by means of an internal DCDC. The DCDC workpoint can be dynamically adjusted according to the system's working frequency requirement.

13.4.1.1.2 Analog LDO regulators

There are two analog LDO regulators used for general system purposes:

- LDO_1P1 - The LDO_1P1 (VDD_HIGH_IN, NVCC_PLL) linearly regulates down a higher supply voltage (2.8V-3.3V) to produce a nominal 1.1V output voltage. This regulator supplies digital portions of USB PHYs, PLLs, and the internal 24 MHz oscillator.
- LDO_2P5 - The LDO_2P5 (VDD_HIGH_IN, VDD_HIGH_CAP) linearly regulates down a higher supply voltage (2.8V-3.3V) to produce a nominal 2.5V output voltage. The regular 2.5V LDO is combined with an alternate self-biased low-precision weak regulator which can be enabled for applications that need to keep the 2.5V output voltage alive during low power modes, where the main regulator and its associated global bandgap reference module are disabled to save power. The output of this weak-regulator is not programmable and is a function of its input power supply as well as its load current. Typically with a 3V input power supply, the weak-regulator output is 2.525V and its output impedance is approximately 40Ohm. Special procedure is recommended to move load back and forth between the main and low power (weak) regulators. This regulator supplies most of the analog circuitry of the integrated PHYs, and other analog and mix signal components integrated into the SoC.

13.4.1.1.3 USB LDO

The USB_LDO linearly regulates down the USB VBUS input voltages (typically 5V) to produce a nominal 3.0V output voltage. This regulator has a built in power-mux that allows the user to run the regulator from either one of the VBUS supplies when both are present. If only one of the VBUS voltages is present, the regulator automatically selects that supply. Current limit is also included to help the system keep the in-rush current within limits as required in USB 2.0 specification. This regulator supplies only low speed and full speed transceivers of USB PHYs.

13.4.1.1.4 SNVS regulator

The SNVS regulator takes the SNVS_IN supply and generates the SNVS_CAP supply, which in turn powers the real time clock and low power section of the SNVS modules. If VDDHIGH_IN is present, then the SNVS_IN supply is internally shorted to the VDDHIGH_IN supply to allow coin cell recharging if necessary.

13.4.1.2 GPC - General Power Controller

The GPC block includes the sub-blocks listed here.

- Power Gating Controller (PGC) - This sub-block of GPC has the following functions:
  - Provides the user with the ability to switch off power to a target subsystem.
  - Generates power-up and power-down control sequences. This includes interaction with CCM/LPCG and SRC, and control for clock and reset generation for power domains affected by power gating.
  - Provides programmable registers that adjust the timing of the power control signals.
  - Controls the CPU power domain and Memory sub-domain (PDRAM*x*).
- Wake-up interrupt controller - This controller initiates the system wake-up from low power modes when only low frequency real time clock remains active, and thus the Generic Interrupt Controller (GIC) can not handle synchronous interrupt signals. Additional features are as follows:
  - Supports up to 160 interrupts
  - Provides an option to mask/unmask each interrupt
  - Detects interrupts and generates the wake up signal

See General Power Controller (GPC) for further details on GPC, its sub-blocks, and information on its functional description and programmability.

13.4.1.3 SRC - System reset Controller

The reset controller is responsible for the generation of all reset signals and boot configuration decoding.

It determines the source and the type of reset, such as POR and COLD, and performs the necessary reset signal qualifications. SRC is capable of generating reset sequences in the following conditions:

- in interaction with external PMIC, based on external POR_B signal and "power ready" signals generated by the integrated PMU
- or in interaction with the integrated PMU only, based on its "power ready" signal.

Based on the type of reset, the reset logic generates the reset sequence for either the entire SoC or for the blocks that are power-gated.

See System Reset Controller (SRC) for further details on SRC functional description and programmability.

13.4.1.4 Power domain(s)

A power domain is a group of blocks or sub-blocks fed by power sources controlled by the same power controls in GPC.

Some power domains can be split into a logic sub-domain and a memory sub-domain. The memory sub-domain in such case may contain two entities:

- Memory array(s) - Powered by a dedicated voltage rail enabling memory retention while core is OFF.
- Memory interface logic - Powered by the same voltage source as the logic sub-domain of the power domain.

Signals crossing power domain boundaries or sub-domain boundaries are passed through proper isolation and/or level-shifting cells to ensure robust operations of the SoC when some of domains are power gated or working at a reduced voltage.

The following figure shows the power domain interface within the system.

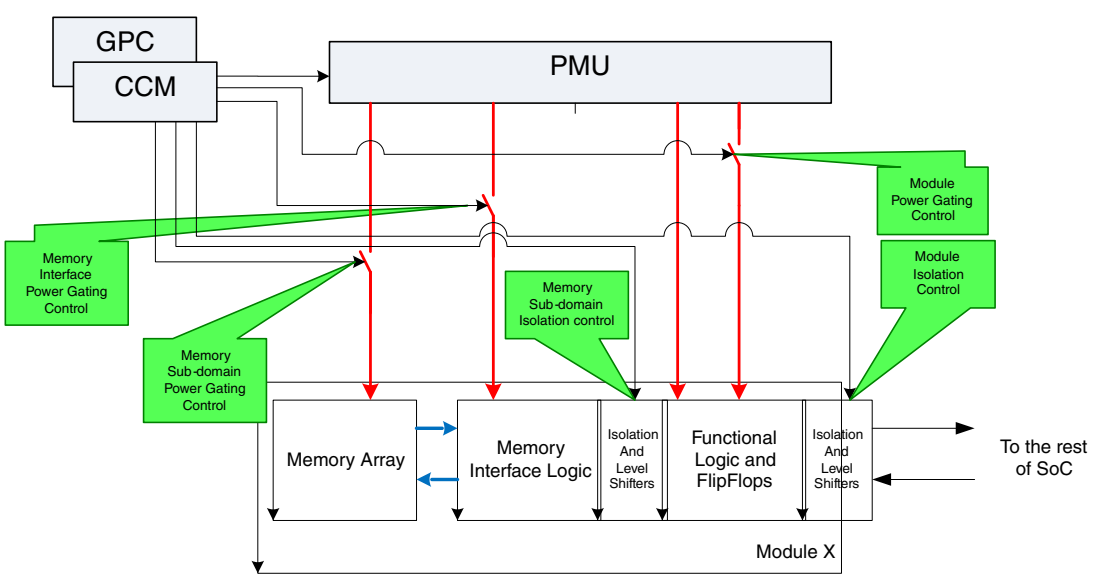


Figure 13-5. Power domain interface

13.4.1.4.1 Power distribution

The power distribution tree is comprised of multiple power domains. The main power domains are:

- Arm - The Arm domain contains the Arm Core platform . This domain can be supplied from internal or external controllable regulator.
- FlexRAM memory array - Bank0 to Bank15 of FlexRAM.
- OCRAM2 memory array - The OCRAM2 memory array is partitioned into 2 sub-domains: 64KB always on domain (also known as PDRET domain, starting from address 2020_0000h), 448KB PDRAM1 domain.
- SNVS/RTC low power domain - The SRTC (Secure Real Time Counter) domain contains only counter, comparator and compared data of the on-chip RTC. This domain should be supplied from an external single cell LiION battery and/or an external pre-regulated power supply. This SNVS domain also contains Low Power State Retention (LPSR) GPIO, IOMUX, and LPSR general purpose register.
- Analog domain - The analog domain contains the PLLs, LDOs and USB PHY. The domain supplies should be constant to allow continuous clock during any dynamic voltage scaling techniques. The digital supply should be provided from an internal regulator, and can be combined with the memory array supply. The analog supply should be provided from internal low noise regulator.
- Main SoC logic - The main SoC logic domain contains the rest of the logic of the SoC. It is supplied by the same power supply as the Arm Core Platform.

From a DVFS and Power Gating standpoint, the following digital logic domains are affected:

- Arm Core Platform - DVFS and power gating.
- FlexRAM memories - DVFS and power gating.
- PDRAM1 (within OCRAM2) memories - DVFS and power gating.
- Main SoC logic - DVFS, no power gating.

13.4.1.4.2 Domain memory and domain logic state retention in case of power gating

The following is the list of relevant memories and logic domains with the description of their state-retention support.

NOTE

For more detailed infomation, see the table "Low Power Mode Definition" in this chapter.

- Arm Core Platform logic - the software state retention for all logic is implemented in this domain and L1 cache memories. Software state retention means that the content

of relevant registers should be stored in some memory retaining its state (OCRAM for example) while the logic domain is power-gated.
- PDRET memories - hardware state-retention in all power modes except SNVS mode.
- FlexRAM PDRAM0 memories - hardware state-retention . When Arm core is power gated, PDRAM0 memories can be either left on with its content retained or power gated together with the core.
- FlexRAM PDRAM1 memories - can be powered gated (together) independently of Arm core power gating.
- SoC - hardware state-retention .
- SNVS_LP - hardware state-retention even when SoC supplies are removed.

13.4.1.4.3 Power Gating Domain Management

The following bullets provide the sequence required for power-gating the relevant power-domains:

13.4.1.4.3.1 Arm Core Platform

1. Copy through software all the Core configuration registers to a powered-on memory
2. Configure the GPC/PGC CPU registers in PGC Memory Map/Register Definition as follows to power-down the core on the next "WFI" instruction:
   - Configure the GPC/PGC PGC_CPU_PDNSCR Register ISO and ISO2SW bits. The ISO field determines the delay between the power-down request and enabling the platform isolation. The ISO2SW field determines the delay between the platform isolation and the actual power-off switch to the supplies.
   - Configure the GPC/PGC PGC_CPU_PUPSCR Register SW and SW2ISO bits. The SW field determines the delay between the power-up request and the actual power-up of the supplies. The SW2ISO field determines the delay between asserting power toggle and negating platform isolation.
   - Configure the GPC PGC PGC_CPU_CTRL PCR bit to allow the power down of the platform
   - Arm Core Platform should execute a "WFI" instruction.

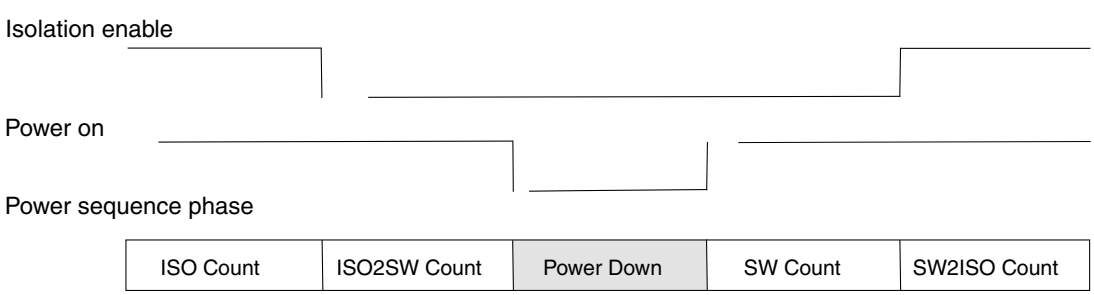


Figure 13-6. Arm Core Platform isolation and power on switch flow

13.4.1.4.3.2 SoC

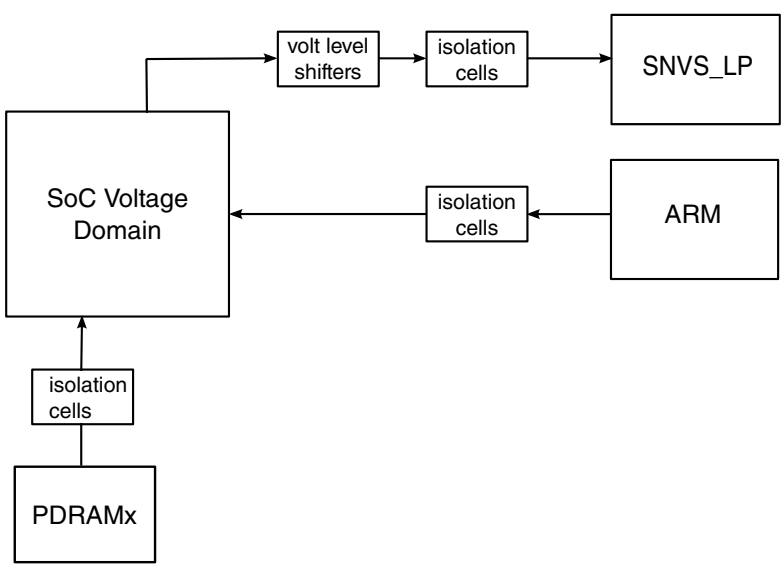
For additional power reduction it is possible to do the following:

- Power-down the internal oscillator by configuring the following bits CCM_CCR[COSC_EN] (CCM Memory Map/Register Definition). This can be done only in case there is no dependency on 24 MHz XTAL for wake-up.
- It is possible to turn off and turn on the PMIC supplies to the SoC even when the SoC supplies are off. Since SNVS_LP is powered through an "always on" supply, configuring the SNVS_LP DP_EN to "1" allows changing the PMIC_ON_REQ pad (SoC on/off supply indication to the PMIC) through the ONOFF pad.

13.4.1.4.4 Power Gating domain dependencies

There are 3 power domains that need to be isolated in different power-down cases:

- Arm Core Platform and PDRAM*x* - Isolation needs to be enabled before power-down. This is taken care of automatically after CCM and PGC are configured and the Arm Core Platform executes the "WFI" instruction.
- SNVS_LP - Different from the 2 domains above, the SNVS_LP isolation isolates the signals coming from the SoC to the SNVS_LP. This is required for saving the contents of the SNVS_LP (such as the real-time clock). The isolation is activated in 2 ways:
  - Automatically through the power-fail detector in the PMU
  - Through software configuration



Note: The arrows refer to the signal directions for the voltage level shifters and isolation cells

Figure 13-7. Isolation cells and Voltage level shifters placing

13.4.1.5 Voltage domains

The list found here states the different voltage domains and their scalability in regarding to power-saving in dynamic and static scenarios.

- SoC domain: including Core Platform, FlexRAM memories and SoC logic - Scalable voltage in both dynamic and static scenarios
- ANALOG components including PLLs - Fixed voltage
- I/O - Fixed voltage
- SNVS_LP - Fixed voltage

13.4.1.6 Voltage domain management

13.4.1.6.1 Dynamic

13.4.1.6.1.1 Voltage Scaling

A simplistic way to reduce power consumption in dynamic scenarios is to scale down the Arm and SoC voltage according to the allowed voltage points and corresponding frequencies specified in datasheet.

13.4.1.6.2 Static

13.4.1.6.2.1 Standby Leakage reduction (SLR)

Standby leakage reduction is a power-management technique utilizing:

- Reduced supply voltage for relevant domains

With SLR, the device switches into low-power active system modes automatically or in response to user requests during system Stop, Wait, or DSM modes (that is, in situations when no application is started and no system activity is presented).

When applying SLR, the system remains in the lowest static power mode while retaining logic and memory states. This technique trades static power consumption for wake-up latency while maintaining fast system response time suitable for most applications.

See CCM Control Register (CCM_CCR), CCM Low Power Control Register (CCM_CLPCR) and PMU Miscellaneous Register 0 (PMU_MISC0) for further details on SLR programmability options.

The following describes the flow for applying standby voltage:

- Configure the external PMIC standby voltage, refer to chip datasheet.
- Configure CCM_CCR[RBC_EN] bits to bypass and disable PMU regulators in the next Arm "WFI" execution.
- Configure CCM_CCR[REG_BYP_COUNT] bits to allow proper voltage restoration by the external PMIC when exiting standby.
- Arm Core Platform executes the "WFI" instruction that completes the software sequence putting the SoC into low power mode

13.4.1.7 System domains layout

The following table describes the different power modes.

For more information regarding the low-power application design points on i.MX RT series, see the application note AN12085: How to use i.MX RT Low Power Feature.

NOTE

The IDLE, SUSPEND and SNVS are usually referred as low power mode.

- IDLE Modes: CPU in WFI state, some portion of the chip can be clock gated or power gated. The chip can wakeup from this mode from IRQ with a very short latency. IDLE mode is entered automatically from RUN mode when CPU executes WFI instruction, so the state of IDLE mode will have dependency on the state of the RUN mode. To avoid showing a lot of IDLE modes, the following two IDLE states are defined:
  - System IDLE: the IDLE mode entered from Low Speed Run mode. When entering from other RUN modes, the difference is mainly the CPU/Bus frequency.
  - Low Power IDLE: the IDLE mode entered from Low Power RUN mode.
- SUSPEND Mode: CPU power gated, all clocks gated only except 32KHz, analog modules also enter low power state. It provides the lowest power while keeps the system alive, but it would have longer exit time.
- SNVS Mode: All the modules are turned off, only except RTC is active.

Table 13-1. Low Power Mode Definition

	Overdrive Run	Full Speed Run	Low Power Run	System Idle	Low Power Idle	Suspend	SNVS
VDD_SOC_IN voltage	1.25V min	1.15V min	0.925V min	1.15V min	0.925V min	0.925V min	OFF
CCM LPM Mode	Run	Run	Run	WAIT	WAIT	STOP	N/A
Arm Core	up to 600 MHz	up to 528 MHz	up to 24 MHz	WFI	WFI	Powered down	OFF
AHB Clock	up to 600 MHz	up to 528 MHz	up to 24 MHz	up to 528 MHz	up to 24 MHz	OFF	OFF
IPG clock	up to 150 MHz	up to 132 MHz	up to 24 MHz	up to 132 MHz	up to 24 MHz	OFF	OFF
PER clock	up to 75 MHz	up to 72 MHz	up to 24 MHz	up to 72 MHz	up to 24 MHz	OFF	OFF
L1 Cache	ON	ON	ON	ON	ON	Powered down	OFF
PDRET	ON	ON	ON	ON	ON	ON	OFF
PDRAM0	ON	ON	ON	ON	ON	ON/OFF	OFF
PDRAM1	ON	ON	ON	ON/OFF	ON/OFF	OFF	OFF
System PLL	ON	ON	Powered down	ON	Powered down	Powered down	OFF
Other PLL	ON	ON	Powered down	Powered down	Powered down	Powered down	OFF
XTAL	ON	ON	OFF	ON	OFF	OFF	OFF
RC OSC	OFF	OFF	ON	OFF	ON	OFF	OFF
LDO2P5	ON	ON	OFF	ON	OFF	OFF	OFF
LDO1P1	ON	ON	OFF	ON	OFF	OFF	OFF
WEAK2P5	OFF	OFF	ON	OFF	ON	OFF	OFF
WEAK1P1	OFF	OFF	ON	OFF	ON	OFF	OFF
Bandgap	ON	ON	OFF	ON	OFF	OFF	OFF
Low Power Bandgap	ON	ON	ON	ON	ON	ON	OFF
Module clocks	ON as configured in CCM	ON as configured in CCM	ON as configured in CCM	ON as configured in CCM	ON as configured in CCM	OFF	OFF
GPIO wakeup	N/A	N/A	N/A	Yes	Yes	Yes	Yes (1 pin only)
RTC wakeup	N/A	N/A	N/A	Yes	Yes	Yes	Yes
USB remote wakeup	N/A	N/A	N/A	Yes	Yes	Yes	No
Other wakeup source	N/A	N/A	N/A	Yes	Yes	No	No

There is a single hardware signal (stop_mode) coming into PMU which sets the PMU in either of two "STOP" states. The STOP state is controlled by the PMU_MISC0[STOP_MODE_CONFIG] bit (See PMU Memory Map/Register

Definition). It is recommended that the blocks be configured for safe powerdown/up through the registers before asserting the stop_mode signal. Blocks not described in the section below are unaffected by stop_mode.

If the stop_mode_config is set to zero, thus in the STOP mode all blocks powered down in minimum power configuration.

If the stop_mode_config is set to one, thus in the STOP mode some of the blocks remain powered and in different states as defined in the table below.

Table 13-2. STOP mode configuration

Block	STOP_MODE_CONFIG=0	STOP_MODE_CONFIG=1
reg1p1	off	on
reg2p5	off	on
reg3p0	off/on depending on vbus. Uses crude local reference if vbus is present	off/on depending on vbus . Uses analog central bandgap if VBUS is present.
bandgap	off	functional
temp_sensor	off	off
well_bias	hardware controlled	hardware controlled
All PLLs	off	off
OSC24M	off	Controlled by CCM configuration

13.4.2 Power management techniques

The device supports the power-management techniques with the features found here.

- Partitioning of the device into voltage, power, clock, and reset domains
- Domain isolation that allows flexible configurations of domains on/off states to form use cases targeting various applications
- Clock tree with selective clock-gating conditions and almost independent clock roots
- Power, reset, and clock control hardware mechanism to manage sleep and wake-up dependencies of power domains
- Software-controllable and hardware-controllable clock gating for functional modules and buses
- Memory retention and state retention capability (Software State Retention for Arm core) for preserving memory contents and device state in low-power modes
- Support for low-power device modes input/output (I/O) pad configuration for minimum power
- Variety of operating modes to optimize device performance and wake-up times
- Thermal monitoring and thermal aware performance management

Many of the low power features are fully or partially software controllable and can be configured for the specific requirements of a target system.

Combining these techniques, the system designer may meet tight requirements of low-power standby and operational modes while maintaining high performance for time-critical tasks.

13.4.2.1 Power saving techniques

The table below lists power saving techniques supported by the SoC in their connection to different components of power consumption.

Table 13-3. Power saving design/architecture and power saving techniques

Techniques	Active SoC Power	Standby SoC Power	System Power
Temperature Monitoring, and active frequency throttling	√		
Arm Core Platform SRPG (Software)		√	
Arm Core Platform Power Gating		√	
Clock gating (automatic dynamic and forced)	√		
Integrated PMU (IR drop, efficiency, accuracy)	√		√
C4 package (IR drop, thermal)	√		
Display Backlight optimization (SW)			√
Architecture: FlexRAM memories	√		
Architecture: USB integration			√
Low Power DRAM: SDRAM			√

13.4.2.2 Thermal-aware power management

The temperature sensor block (TEMPMON) implements a temperature sensor/conversion function. The block features an alarm function that can raise an interrupt signal if the temperature is above a specified threshold.

Software may implement temperature aware DVFS for the Arm domain as well as temperature aware frequency scaling for other system components to ensure that both the frequency and voltage is lowered when the die temperature is above the specified limit.

Software may also implement temperature aware task scheduling to ensure that non-critical tasks are suspended when the die temperature is above the specified limit.

See Temperature Monitor (TEMPMON) for further details on temperature monitor functions and programmability options.

13.4.2.3 Peripheral Power management

13.4.2.3.1 IO power reduction

Software configures IO to low power modes:

- PHYs - make sure that all unused PHYs are placed to lowest power state. Please refer relevant chapter for further information about different PHYs
- Digital IOs - Make sure all unnecessary PU/PD are disabled and IO are switched to either minimal drive strength or to input mode (when applicable)

13.4.2.4 Examples of External Power Supply Interface

This section presents the example of external power supply interfacing to the chip.

The scenario based on integrated PMU system is presented in the following figure. This scenario minimizes BoM and board design complexity.

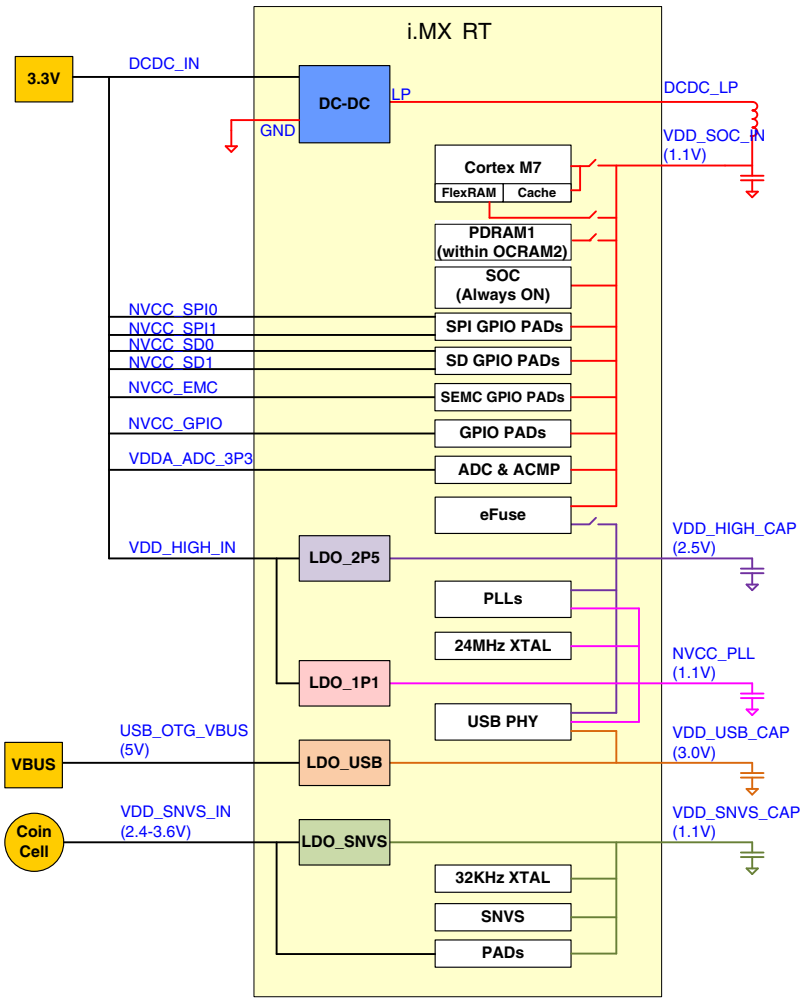


Figure 13-8. SOC Power Architecture

13.5 ONOFF (Button)

The chip supports the use of a button input signal to request main SoC power state changes (i.e. On or Off) from the PMU.

The ONOFF logic inside of SNVS_LP allows for connecting directly to a PMIC or other voltage regulator device. The logic takes a button input signal and then outputs a pmic_en_b and set_pwr_off_irq signal. PMIC logic also supports the SNVS_LP tamper logic which will allow waking the system up when a tamper event has happened while in the OFF state. The logic has two different modes of operation (Dumb and Smart mode).

The Dumb PMIC mode uses a 2 state state machine, as shown below. The output of the pmic_en_b is generated by the state of the state machine.

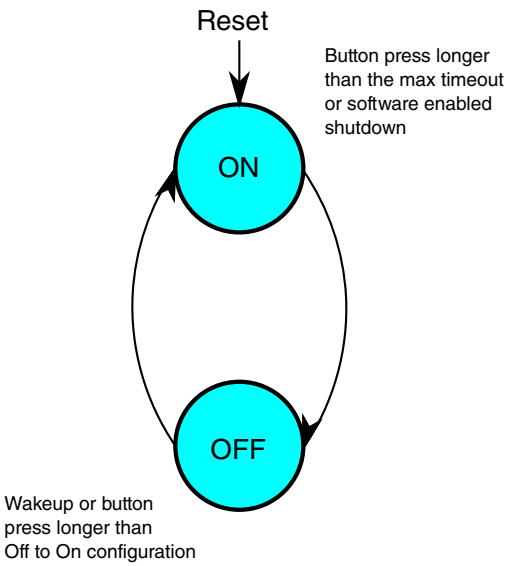


Figure 13-9. Dumb PMIC Mode State Machine

The Dumb PMIC Mode uses pmic_en_b to issue a level signal for on and off. Dumb pmic mode has many different configuration options which include (debounce, off to on time, and max time out).

- Debounce: The debounce configuration supports 0 msec, 50 msec, 100 msec and 500 msec. The debounce is used to generate the set_pwr_off_irq interrupt. While in the ON state and the button is pressed longer than the debounce time the set_pwr_off_irq is generated.
- Off to On Time: The Off to On configuration supports 0 msec, 50 msec, 100 msec, and 500 msec. This configuration supports the time it takes to request power on after the configured button press time has been reached. After the button is pressed longer than the configuration time, the state machine will transition from the OFF to the ON state.
- Max Timeout: The max timeout configuration supports 5 secs, 10 secs, 15 secs and disable. This configuration supports the time it takes to request power down after the button has been pressed for the defined time.

The Smart PMIC mode is meant to connect to another PMIC. The pmic_en_b signal issues a pulse instead of a level signal. The only configuration option available for this mode is the Debounce configuration that is used for the set_pwr_off_irq.

13.6 WAKEUP Pin

This chip supports the use of a WAKEUP pin (GPIO5_IO00 ALT5) to request main SoC power on to exit SNVS mode. To use it, follow the tips below.

- SNVS must be in Dumb PMIC mode (*default*, and must for on-chip DCDC)
- Configure IOMUXC_SNVS to select WAKEUP pin ALT5 to mux it to GPIO5_IO00
- Configure GPIO5 ICR to either low level or high level sensitive
- Set GPIO5 IMR bit0 to enable GPIO5_IO00 interrupt
- Enter SNVS mode by SW or ON/OFF button, then on-chip DCDC will be off, and PMIC_ON_REQ pin will also go to low to notify outside
- Assert WAKEUP pin for at least two 32 KHz cycles (active high or low depends on GPIO5 ICR configuration), so that GPIO5 interrupt gets sampled by SNVS module
- SNVS will assert PMIC_ON_REQ pin high and on-chip DCDC begins to ramp on
- SoC power on procedure (except SNVS) begins (ROM boot, …)

Chapter 14
Clock Controller Module (CCM)

14.1 Chip-specific CCM information

Table 14-1. Reference links to related information

Topic	Related module(s)	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
Interrupts, DMA Events and XBAR Assignments	-	Interrupts, DMA Events and XBAR Assignments

NOTE

The OCRAM clock cannot be turned off when CM7 Cache is running, on this device. See CCM_CCGR3[CG14] bitfield.

System STOP/WAIT status can be observed on pin. See CCM_STOP/CCM_WAIT in the "Muxing Options" table of the External Signals and Pin Multiplexing chapter, for pin muxing details.

14.2 Overview

The Clock Control Module (CCM) generates and controls clocks to the various modules in the design and manages low power modes. This module uses the available clock sources to generate the clock roots.

The Clock Controller Module controls the following functions:

- Uses the available clock sources to generate clock roots to various parts of the chip:
  - PLL1 also referenced as ARM PLL
  - PLL2 also referenced as System PLL
  - PLL3 also referenced as USB1 PLL
  - PLL4 also referenced as Audio PLL
  - PLL5 also referenced as Video PLL
  - PLL6 also referenced as ENET PLL
  - PLL7 also referenced as USB2 PLL (This PLL is only used by the USB UTM interface through a direct connection.)
- Uses programmable bits to control frequencies of the clock roots.
- Controls the low power mechanism.
- Provides control signals to LPCG for gating clocks.
- Provides handshake with SRC for reset performance.
- Provides handshake with GPC for support of power gating operations.

14.2.1 Features

The CCM includes these distinctive features:

- Provides root clock to SoC modules based on several source clocks.
- Arm core root clock is generated from a dedicated source clock.
- Includes separate dividers to control generation of core and bus root clocks (AXI, AHB, IPG).
- Includes separate dividers and clock source selectors for each serial root clock.
- Optional external clocks to bypass PLL clocks.
- Selects clock signals to output on CCM_CLKO onto the pads for observability.
- Controllable registers are accessible via IP bus.
- Manages the Low Power Modes, namely RUN, WAIT and STOP. The gating of the peripheral clocks is programmable in RUN and WAIT modes.
- Manages frequency scaling procedure for Arm core clock by shifting between PLL sources, without loss of clocks.
- Manages frequency scaling procedure for peripheral root clock by programmable divider. The division is done on the fly without loss of clocks.
- Interface for the following IPs:
  - PLL - Interfaces for each PLL
  - LPCG - Low Power Clock Gating unit
  - SRC - System Reset Controller
  - GPC - General Power Controller

14.2.2 CCM Block Diagram

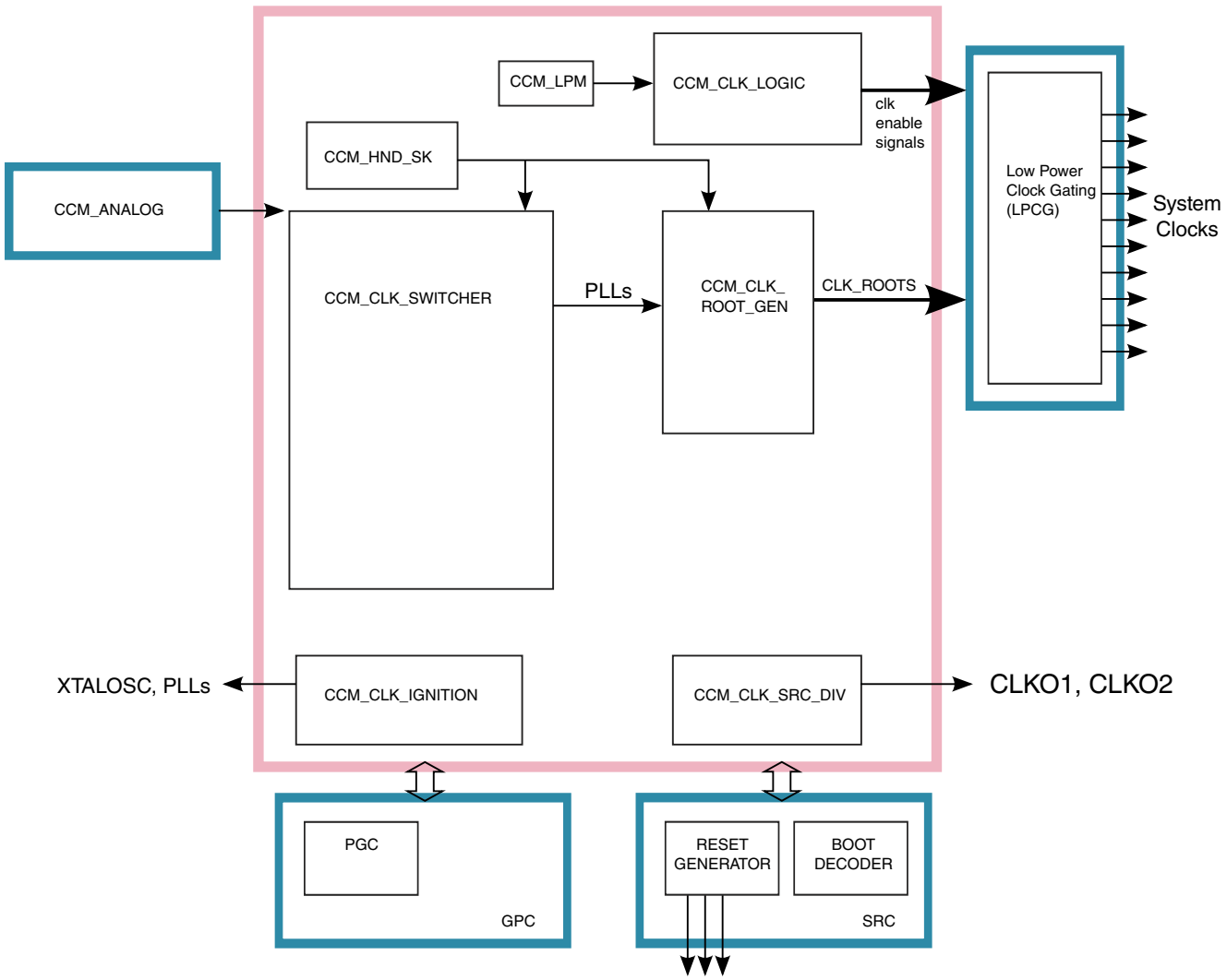


Figure 14-1. Block Diagram

CCM contains the following sub-blocks:

Table 14-2. CCM Sub-blocks

CCM_CLK_SRC_DIV Sub-block	Description
CCM_CLK_IGNITION	Manages the ignition process. This module starts its functionality after CCM comes out of reset. It manages the process that begins with starting the OSC, PLLs and finishes with creation of stable output root clocks after reset.
CCM_CLK_SWITCHER	Receives the clock outputs of the PLLs, together with the bypass clocks for the PLLs, and generates switcher clock outputs (pll3_sw_clk) for the CCM_CLK_ROOT_GEN sub-module.

Table continues on the next page...

Table 14-2. CCM Sub-blocks (continued)

CCM_CLK_SRC_DIV Sub-block	Description
CCM_CLK_ROOT_GEN	Receives the main clocks (PLLs / PFDs) and generates the output root clocks.
CCM_CLK_LOGIC	Generates the clock enables. It generates the clock enable signals based on info from CCM_LPM and CCM_IP. The clock enables are used in LPCG to turn off and on the split clocks.
CCM_LPM	Manages the low power modes of the IC.
CCM_HND_SK	Manages the handshake when changing certain root clock dividers that require handshake.

14.3 External Signals

The following table describes the external signals of CCM:

Table 14-3. CCM External Signals

Signal	Description	Pad	Mode	Direction
CCM_REF_EN_B	Enable external reference clock (CKIH)	GPIO_SD_B1_07	ALT6	O
CCM_CLKO1	Observability clock 1 output	GPIO_SD_B0_04	ALT6	O
CCM_CLKO2	Observability clock 2 output	GPIO_SD_B0_05	ALT6	O
CCM_PMIC_STBY_REQ	Signal coming from PMIC to indicate that the voltage started to change in PMIC_STBY_REQ. CCM will ignore this indication if CLPCR [BYPASS_PMIC_READY] is set.	PMIC_STBY_REQ	ALT0	I
CCM_PMIC_READY	Signal goes to STANBY_REQ pin, which notifies external power management IC to move from functional voltage to standby voltage.	GPIO_AD_B0_12 pin 24	ALT1	O
		GPIO_EMC_32 pin 28	ALT3	O
		GPIO_AD_B1_08 pin 22	ALT3	O
		GPIO_AD_B1_01 pin 18	ALT4	O
		GPIO_SD_B1_03	ALT6	O

14.4 CCM Clock Tree

The following figures (Part 1 and Part 2) show the clock tree configuration and clock roots for CCM.

For detailed sub-block information, see:

- Clock Switcher
- Clock Root Generator
- Low Power Clock Gating module (LPCG)
- System Clocks

NOTE

The default frequency values (in MHz) for the PLLs and PFDs are shown in the Clock Tree diagram that follows. The PLLs and PFDs control registers may be reprogrammed according to the speed grade of the SoC being used, but should not exceed that maximum setting for that speed grade.

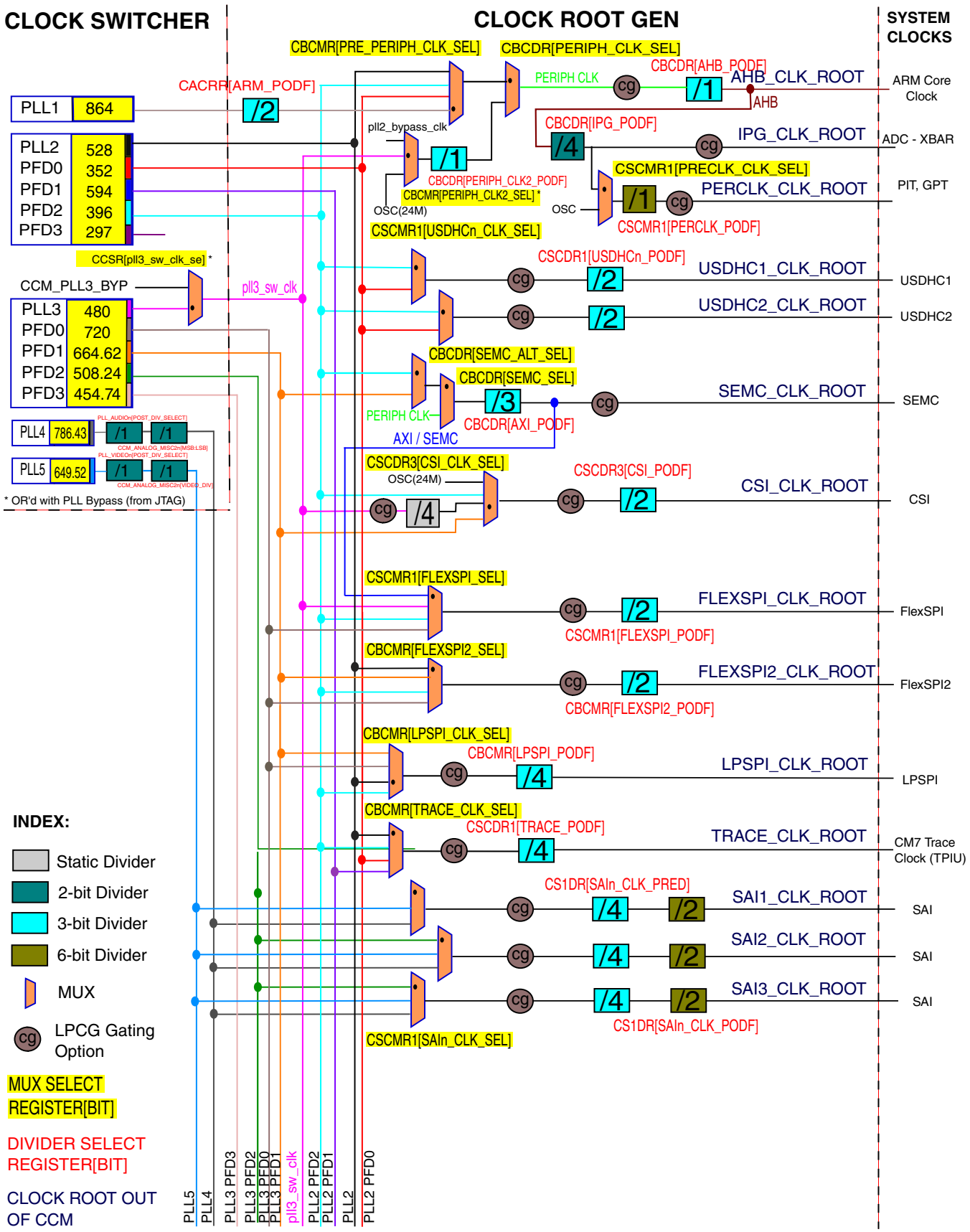


Figure 14-2. Clock Tree - Part 1

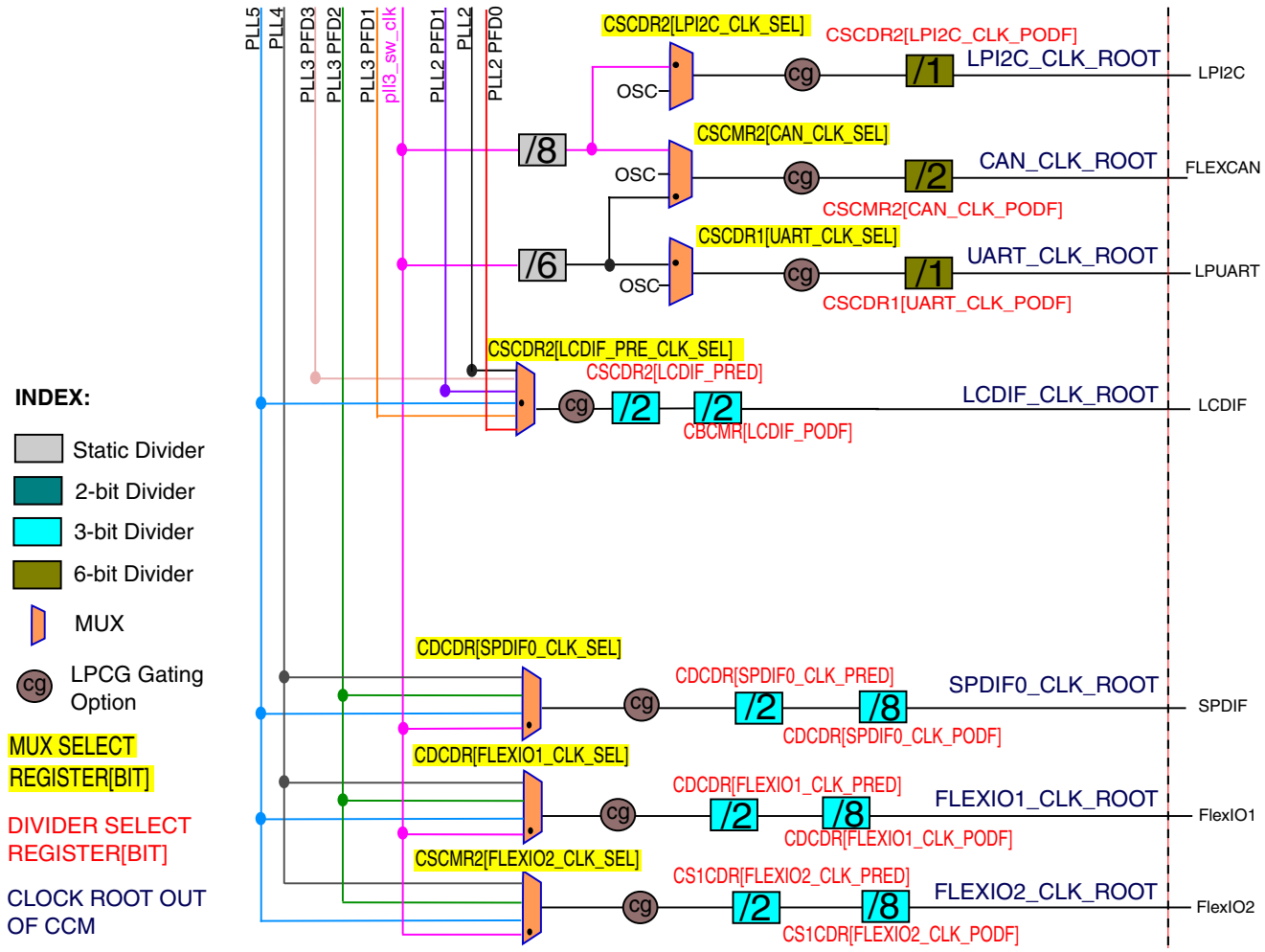


Figure 14-3. Clock Tree - Part 2

NOTE

In this device, the clock root for FlexIO3 is also flexio2_clk_root, and the bitfield of clock selector/divider is shared for FlexIO2/FlexIO3.

14.5 System Clocks

The table found here shows the CCM output clocks' system-level connectivity.

The gating option in the table can either be CGR bit or clock enable from the block itself. Applicable override bits are also shown.

Table 14-4. System Clocks, Gating, and Override

Module	Module Clock	Clock Root	Module Clock Gating Enable	Module Override Enable
ACMP*n*	acmp1 clocks	ipg_clk_root	CCGR3[CG10] (acmp1_clk_enable)	
	acmp2 clocks	ipg_clk_root	CCGR3[CG11] (acmp2_clk_enable)	
	acmp3 clocks	ipg_clk_root	CCGR3[CG12] (acmp3_clk_enable)	
	acmp4 clocks	ipg_clk_root	CCGR3[CG13] (acmp4_clk_enable)	
ADC*n*	adc1_ipg_clk	ipg_clk_root	CCGR1[CG8] (adc1_clk_enable)	
	adc2_ipg_clk	ipg_clk_root	CCGR1[CG4] (adc2_clk_enable)	
ADC_ETC	adc_etc_ipg_clk	ipg_clk_root		
AIPS	aips_tz1_hclk	ahb_clk_root	CCGR0[CG0] (aips_tz1_clk_enable)	
	aips_tz2_hclk	ahb_clk_root	CCGR0[CG1] (aips_tz2_clk_enable)	
	aips_tz3_hclk	ahb_clk_root	CCGR6[CG9] (aips_tz3_clk_enable)	
	aips_tz4_hclk	ahb_clk_root	CCGR5[CG6] (aips_tz4_clk_enable)	
AOI*n*	aoi1_ipg_clk	ipg_clk_root	CCGR3[CG4] (aoi1_clk_enable)	
	aoi2_ipg_clk	ipg_clk_root	CCGR1[CG7] (aoi2_clk_enable)	
Arm CM7	cm7_mxrt_ipg_clk	ipg_clk_root		
	cm7_mxrt_ipg_clk_s	ipg_clk_root		
	cm7_mxrt_trace_clk_in	trace_clk_root	CCGR0[CG11] (trace_clk_enable)	
	cm7_mxrt_main_clk	ahb_clk_root		
	cm7_mxrt_axi_clk	ipg_clk_root		
BEE	bee_clk	ipg_clk_root	CCGR4[CG3] (bee_clk_enable)	
CAN*n*	can1_ipg_clk	ipg_clk_root	CCGR0[CG7] (can1_clk_enable)	
	can1_ipg_clk_chi	ipg_clk_root	CCGR0[CG7] (can1_clk_enable)	
	can1_ipg_clk_pe	can_clk_root	CCGR0[CG8] (can1_serial_clk_enable)	CMEOR[MOD_EN_OV_CAN1_CPI]
	can1_ipg_clk_pe_nogate	can_clk_root		

Table continues on the next page...

Table 14-4. System Clocks, Gating, and Override (continued)

Module	Module Clock	Clock Root	Module Clock Gating Enable	Module Override Enable
	can1_ipg_clk_s	ipg_clk_root	CCGR0[CG7] (can1_clk_enable)	
	can1_mem_ram_CLK	ipg_clk_root	CCGR0[CG7] (can1_clk_enable)	
	can2_ipg_clk	ipg_clk_root	CCGR0[CG9] (can2_clk_enable)	
	can2_ipg_clk_chi	ipg_clk_root	CCGR0[CG9] (can2_clk_enable)	
	can2_ipg_clk_pe	can_clk_root	CCGR0[CG10] (can2_serial_clk_enable)	CMEOR[MOD_EN_OV_CAN2_CPI]
	can2_ipg_clk_pe_nogate	can_clk_root		
	can2_ipg_clk_s	ipg_clk_root	CCGR0[CG9] (can2_clk_enable)	
	can2_mem_ram_CLK	ipg_clk_root	CCGR0[CG9] (can2_clk_enable)	
	canfd_ipg_clk	ipg_clk_root	CCGR7[CG3] (canfd_clk_enable)	
	canfd_ipg_clk_chi	ipg_clk_root	CCGR7[CG3] (canfd_clk_enable)	
	canfd_ipg_clk_pe	can_clk_root	CCGR7[CG4] (canfd_serial_clk_enable)	CMEOR[MOD_EN_OV_CANFD_CPI]
	canfd_ipg_clk_pe_nogate	can_clk_root		
	canfd_ipg_clk_s	ipg_clk_root	CCGR7[CG3] (canfd_clk_enable)	
	canfd_mem_ram_CLK	ipg_clk_root	CCGR7[CG3] (canfd_clk_enable)	
CCM	ccm_ccm_ipg_clk_s	ipg_clk_root		
	ccm_ipg_clk	ipg_clk_root		
CSI	csi_csi_hclk	ipg_clk_root	CCGR2[CG1] (csi_clk_enable)	
	csi_ipg_clk	ipg_clk_root	CCGR2[CG1] (csi_clk_enable)	
	csi_ipg_clk_s	ipg_clk_root	CCGR2[CG1] (csi_clk_enable)	
	csi_ipg_clk_s_raw	ipg_clk_root	CCGR2[CG1] (csi_clk_enable)	
	csi_mem_rxfifo_clk	ipg_clk_root	CCGR2[CG1] (csi_clk_enable)	
CSU	csu_ap_ckil_clk	ckil_sync_clk_root		
	csu_ipg_clk_s	ipg_clk_root	CCGR1[CG14] (csu_clk_enable)	
DCDC	dcdc clock	ipg_clk_root	CCGR6[CG3] (dcdc_clk_enable)	

Table continues on the next page...

Table 14-4. System Clocks, Gating, and Override (continued)

Module	Module Clock	Clock Root	Module Clock Gating Enable	Module Override Enable
DCP	dcp_clk	ipg_clk_root	CCGR0[CG5] (dcp_clk_enable)	
	dcp_mem_clk	ipg_clk_root	CCGR0[CG5] (dcp_clk_enable)	
DMA_MUX	dma_ch_mux_ipg_clk	ipg_clk_root	CCGR5[CG3] (dma_clk_enable)	
	dma_ch_mux_ipg_clk_s	ipg_clk_root	CCGR5[CG3] (dma_clk_enable)	
eDMA	edma_ipg_clk	ipg_clk_root	CCGR5[CG3] (dma_clk_enable)	
	edma_hclk	ipg_clk_root	CCGR5[CG3] (dma_clk_enable)	
ENET	enet_ipg_clk	ipg_clk_root	CCGR1[CG5] (enet_clk_enable)	
	enet_ipg_clk_mac0	ipg_clk_root	CCGR1[CG5] (enet_clk_enable)	
	enet_ipg_clk_mac0_s	ipg_clk_root	CCGR1[CG5] (enet_clk_enable)	
	enet_ipg_clk_s	ipg_clk_root	CCGR1[CG5] (enet_clk_enable)	
	enet_ipg_clk_time	ref_enetpll2_clk	CCGR1[CG5] (enet_clk_enable)	
	enet_mem_mac0_mem_clk	ipg_clk_root	CCGR1[CG5] (enet_clk_enable)	
ENET2	enet2_ipg_clk	ipg_clk_root	CCGR7[CG0] (enet_clk_enable)	
	enet2_ipg_clk_mac0	ipg_clk_root	CCGR7[CG0] (enet_clk_enable)	
	enet2_ipg_clk_mac0_s	ipg_clk_root	CCGR7[CG0] (enet_clk_enable)	
	enet2_ipg_clk_s	ipg_clk_root	CCGR7[CG0] (enet_clk_enable)	
	enet2_ipg_clk_time	ref_enetpll2_clk	CCGR7[CG0] (enet_clk_enable)	
	enet2_mem_mac0_mem_clk	ipg_clk_root	CCGR7[CG0] (enet_clk_enable)	
EWM	ewm_ipg_clk	ipg_clk_root	CCGR3[CG7] (ewm_clk_enable)	
	ewm_ipg_clk_s	ipg_clk_root	CCGR3[CG7] (ewm_clk_enable)	
	ewm_lpo_clk_0	ref_1m_clk (1MHz clock generated from the 24MHz RC oscillator)		
	ewm_lpo_clk_1	ckil_sync_clk_root		

Table continues on the next page...

Table 14-4. System Clocks, Gating, and Override (continued)

Module	Module Clock	Clock Root	Module Clock Gating Enable	Module Override Enable
FLEXIO*n*	flexio1_ipg_clk	ipg_clk_root	CCGR5[CG1] (flexio1_clk_enable)	
	flexio1_ipg_clk_s	ipg_clk_root	CCGR5[CG1] (flexio1_clk_enable)	
	flexio1_flexio_clk	flexio1_clk_root	CCGR5[CG1] (flexio1_clk_enable)	
	flexio2_ipg_clk	ipg_clk_root	CCGR3[CG0] (flexio2_clk_enable)	
	flexio2_ipg_clk_s	ipg_clk_root	CCGR3[CG0] (flexio2_clk_enable)	
	flexio2_flexio_clk	flexio2_clk_root	CCGR3[CG0] (flexio2_clk_enable)	
	flexio3_ipg_clk	ahb_clk_root	CCGR7[CG6] (flexio3_clk_enable)	
	flexio3_ipg_clk_s	ahb_clk_root	CCGR7[CG6] (flexio3_clk_enable)	
	flexio3_flexio_clk	flexio2_clk_root	CCGR7[CG6] (flexio3_clk_enable)	
FLEXPWM*n*	flexpwm1_ipg_clk_0	ipg_clk_root	CCGR4[CG8] (pwm1_clk_enable)	
	flexpwm1_ipg_clk_1	ipg_clk_root	CCGR4[CG8] (pwm1_clk_enable)	
	flexpwm1_ipg_clk_2	ipg_clk_root	CCGR4[CG8] (pwm1_clk_enable)	
	flexpwm1_ipg_clk_3	ipg_clk_root	CCGR4[CG8] (pwm1_clk_enable)	
	flexpwm1_ipg_clk_flt	ipg_clk_root	CCGR4[CG8] (pwm1_clk_enable)	
	flexpwm1_ipg_clk_cfg	ipg_clk_root	CCGR4[CG8] (pwm1_clk_enable)	
	flexpwm2_ipg_clk_0	ipg_clk_root	CCGR4[CG9] (pwm2_clk_enable)	
	flexpwm2_ipg_clk_1	ipg_clk_root	CCGR4[CG9] (pwm2_clk_enable)	
	flexpwm2_ipg_clk_2	ipg_clk_root	CCGR4[CG9] (pwm2_clk_enable)	
	flexpwm2_ipg_clk_3	ipg_clk_root	CCGR4[CG9] (pwm2_clk_enable)	
	flexpwm2_ipg_clk_flt	ipg_clk_root	CCGR4[CG9] (pwm2_clk_enable)	
	flexpwm2_ipg_clk_cfg	ipg_clk_root	CCGR4[CG9] (pwm2_clk_enable)	
	flexpwm3_ipg_clk_0	ipg_clk_root	CCGR4[CG10] (pwm3_clk_enable)	

Table continues on the next page...

Table 14-4. System Clocks, Gating, and Override (continued)

Module	Module Clock	Clock Root	Module Clock Gating Enable	Module Override Enable
	flexpwm3_ipg_clk_1	ipg_clk_root	CCGR4[CG10] (pwm3_clk_enable)	
	flexpwm3_ipg_clk_2	ipg_clk_root	CCGR4[CG10] (pwm3_clk_enable)	
	flexpwm3_ipg_clk_3	ipg_clk_root	CCGR4[CG10] (pwm3_clk_enable)	
	flexpwm3_ipg_clk_flt	ipg_clk_root	CCGR4[CG10] (pwm3_clk_enable)	
	flexpwm3_ipg_clk_cfg	ipg_clk_root	CCGR4[CG10] (pwm3_clk_enable)	
	flexpwm4_ipg_clk_0	ipg_clk_root	CCGR4[CG11] (pwm4_clk_enable)	
	flexpwm4_ipg_clk_1	ipg_clk_root	CCGR4[CG11] (pwm4_clk_enable)	
	flexpwm4_ipg_clk_2	ipg_clk_root	CCGR4[CG11] (pwm4_clk_enable)	
	flexpwm4_ipg_clk_3	ipg_clk_root	CCGR4[CG11] (pwm4_clk_enable)	
	flexpwm4_ipg_clk_flt	ipg_clk_root	CCGR4[CG11] (pwm4_clk_enable)	
	flexpwm4_ipg_clk_cfg	ipg_clk_root	CCGR4[CG11] (pwm4_clk_enable)	
FLEXRAM	flexram clock	ipg_clk_root	CCGR3[CG9] (flexram_clk_enable)	
FLEXSPI	flexspi_hclk	ipg_clk_root	CCGR6[CG5] (flexspi_clk_enable)	
	flexspi_ipg_clk	ipg_clk_root	CCGR6[CG5] (flexspi_clk_enable)	
	flexspi_ipg_clk_sfck	flexspi_clk_root	CCGR6[CG5] (flexspi_clk_enable)	
	flexspi_ipg_clk_s	ipg_clk_root	CCGR6[CG5] (flexspi_clk_enable)	
FLEXSPI2	flexspi2_hclk	ipg_clk_root	CCGR7[CG1] (flexspi_clk_enable)	
	flexspi2_ipg_clk	ipg_clk_root	CCGR7[CG1] (flexspi_clk_enable)	
	flexspi2_ipg_clk_sfck	flexspi2_clk_root	CCGR7[CG1] (flexspi_clk_enable)	
	flexspi2_ipg_clk_s	ipg_clk_root	CCGR7[CG1] (flexspi_clk_enable)	
GPC	gpc_ipg_clk	ipg_clk_root		
	gpc_ipg_clk_s	ipg_clk_root		
	gpc_pgc_clk	ipg_clk_root		
	gpc_sys_clk	ipg_clk_root		

Table continues on the next page...

Table 14-4. System Clocks, Gating, and Override (continued)

Module	Module Clock	Clock Root	Module Clock Gating Enable	Module Override Enable
GPIO*n*	gpio1_ipg_clk_s	ipg_clk_root	CCGR1[CG13] (gpio1_clk_enable)	
	gpio2_ipg_clk_s	ipg_clk_root	CCGR0[CG15] (gpio2_clk_enable)	
	gpio3_ipg_clk_s	ipg_clk_root	CCGR2[CG13] (gpio3_clk_enable)	
	gpio4_ipg_clk_s	ipg_clk_root	CCGR3[CG6] (gpio4_clk_enable)	
	gpio5_ipg_clk_s	ipg_clk_root	CCGR1[CG15] (gpio5_clk_enable)	
	gpio6_ipg_clk_s	ahb_clk_root		
	gpio7_ipg_clk_s	ahb_clk_root		
	gpio8_ipg_clk_s	ahb_clk_root		
	gpio9_ipg_clk_s	ahb_clk_root		
GPT*n*	gpt1_ipg_clk	perclk_clk_root	CCGR1[CG10] (gpt1_clk_enable)	CMEOR[mod_en_ov_gpt]
	gpt1_ipg_clk_24m	xtal_clkout		
	gpt1_ipg_clk_32k	ckil_sync_clk_root		
	gpt1_ipg_clk_highfreq	perclk_clk_root	CCGR1[CG11] (gpt1_serial_clk_enable)	
	gpt1_ipg_clk_s	perclk_clk_root	CCGR1[CG10] (gpt1_clk_enable)	
	gpt2_ipg_clk	perclk_clk_root	CCGR0[CG12] (gpt2_clk_enable)	CMEOR[mod_en_ov_gpt]
	gpt2_ipg_clk_24m	xtal_clkout		
	gpt2_ipg_clk_32k	ckil_sync_clk_root		
	gpt2_ipg_clk_highfreq	perclk_clk_root	CCGR0[CG13] (gpt2_serial_clk_enable)	
	gpt2_ipg_clk_s	perclk_clk_root	CCGR0[CG12] (gpt2_clk_enable)	
IOMUXC	iomuxc_ipg_clk_s	ipg_clk_root	CCGR4[CG1] (iomuxc_clk_enable)	
	iomuxc_snvs_ipg_clk_s	ipg_clk_root	CCGR2[CG2] (iomuxc_snvs_clk_enable)	
	iomuxc_snvs_gpr_ipg_clk_s	ipg_clk_root	CCGR3[CG15] (iomuxc_snvs_gpr_clk_enable)	
	iomuxc_gpr_ipg_clk_s	ipg_clk_root	CCGR4[CG2] (iomuxc_gpr_clk_enable)	
KPP	kpp_ipg_clk_32k	ckil_sync_clk_root		
	kpp_ipg_clk_s	ipg_clk_root	CCGR5[CG4] (kpp_clk_enable)	
LCDIF	lcdif_apb_clk	ipg_clk_root	CCGR2[CG14] (lcdif_clk_enable)	

Table continues on the next page...

Table 14-4. System Clocks, Gating, and Override (continued)

Module	Module Clock	Clock Root	Module Clock Gating Enable	Module Override Enable
	lcdif_pix_clk	lcdif_clk_root	CCGR3[CG5] (lcdif_pix_clk_enable)	
LPI2C*n*	lpi2c1_ipg_clk Wire	ipg_clk_root	CCGR2[CG3] (lpi2c1_clk_enable)	
	lpi2c1_ipg_clk_s	ipg_clk_root	CCGR2[CG3] (lpi2c1_clk_enable)	
	lpi2c1_lpi2c_clk	lpi2c_clk_root	CCGR2[CG3] (lpi2c1_clk_enable)	
	lpi2c1_lpi2c_div_clk	lpi2c_clk_root	CCGR2[CG3] (lpi2c1_clk_enable)	
	lpi2c2_ipg_clk Wire3 on MM	ipg_clk_root	CCGR2[CG4] (lpi2c2_clk_enable)	
	lpi2c2_ipg_clk_s	ipg_clk_root	CCGR2[CG4] (lpi2c2_clk_enable)	
	lpi2c2_lpi2c_clk	lpi2c_clk_root	CCGR2[CG4] (lpi2c2_clk_enable)	
	lpi2c2_lpi2c_div_clk	lpi2c_clk_root	CCGR2[CG4] (lpi2c2_clk_enable)	
	lpi2c3_ipg_clk Wire1 (Wire2 on MM)	ipg_clk_root	CCGR2[CG5] (lpi2c3_clk_enable)	
	lpi2c3_ipg_clk_s	ipg_clk_root	CCGR2[CG5] (lpi2c3_clk_enable)	
	lpi2c3_lpi2c_clk	lpi2c_clk_root	CCGR2[CG5] (lpi2c3_clk_enable)	
	lpi2c3_lpi2c_div_clk	lpi2c_clk_root	CCGR2[CG5] (lpi2c3_clk_enable)	
	lpi2c4_ipg_clk Wire2 (Wire1 on MM)	ipg_clk_root	CCGR6[CG12] (lpi2c4_clk_enable)	
	lpi2c4_ipg_clk_s	ipg_clk_root	CCGR6[CG12] (lpi2c4_clk_enable)	
	lpi2c4_lpi2c_clk	lpi2c_clk_root	CCGR6[CG12] (lpi2c4_clk_enable)	
	lpi2c4_lpi2c_div_clk	lpi2c_clk_root	CCGR6[CG12] (lpi2c4_clk_enable)	
LPSPI*n*	lpspi1_ipg_clk SPI2	ipg_clk_root	CCGR1[CG0] (lpspi1_clk_enable)	
	lpspi1_lpspi_clk	lpspi_clk_root	CCGR1[CG0] (lpspi1_clk_enable)	
	lpspi1_lpspi_div_clk	lpspi_clk_root	CCGR1[CG0] (lpspi1_clk_enable)	
	lpspi1_ipg_clk_s	ipg_clk_root	CCGR1[CG0] (lpspi1_clk_enable)	
	lpspi2_ipg_clk	ipg_clk_root	CCGR1[CG1] (lpspi2_clk_enable)	

Table continues on the next page...

Table 14-4. System Clocks, Gating, and Override (continued)

Module	Module Clock	Clock Root	Module Clock Gating Enable	Module Override Enable
	lpspi2_lpspi_clk	lpspi_clk_root	CCGR1[CG1] (lpspi2_clk_enable)	
	lpspi2_lpspi_div_clk	lpspi_clk_root	CCGR1[CG1] (lpspi2_clk_enable)	
	lpspi2_ipg_clk_s	ipg_clk_root	CCGR1[CG1] (lpspi2_clk_enable)	
	lpspi3_ipg_clk SPI1	ipg_clk_root	CCGR1[CG2] (lpspi3_clk_enable)	
	lpspi3_lpspi_clk	lpspi_clk_root	CCGR1[CG2] (lpspi3_clk_enable)	
	lpspi3_lpspi_div_clk	lpspi_clk_root	CCGR1[CG2] (lpspi3_clk_enable)	
	lpspi3_ipg_clk_s	ipg_clk_root	CCGR1[CG2] (lpspi3_clk_enable)	
	lpspi4_ipg_clk SPI	ipg_clk_root	CCGR1[CG3] (lpspi4_clk_enable)	
	lpspi4_lpspi_clk	lpspi_clk_root	CCGR1[CG3] (lpspi4_clk_enable)	
	lpspi4_lpspi_div_clk	lpspi_clk_root	CCGR1[CG3] (lpspi4_clk_enable)	
	lpspi4_ipg_clk_s	ipg_clk_root	CCGR1[CG3] (lpspi4_clk_enable)	
LPUART*n*	lpuart1_ipg_clk	ipg_clk_root	CCGR5[CG12] (lpuart1_clk_enable)	
	lpuart1_ipg_clk_s	ipg_clk_root	CCGR5[CG12] (lpuart1_clk_enable)	
	lpuart1_lpuart_baud_clk	uart_clk_root		
	lpuart1_lpuart_baud_gated_clk	uart_clk_root	CCGR5[CG12] (lpuart1_clk_enable)	
	lpuart2_ipg_clk	ipg_clk_root	CCGR0[CG14] (lpuart2_clk_enable)	
	lpuart2_ipg_clk_s	ipg_clk_root	CCGR0[CG14] (lpuart2_clk_enable)	
	lpuart2_lpuart_baud_clk	uart_clk_root		
	lpuart2_lpuart_baud_gated_clk	uart_clk_root	CCGR0[CG14] (lpuart2_clk_enable)	
	lpuart3_ipg_clk	ipg_clk_root	CCGR0[CG6] (lpuart3_clk_enable)	
	lpuart3_ipg_clk_s	ipg_clk_root	CCGR0[CG6] (lpuart3_clk_enable)	
	lpuart3_lpuart_baud_clk	uart_clk_root		
	lpuart3_lpuart_baud_gated_clk	uart_clk_root	CCGR0[CG6] (lpuart3_clk_enable)	
	lpuart4_ipg_clk	ipg_clk_root	CCGR1[CG12] (lpuart4_clk_enable)	

Table continues on the next page...

Table 14-4. System Clocks, Gating, and Override (continued)

Module	Module Clock	Clock Root	Module Clock Gating Enable	Module Override Enable
	lpuart4_ipg_clk_s	ipg_clk_root	CCGR1[CG12] (lpuart4_clk_enable)	
	lpuart4_lpuart_baud_clk	uart_clk_root		
	lpuart4_lpuart_baud_gated_clk	uart_clk_root	CCGR1[CG12] (lpuart4_clk_enable)	
	lpuart5_ipg_clk	ipg_clk_root	CCGR3[CG1] (lpuart5_clk_enable)	
	lpuart5_ipg_clk_s	ipg_clk_root	CCGR3[CG1] (lpuart5_clk_enable)	
	lpuart5_lpuart_baud_clk	uart_clk_root		
	lpuart5_lpuart_baud_gated_clk	uart_clk_root	CCGR3[CG1] (lpuart5_clk_enable)	
	lpuart6_ipg_clk	ipg_clk_root	CCGR3[CG3] (lpuart6_clk_enable)	
	lpuart6_ipg_clk_s	ipg_clk_root	CCGR3[CG3] (lpuart6_clk_enable)	
	lpuart6_lpuart_baud_clk	uart_clk_root		
	lpuart6_lpuart_baud_gated_clk	uart_clk_root	CCGR3[CG3] (lpuart6_clk_enable)	
	lpuart7_ipg_clk	ipg_clk_root	CCGR5[CG13] (lpuart7_clk_enable)	
	lpuart7_ipg_clk_s	ipg_clk_root	CCGR5[CG13] (lpuart7_clk_enable)	
	lpuart7_lpuart_baud_clk	uart_clk_root		
	lpuart7_lpuart_baud_gated_clk	uart_clk_root	CCGR5[CG13] (lpuart7_clk_enable)	
	lpuart8_ipg_clk	ipg_clk_root	CCGR6[CG7] (lpuart8_clk_enable)	
	lpuart8_ipg_clk_s	ipg_clk_root	CCGR6[CG7] (lpuart8_clk_enable)	
	lpuart8_lpuart_baud_clk	uart_clk_root		
	lpuart8_lpuart_baud_gated_clk	uart_clk_root	CCGR6[CG7] (lpuart8_clk_enable)	
MQS	mqs clock	sai3_clk_root	CCGR0[CG2] (mqs_hmclk_clock_enable)	
OCOTP	ocotp_ctrl_wrapper_ipg_clk	ipg_clk_root	CCGR2[CG6] (ocotp_clk_enable)	
	ocotp_ctrl_wrapper_ipg_clk_s	ipg_clk_root	CCGR2[CG6] (ocotp_clk_enable)	
PIT	pit_ipg_clk	perclk_clk_root	CCGR1[CG6] (pit_clk_enable)	CMEOR[mod_en_ov_pit]
	pit_ipg_clk_osc_rti	ckil_sync_clk_root		
	pit_ipg_clk_sync	perclk_clk_root	CCGR1[CG6] (pit_clk_enable)	
	pit_ipg_clk_s	perclk_clk_root	CCGR1[CG6] (pit_clk_enable)	

Table continues on the next page...

Table 14-4. System Clocks, Gating, and Override (continued)

Module	Module Clock	Clock Root	Module Clock Gating Enable	Module Override Enable
PXP	pxp_clk	ipg_clk_root	CCGR2[CG15] (pxp_clk_enable)	
QDC*n*	qdc1_ipg_clk	ipg_clk_root	CCGR4[CG12] (qdc1_clk_enable)	
	qdc2_ipg_clk	ipg_clk_root	CCGR4[CG13] (qdc2_clk_enable)	
	qdc3_ipg_clk	ipg_clk_root	CCGR4[CG14] (qdc3_clk_enable)	
	qdc4_ipg_clk	ipg_clk_root	CCGR4[CG15] (qdc4_clk_enable)	
QTIMER*n*	qtimer1_clk0	ipg_clk_root	CCGR6[CG13] (qtimer1_clk_enable)	
	qtimer1_clk1	ipg_clk_root	CCGR6[CG13] (qtimer1_clk_enable)	
	qtimer1_clk2	ipg_clk_root	CCGR6[CG13] (qtimer1_clk_enable)	
	qtimer1_clk3	ipg_clk_root	CCGR6[CG13] (qtimer1_clk_enable)	
	qtimer2_clk0	ipg_clk_root	CCGR6[CG14] (qtimer2_clk_enable)	
	qtimer2_clk1	ipg_clk_root	CCGR6[CG14] (qtimer2_clk_enable)	
	qtimer2_clk2	ipg_clk_root	CCGR6[CG14] (qtimer2_clk_enable)	
	qtimer2_clk3	ipg_clk_root	CCGR6[CG14] (qtimer2_clk_enable)	
	qtimer3_clk0	ipg_clk_root	CCGR6[CG15] (qtimer3_clk_enable)	
	qtimer3_clk1	ipg_clk_root	CCGR6[CG15] (qtimer3_clk_enable)	
	qtimer3_clk2	ipg_clk_root	CCGR6[CG15] (qtimer3_clk_enable)	
	qtimer3_clk3	ipg_clk_root	CCGR6[CG15] (qtimer3_clk_enable)	
	qtimer4_clk0	ipg_clk_root	CCGR6[CG8] (qtimer4_clk_enable)	
	qtimer4_clk1	ipg_clk_root	CCGR6[CG8] (qtimer4_clk_enable)	
	qtimer4_clk2	ipg_clk_root	CCGR6[CG8] (qtimer4_clk_enable)	
	qtimer4_clk3	ipg_clk_root	CCGR6[CG8] (qtimer4_clk_enable)	
SAI*n*	sai1_ipg_clk	ipg_clk_root	CCGR5[CG9] (sai1_clk_enable)	

Table continues on the next page...

Table 14-4. System Clocks, Gating, and Override (continued)

Module	Module Clock	Clock Root	Module Clock Gating Enable	Module Override Enable
	sai1_ipg_clk_s	ipg_clk_root	CCGR5[CG9] (sai1_clk_enable)	
	sai1_ipg_clk_sai_mclk[1]	sai1_mclk1_mux_clk IOMUXC_GPR_GPR1[SAI1_MCLK1_SEL]	CCGR5[CG9] (sai1_clk_enable)	
	sai1_ipg_clk_sai_mclk[2]	sai1_mclk2_mux_clk IOMUXC_GPR_GPR1[SAI1_MCLK2_SEL]	CCGR5[CG9] (sai1_clk_enable)	
	sai1_ipg_clk_sai_mclk[3]	sai1_mclk3_mux_clk IOMUXC_GPR_GPR1[SAI1_MCLK3_SEL]	CCGR5[CG9] (sai1_clk_enable)	
	sai1_ipt_clk_sai_bclk	sai1_clk_root	CCGR5[CG9] (sai1_clk_enable)	
	sai1_ipt_clk_sai_bclk_b	sai1_clk_root	CCGR5[CG9] (sai1_clk_enable)	
	sai1_ipt_clk_sai_mclk	sai1_clk_root	CCGR5[CG9] (sai1_clk_enable)	
	sai2_ipg_clk	ipg_clk_root	CCGR5[CG10] (sai2_clk_enable)	
	sai2_ipg_clk_s	ipg_clk_root	CCGR5[CG10] (sai2_clk_enable)	
	sai2_ipg_clk_sai_mclk[1]	sai2_clk_root	CCGR5[CG10] (sai2_clk_enable)	
	sai2_ipg_clk_sai_mclk[2]	iomux_bottom.sai2_ipg_clk_sai_mclk IOMUXC_GPR_GPR1[SAI2_MCLK2_SEL]	CCGR5[CG10] (sai2_clk_enable)	
	sai2_ipg_clk_sai_mclk[3]	sai2_mclk3_mux_clk IOMUXC_GPR_GPR1[SAI2_MCLK3_SEL]	CCGR5[CG10] (sai2_clk_enable)	
	sai2_ipt_clk_sai_bclk	sai2_clk_root	CCGR5[CG10] (sai2_clk_enable)	
	sai2_ipt_clk_sai_bclk_b	sai2_clk_root	CCGR5[CG10] (sai2_clk_enable)	
	sai2_ipt_clk_sai_mclk	sai2_clk_root	CCGR5[CG10] (sai2_clk_enable)	
	sai3_ipg_clk	ipg_clk_root	CCGR5[CG11] (sai3_clk_enable)	
	sai3_ipg_clk_s	ipg_clk_root	CCGR5[CG11] (sai3_clk_enable)	
	sai3_ipg_clk_sai_mclk[1]	sai3_clk_root	CCGR5[CG11] (sai3_clk_enable)	
	sai3_ipg_clk_sai_mclk[2]	iomux_top.sai3_ipg_clk_sai_mclk	CCGR5[CG11] (sai3_clk_enable)	

Table continues on the next page...

Table 14-4. System Clocks, Gating, and Override (continued)

Module	Module Clock	Clock Root	Module Clock Gating Enable	Module Override Enable
		IOMUXC_GPR_GPR1[SAI3_MCLK2_SEL]		
	sai3_ipg_clk_sai_mclk[3]	sai3_mclk3_mux_clk IOMUXC_GPR_GPR1[SAI3_MCLK3_SEL]	CCGR5[CG11] (sai3_clk_enable)	
	sai3_ipt_clk_sai_bclk	sai3_clk_root	CCGR5[CG11] (sai3_clk_enable)	
	sai3_ipt_clk_sai_bclk_b	sai3_clk_root	CCGR5[CG11] (sai3_clk_enable)	
	sai3_ipt_clk_sai_mclk	sai3_clk_root	CCGR5[CG11] (sai3_clk_enable)	
SEMC	semc clock	semc_clk_root	CCGR3[CG2] (semc_clk_enable)	
SNVS	snvs_hp_wrapper_ipg_clk	ipg_clk_root	CCGR5[CG14] (snvs_hp_clk_enable)	
	snvs_hp_wrapper_ipg_clk_s	ipg_clk_root	CCGR5[CG14] (snvs_hp_clk_enable)	
	snvs_hp_wrapper_ipg_hp_rtc_clk	ckil_sync_clk_root		
	snvs_lp_wrapper_ipg_clk	ipg_clk_root	CCGR5[CG15] (snvs_lp_clk_enable)	
	snvs_lp_wrapper_ipg_clk_s	ipg_clk_root	CCGR5[CG15] (snvs_lp_clk_enable)	
	snvs_lp_wrapper_ipg_dryice_clk_s	ipg_clk_root		
SPDIF	spdif_gclkw_t0	ipg_clk_root	CCGR5[CG7] (spdif_clk_enable)	
	spdif_ipg_clk_s	ipg_clk_root		
	spdif_tx_clk	spdif0_clk_root	CCGR5[CG7] (spdif_clk_enable)	
SRC	src_src_ipg_clk_s	ipg_clk_root		
TRNG	trng_ipg_clk	ipg_clk_root	CCGR6[CG6] (trng_clk_enable)	CMEOR[mod_en_ov_trng]
TSC	tsc_dig_ipg_clk	ipg_clk_root	CCGR4[CG5] (tsc_clk_enable)	
	tsc_dig_ipg_clk_s	ipg_clk_root	CCGR4[CG5] (tsc_clk_enable)	
	tsc_dig_lp_clk	ckil_sync_clk_root		
USB	usb_ipg_ahb_clk	ipg_clk_root	CCGR6[CG0] (usboh3_clk_enable)	
	usb_ipg_clk_32khz	ckil_sync_clk_root		
	usb_ipg_clk_s	ipg_clk_root	CCGR6[CG0] (usboh3_clk_enable)	
	usb_ipg_clk_s_pl301	ipg_clk_root	CCGR6[CG0] (usboh3_clk_enable)	

Table continues on the next page...

Table 14-4. System Clocks, Gating, and Override (continued)

Module	Module Clock	Clock Root	Module Clock Gating Enable	Module Override Enable
USDHC*n*	usdhc1_hclk	ipg_clk_root	CCGR6[CG1] (usdhc1_clk_enable)	CMEOR[mod_en_ov_usdhc]
	usdhc1_ipg_clk	ipg_clk_root	CCGR6[CG1] (usdhc1_clk_enable)	CMEOR[mod_en_ov_usdhc]
	usdhc1_ipg_clk_perclk	usdhc1_clk_root	CCGR6[CG1] (usdhc1_clk_enable)	CMEOR[mod_en_ov_usdhc]
	usdhc1_ipg_clk_s	ipg_clk_root	CCGR6[CG1] (usdhc1_clk_enable)	CMEOR[mod_en_ov_usdhc]
	usdhc2_hclk	ipg_clk_root	CCGR6[CG2] (usdhc2_clk_enable)	CMEOR[mod_en_ov_usdhc]
	usdhc2_ipg_clk	ipg_clk_root	CCGR6[CG2] (usdhc2_clk_enable)	CMEOR[mod_en_ov_usdhc]
	usdhc2_ipg_clk_perclk	usdhc2_clk_root	CCGR6[CG2] (usdhc2_clk_enable)	CMEOR[mod_en_ov_usdhc]
	usdhc2_ipg_clk_s	ipg_clk_root	CCGR6[CG2] (usdhc2_clk_enable)	CMEOR[mod_en_ov_usdhc]
WDOG*n*	wdog1_ipg_clk	ipg_clk_root	CCGR3[CG8] (wdog1_clk_enable)	
	wdog1_ipg_clk_32k	ckil_sync_clk_root		
	wdog1_ipg_clk_s	ipg_clk_root	CCGR3[CG8] (wdog1_clk_enable)	
	wdog2_ipg_clk	ipg_clk_root	CCGR5[CG5] (wdog2_clk_enable)	
	wdog2_ipg_clk_32k	ckil_sync_clk_root		
	wdog2_ipg_clk_s	ipg_clk_root	CCGR5[CG5] (wdog2_clk_enable)	
	wdog3_ipg_clk_s	ipg_clk_root		
	wdog3_ipg_clk	ipg_clk_root	CCGR5[CG2] (wdog3_clk_enable)	
	wdog3_lpo_clk	ckil_sync_clk_root		
	wdog3_int_clk	ckil_sync_clk_root		
	wdog3_ext_clk	ref_1m_clk (1MHz clock generated from the 24MHz RC oscillator)		
XBAR*n*	xbar1_ipb_clk	ipg_clk_root	CCGR2[CG11] (xbar1_clk_enable)	
	xbar2_ipb_clk	ipg_clk_root	CCGR2[CG12] (xbar2_clk_enable)	
	xbar3_ipb_clk	ipg_clk_root	CCGR2[CG7] (xbar3_clk_enable)	

Table 14-5. System Clock Frequency Values

Clock Root	Default Frequency (POR) (MHz)	Maximum Frequency (MHz)				
		Overdrive Run	Full Speed Run	Low Power Run	System Idle	Low Power Idle
VDD_SOC_IN voltage		1.25V min	1.15V min	0.925V min	1.15V min	0.925V min
AHB_CLK_ROOT	12	600	528	24	528	24
IPG_CLK_ROOT	3	150	132	24	132	24
PERCLK_CLK_ROOT	3	75	72	24	72	24
USDHC1_CLK_ROOT	12	200	200	24	200	24
USDHC2_CLK_ROOT	12	200	200	24	200	24
SEMC_CLK_ROOT	8	166	166	24	166	24
CSI_CLK_ROOT	12	80	80	24	80	24
FLEXSPI_CLK_ROOT	2	332	332	24	332	24
FLEXSPI2_CLK_ROOT	12	332	332	24	332	24
LPSPI_CLK_ROOT	6	132	132	24	132	24
TRACE_CLK_ROOT	6	150	132	24	132	24
SAI1_CLK_ROOT	3	66	66	24	66	24
SAI2_CLK_ROOT	3	66	66	24	66	24
SAI3_CLK_ROOT	3	66	66	24	66	24
LPI2C_CLK_ROOT	24	66	66	24	66	24
CAN_CLK_ROOT	OFF	80	80	24	80	24
UART_CLK_ROOT	4	80	80	24	80	24
LCDIF_CLK_ROOT	3	75	75	24	75	24
SPDIF0_CLK_ROOT	1.5	66	66	24	66	24
FLEXIO1_CLK_ROOT	1.5	120	120	24	120	24
FLEXIO2_CLK_ROOT	1.5	120	120	24	120	24

14.6 Functional Description

This section provides a complete functional description of the block.

14.6.1 Clock Generation

14.6.1.1 External Low Frequency Clock - CKIL

The chip can use a 32 kHz or 32.768 kHz crystal as the external low-frequency source (XTALOSC). Throughout this chapter, the low-frequency crystal is referred to as the 32 kHz crystal.

This clock source should always be active when the chip is powered on. The 32 kHz entering the CCM are referred to as CKIL. CKIL is synchronized to IPG_CLK and supplied to modules that need it.

14.6.1.1.1 CKIL synchronizing to IPG_CLK

CKIL is synchronized to ipg_clk when the system is in functional mode. When the system is in STOP mode (when there is no IPG_CLK) the CKIL synchronizer is bypassed, and raw CKIL is supplied to the system.

14.6.1.2 External High Frequency Clock - CKIH and internal oscillator

The chip uses an internal oscillator to generate the reference clock (OSC). The internal oscillator is connected to the external crystal (XTALOSC) which generates the 24 MHz reference clock.

14.6.1.3 PLL reference clock

There are several PLLs in this chip.

PLL1 - ARM PLL (typical functional frequency)

PLL2 - System PLL (functional frequency 528 MHz)

PLL3 - USB1 PLL (functional frequency 480 MHz)

PLL4 - Audio PLL

PLL5 - Video PLL

PLL6 - ENET PLL

PLL7 - USB2 PLL (functional frequency 480 MHz)

Some of the PLLs are described in the sections below. See CCM Analog Memory Map/ Register Definition for register information.

14.6.1.3.1 ARM PLL (PLL1)

This PLL synthesizes a low jitter clock from a 24 MHz reference clock. The clock output frequency for this PLL ranges from 650 MHz to 1.3 GHz. The output frequency is selected by a 7-bit register field CCM_ANALOG_PLL_ARM[DIV_SELECT].

PLL output frequency = Fref * DIV_SEL/2

NOTE

The upper frequency range may exceed the maximum frequency supported. Please see the datasheet for more information.

14.6.1.3.2 System PLL (PLL2)

This PLL synthesizes a low jitter clock from the 24 MHz reference clock. The PLL has one output clock, plus 4 PFD outputs. The System PLL supports spread spectrum modulation for use in applications to minimize radiated emissions. The spread spectrum PLL output clock is frequency modulated so that the energy is spread over a wider bandwidth, thereby reducing peak radiated emissions. Due to this feature support, the associated lock time of this PLL is longer than other PLLs in the SoC that do not support spread spectrum modulation.

Spread spectrum operation is controlled by configuring the CCM_ANALOG_PLL_SYS_SS register. When enabled, the PLL output frequency will decrease by the amount defined in the STEP field, until it reaches the limiting frequency in the STOP field. The frequency will then similarly return to the original nominal frequency. The following equations control the spread-spectrum operation:

$$\text{Spread spectrum range} = \text{Fref} \times \frac{\text{CCM_ANALOG_PLL_SYS_SS[STOP]}}{\text{CCM_ANALOG_PLL_SYS_DENOM[B]}}$$

$$\text{Modulation frequency} = \text{Fref} \times \frac{\text{CCM_ANALOG_PLL_SYS_SS[STEP]}}{2 \times \text{CCM_ANALOG_PLL_SYS_SS[STOP]}}$$

Although this PLL does have a DIV_SELECT register field, it is intended that this PLL will only be run at the default frequency of 528 MHz.

This PLL also supports a Fractional-N synthesizer.

14.6.1.3.3 USB1 PLL (PLL3)

These PLLs synthesize a low jitter clock from the 24 MHz reference clock. USB1 PLL has 4 frequency-programmable PFD (phase fractional divider) outputs.

The output frequency of USB1 PLL is 480 MHz. Even though USB1 PLL has a DIV_SELECT register field, this PLL should always be set to 480 MHz in normal operation.

14.6.1.3.4 Audio PLL (PLL4)

The audio PLL synthesize a low jitter clock from a 24 MHz reference clock. The clock output frequency range for this PLL is from 650 MHz to 1.3 GHz. It has a Fractional-N synthesizer.

There are /1, /2, /4 post dividers for the Audio PLL. The output frequency can be set by programming the fields in the CCM_ANALOG_PLL_AUDIO, and CCM_ANALOG_MISC2 register sets according to the following equation.

PLL output frequency = Fref * (DIV_SELECT + NUM/DENOM)

14.6.1.3.5 Video PLL (PLL5)

The video PLL synthesize a low jitter clock from a 24 MHz reference clock. The clock output frequency range for this PLL is from 650 MHz to 1.3 GHz. It has a Fractional-N synthesizer.

There are /1, /2, /4, /8, /16 post dividers for the Video PLL. The output frequency can be set by programming the fields in the CCM_ANALOG_PLL_VIDEO, and CCM_ANALOG_MISC2 register sets according to the following equation.

PLL output frequency = Fref * (DIV_SELECT + NUM/DENOM)

14.6.1.3.6 Ethernet PLL (PLL6)

This PLL synthesizes a low jitter clock from the 24 MHz reference clock.

The reference clocks generated by this PLL are:

- ref_enetpll1 programmable to 25, 50, 100 and 125 MHz by setting CCM_ANALOG_PLL_ENET[DIV_SELECT] bitfield
- ref_enetpll2 fixed at 25 MHz

14.6.1.3.7 USB2 PLL (PLL7)

USB2 PLL is only used by the USB UTM interface through a direct connection.

14.6.1.4 Phase Fractional Dividers (PFD)

There are several PFD outputs from the System PLL and USB1 PLL.

Each PFD output generates a fractional multiplication of the associated PLL's VCO frequency. Where the output frequency is equal to Fvco*18/N, N can range from 12-35. The PFDs allow for clock frequency changes without forcing the relock of the root PLL. This feature is useful in support of dynamic voltage and frequency scaling (DVFS). See CCM Analog Memory Map/Register Definition.

When the related PLL is powered up from the power down state or made to go through a relock cycle due to PLL regrogramming, it is required that the related PFDx_CLKGATE bit in CCM_ANALOG_PFD_480n or CCM_ANALOG_PFD_528n, be cycled on and off (1 to 0) after PLL lock. The PFDs can be in the clock gated state during PLL relock but must be un-clock gated only after lock is achieved. See the engineering bulletin, *Configuration of Phase Fractional Dividers (EB790)* at www.nxp.com for procedure details. CAUTION: setting the PFD CLKGATE bits (to temporarily disable clock output) has seemed to cause very strange problems, especially PLL3 PFD0. See define GATE_PFD_WHILE_CHANGE in core library startup.c and LittleFS issue #63: https://github.com/PaulStoffregen/LittleFS/issues/63

14.6.1.5 CCM internal clock generation

The clock generation is comprised of two sub-modules:

CCM_CLK_SWITCHER

CCM_CLK_ROOT_GEN

14.6.1.5.1 Clock Switcher

The Clock Switcher (CCM_CLK_SWITCHER) sub-module receives the PLL output clocks and the PLL bypass clocks.

Figure 14-4 describes the generation of the three switcher clocks.

The figure also includes the Frequency Switch Control sub-module responsible for frequency change.

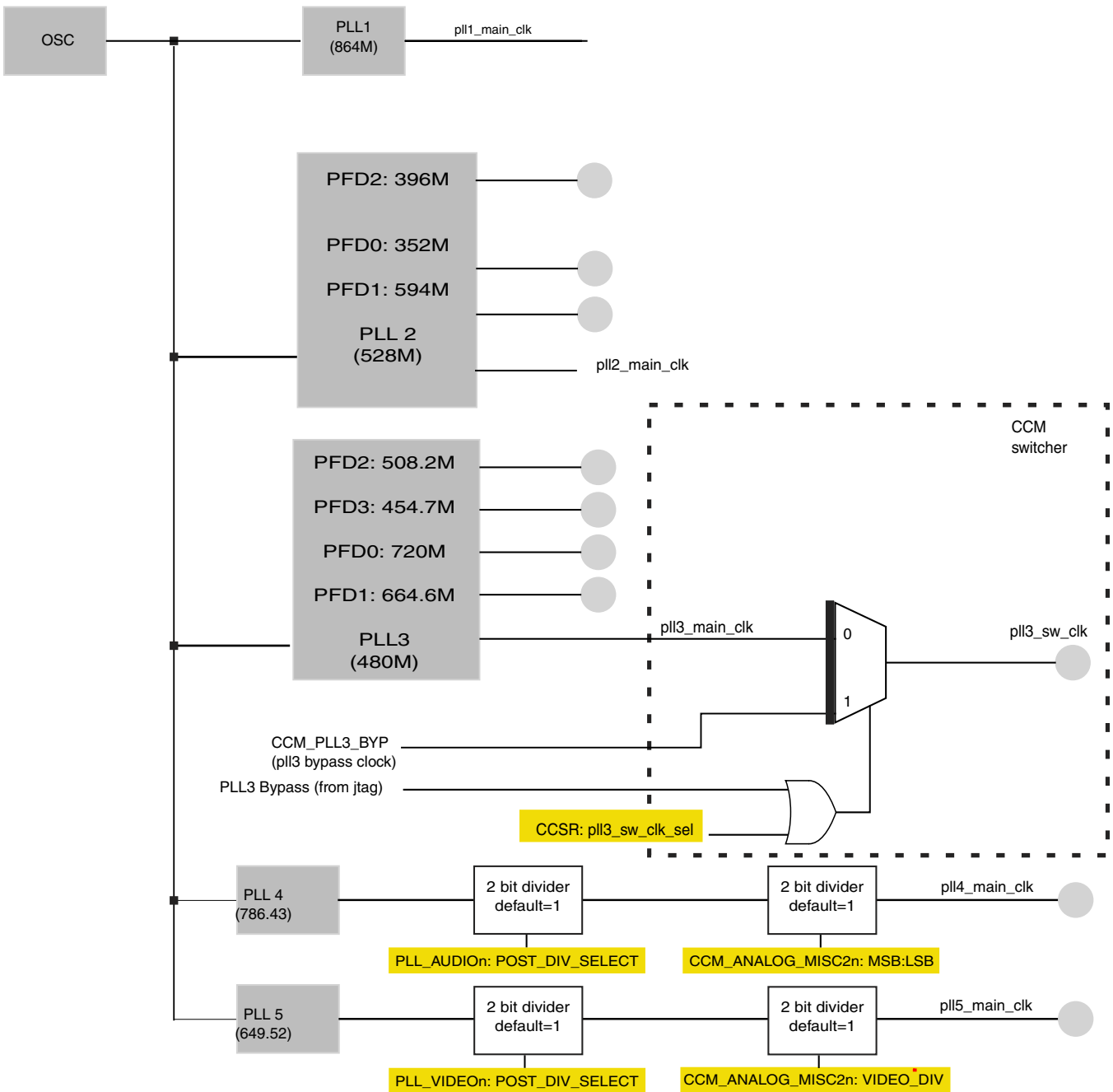


Figure 14-4. Switcher clock generation

14.6.1.5.2 PLL bypass procedure

In addition to PLL bypass options in CCM_ANALOG module, switcher and clk_root_gen sub-modules includes capability for each of the PLL clocks to be bypassed with an external bypass clock.

14.6.1.5.3 PLL clock change

In order to modify or stop the clock output of a specific PLL, all the clocks generated from the current PLL must be transitioned to the new PLL whose frequency is not being modified.

For clocks which can't be stopped (core and bus clocks), this should be done via the glitchless mux. Before changing the PLL setting, power it down. Power up the PLL after the change. See Disabling / Enabling PLLs for more information.

14.6.1.5.4 Clock Root Generator

The Clock Root Generator (CCM_CLK_ROOT_GEN) sub-module generates the root clocks to be delivered to LPCG.

The following figures describe clock generation. The frequencies in parantheses are the default typical frequencies.

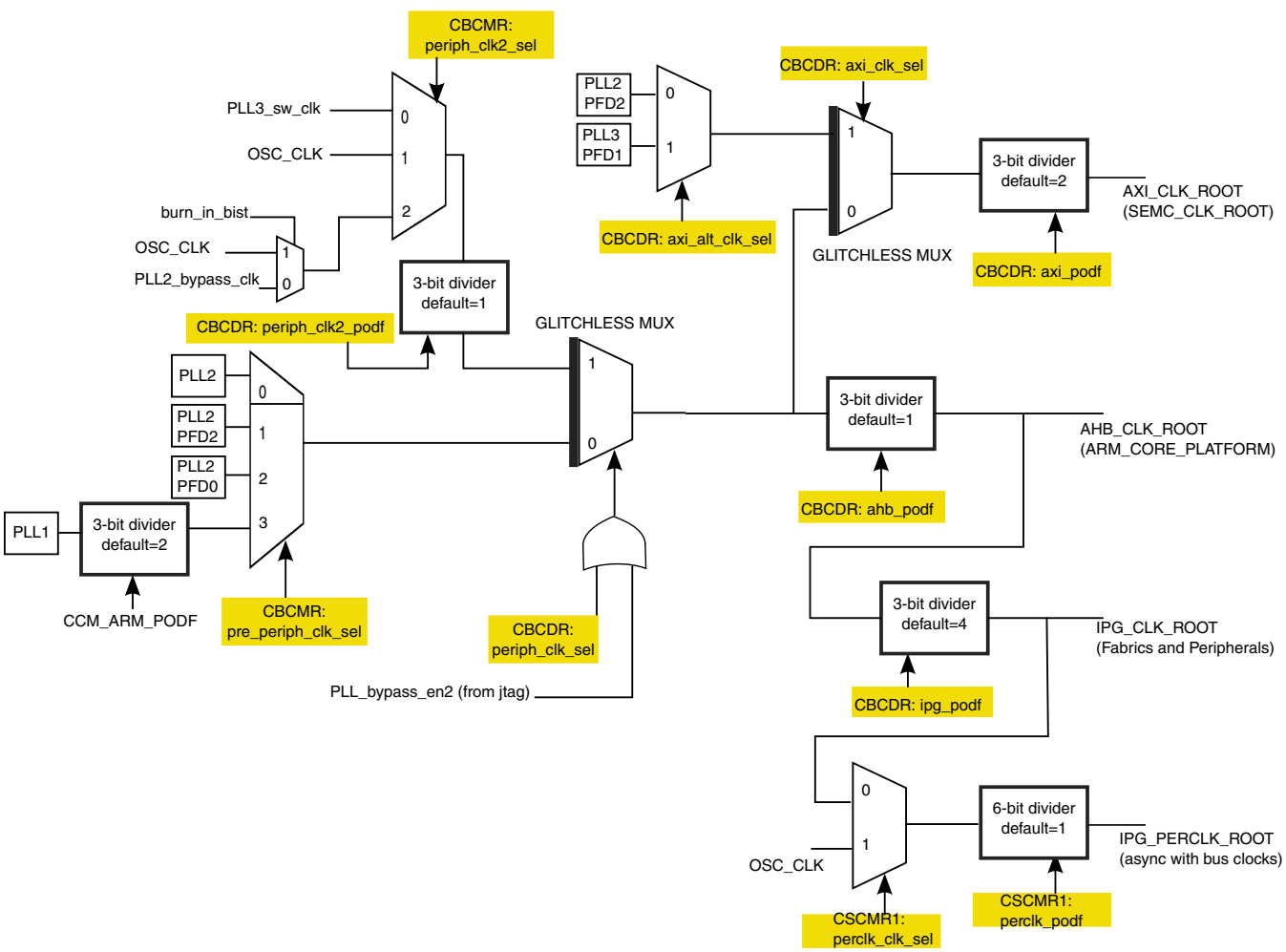


Figure 14-5. BUS clock generation

NOTE

All 6-bit PODF dividers found in the diagrams above can operate on low frequency.

14.6.1.5.5 Divider change handshake

Modifying the following dividers will start the handshake .

- periph_clk_sel
- arm_podf
- ahb_podf

The dividers listed above are designed with a handshake. For dividers without a handshake design, the following sequence must be performed when updating PODF value:

1. Gate the output clock off before updating PODF value.
2. Gate the output clock on after the PODF value is updated and stable.

To update the PODF value without gating the output clock off will cause unpredictable results such as no clock output.

14.6.1.6 Disabling / Enabling PLLs

PLL disabling and enabling is done via analog module.

Before disabling a PLL using the analog registers, software should first move all the clocks generated from that specific PLL to another source. This alternate source could be another PLL, or a PFD driven by another PLL. Alternatively, software can bypass the PLL and use the PLL reference clock (usually 24 MHz) as the output clock. Bypassing the PLL is done by setting the analog BYPASS bit in the control register for that PLL.

14.6.1.7 Clock Switching Multiplexers

There are a multitude of multiplexers available throughout the clock generation logic that provide alternate clock sources for the system clocks controlled by the CCM. The CCM uses several synchronous glitchless clock multiplexers as well as asynchronous glitchy clock multiplexers.

Synchronous muxes ensure there are no glitches between the transition of two asynchronous clocks and that there will be no pulses that are of a frequency higher than either input clock. For the synchronous multiplexer to work properly, both the current clock and the clock to be selected must remain active during the entire selection process.

There are several glitchless (synchronous) muxes used in the CCM. The table below lists the muxes and the respective control bits.

Table 14-6. Glitchless Multiplexers

Glitchless Mux	Mux Select Bit	Handshake Bit
periph_clk_mux	CBCDR[periph_clk_sel]	CDHIPR[periph_clk_sel_busy]
semc_clk_mux	CBCDR[semc_clk_sel]	
pll3_sw_clk_mux	CCSR[pll3_sw_clk_sel]	

NOTE

Any change of the periph_clk_sel sync mux select will involve handshake. Refer to the CDHIPR register for the handshake busy bits.

For critical system bus clocks, changing the clock source can be done in the CCM using the glitchless clock muxes in Figure 14-5. In the figure, the thick bar on the input side indicates the glitchless muxes. Those without the thick bar are regular muxes (not glitchless).

For example, before disabling PLL2, software can switch the AHB_CLK_ROOT away from the PLL2 or one of its PFDs by programming CBCMR[PERIPH_CLK2_SEL] and CBCDR[PERIPH_CLK2_PODF] to provide an appropriate frequency clock, then glitchlessly switch to it by programming CBCDR[PERIPH_CLK_SEL].

Asynchronous multiplexers or glitchy multiplexers, allow the clock to switch immediately after the multiplexer select changed. This immediate switch of two asynchronous clock domains can cause the output clock to glitch. Since both clock sources to the mux are asynchronous, switching the clocks from one source to the other can cause a glitch to be generated, regardless of the input clock source.

The output clocks to the mux are required to be gated before switching the source clock in the CCM clock mux. If the output clocks are not gated, clock glitches can propagate to the logic that follows the clock mux, causing the logic to behave unpredictably.

For serial clocks, software should first disable the module, then gate its clock in the LPCG. Then it should move the mux controlling the source of the clocks to another PLL, and reset the module and its clocks. Only then is it safe to disable the PLL. The mux for the serial clocks is not glitchless.

14.6.1.8 Low Power Clock Gating module (LPCG)

The LPCG module receives the root clocks and splits them to clock branches for each module. The clock branches are gated clocks.

The enables for those gates can come from four sources:

1. Clock enable signal from CCM - this signal is generated by configuring of the CGR bits in the CCM. It is based on the low power mode.
2. Clock enable signal from the module - this signal is generated by the module based on internal logic of the module. Not every enable signal from the module is used. For used clock enable signals from the module, CCM will generate an override signal based on a programable bit in CCM (CMEOR).
3. Clock enable signal from Reset controller (SRC) - this signal will enable the clock during the reset procedure. Please see the SRC chapter for details on the clock enable signal during reset procedure.
4. Hard-coded enable from fuse box.

These enable signals are ANDed to generate the enable signal for the gating cell.

The enable signal for the gating cell is synchronized with the clock it needs to gate in order to prevent glitches on the gated clock.

Notifications are generated for CCM to indicate when clock roots should be opened and closed. All notifications that correspond to the same clock root will be ORed to generate one notification signal to CCM for clock root gating.

The following figure describes the clock split inside the LPCG module. It describes the case of two modules; one module is without an enable signal and one is shown with an enable signal. SRC enable signals and sync flip flops are omitted from this figure.

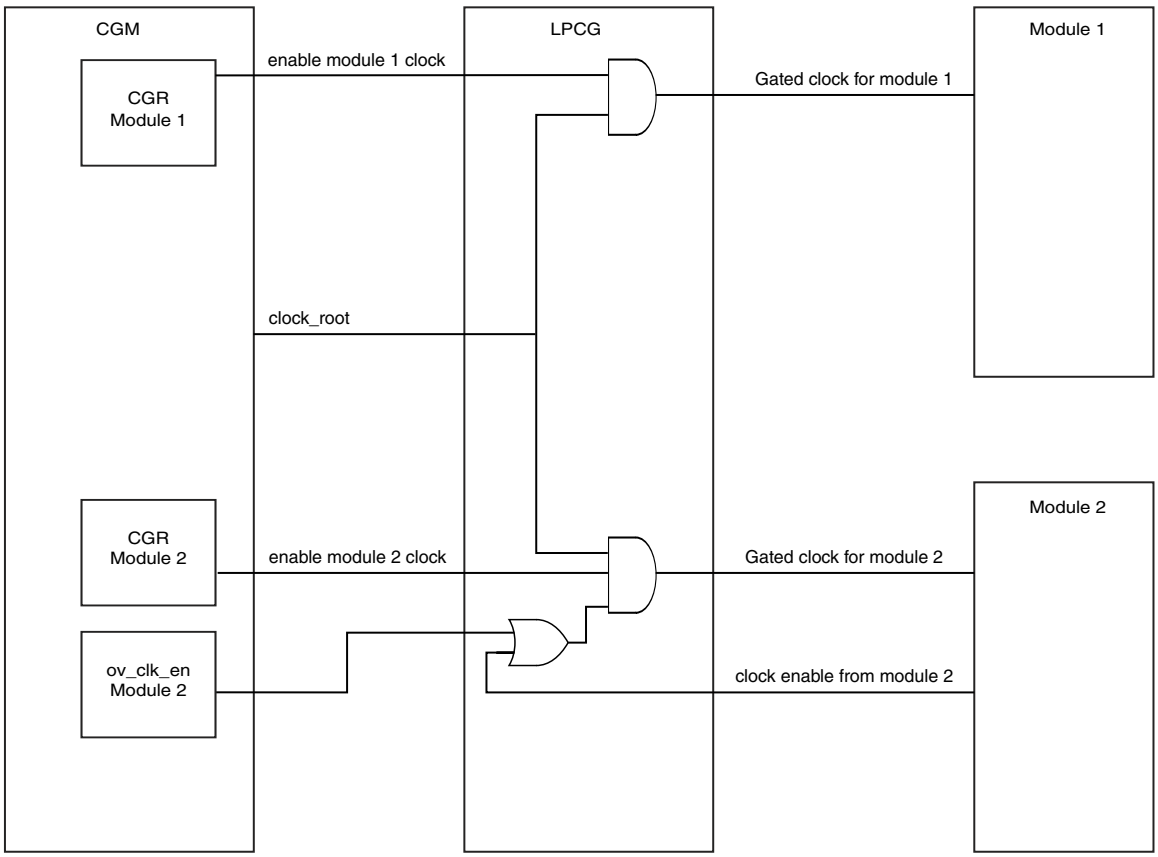


Figure 14-6. Clock split in LPCG

14.6.2 DVFS support

When performing DVFS, the frequency shift procedure for the Arm core clock domain can be performed by software.

CPU PLL frequency and CCM ARM clock divider is controlled by CCM and CPU power domain supply voltage value is controlled by CCM_ANALOG module.

NOTE

The frequency should be shifted down first and then voltage value reduced, and vice-versa, when shifting the frequency up.

NOTE

CCM_ANALOG will not control the voltage value in Bypass mode

14.6.3 Power modes

The chip supports 3 low power modes: RUN mode, WAIT mode, STOP mode.

14.6.3.1 RUN mode

This is the normal/functional operating mode. In this mode, the CPU runs in its normal operational mode. Clocks to the modules can be gated by configuring the corresponding CCGRx bits.

14.6.3.2 WAIT mode

In this mode the CPU clock is gated. All other clocks are functional and can be gated by programming their CGR bits when all Arm cores are in WFI, and L2 cache and SCU are idle.

14.6.3.2.1 Entering WAIT mode

If the CLPCR[LPM] bit is set by software to WAIT mode, when CPU executes the next wait for interrupt (WFI) instruction, WAIT mode sequence will start.

As part of the WFI routine, alternative interrupt controller in GPC should be updated; the CPU platform interrupt controller will be disabled first by software and will be not functional, due to clock gating. Interrupts during WAIT mode are monitored by alternative interrupt controller.

After execution of the WFI routine, the CPU platform will assert idle signals for each component of the platform and CCM will gate clock to the platform.

The next actions can be programmed during WAIT mode:

1. CCM request will be issued if the handshake is not bypassed by programing the CLPCR register. If the corresponding bits are set, the request signal will not be issued to the corresponding module and CCM will not wait for its acknowledge in the process of entering low power mode. After CCM receives all the acknowledge signals needed, then it will enter WAIT mode.
2. Close the clocks to the modules which were defined to be shut at WAIT mode in the CCGR bits.
3. Observability to indicate WAIT mode.

Any enabled interrupt assertion will start the exit from WAIT mode.

14.6.3.2.2 Exiting WAIT mode

As soon as enabled interrupt is asserted, CPU supply will be restored if CPU SRPG was applied and clocks are enabled to CPU and other modules.

14.6.3.3 STOP mode

In this mode all system clocks are stopped, along with the CPU, system buses and all PLLs. Power gating can be applied for Arm platform. External supply voltage can be reduced to decrease leakage.

14.6.3.3.1 Entering STOP mode

Procedure entering STOP mode is the same, as entering WAIT mode until the moment of disabling clocks to modules. (LPM bit should be configured to STOP mode.)

After clocks to modules are gated, the following actions will be taken:

- PLLs are disabled
- CCM_PMIC_STBY_REQ asserted, if vstby bit is set
- osc_en signal is negated
- osc_pwrdn is asserted, if sbyos bit is set

Counter will be triggered after CCM_PMIC_STBY_REQ assertion to allow to external regulator or PMIC to decrease voltage until valid voltage range. On counter completion, stop_mode signal will be asserted, that will trigger disabling analog elements in anatop.

CCM's low power state machine will remain in state STOP_GPC until STOP mode is exited.

14.6.3.3.2 Exiting STOP mode

As soon as an enabled interrupt is asserted, the CCM will begin the process of exiting STOP mode.

The following will take place:

1. If vstby bit was set, deassert PMIC_STBY_REQ to notify power management IC to change voltage from standby voltage to functional voltage.
2. If sbyos was set, and CCM closed either external oscillator or on board oscillator, then CCM will start oscillator by asserting ref_en_b signal and deasserting cosc_pwrdown signal respectively.

3. After the number of cycles of CKILs defined in stby_count bits, wait until PMIC functional voltage is ready. This is the notification from power management IC that the voltage is ready at its functional value. Only then will CCM continue the steps.
4. Start osc. If oscillator was started, wait until oscnt has finished its counting to make sure that oscillator is ready.
5. Start PLLs. Only the PLLs that were configured to be on prior to the entrance to STOP mode will be started.
6. CCM will request GPC to restore Arm power by GPC_PUP_REQ. If power was removed from the Arm platform, GPC will notify CCM by asserting signal GPC_PUP_ACK that power to Arm is back on, and its safe to exit from STOP mode. Only then will the CCM progress to the next step.
7. After assertion of notification from src that the resets for the power gated modules has been finished, (src_power_gating_reset_done is set) negate the low power request signals to all modules and enable all module clocks including Arm clocks and CKIL sync, and return to run mode. (Clocks whose CCGR bits are not to be opened in RUN mode will not be opened; they will continued to be gated.)

After the system is in run mode, the ccm_ipg_stop and system_in_stop_mode signals negate.

14.7 CCM Memory Map/Register Definition

NOTE

The register reset values for CCM change depending on the boot configuration. See Clocks at boot time for more information.

CCM memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400F_C000	CCM Control Register (CCM_CCR)	32	R/W	0401_107Fh	14.7.1/1040
400F_C008	CCM Status Register (CCM_CSR)	32	R	0000_0010h	14.7.2/1042
400F_C00C	CCM Clock Switcher Register (CCM_CCSR)	32	R/W	0000_0100h	14.7.3/1044
400F_C010	CCM Arm Clock Root Register (CCM_CACRR)	32	R/W	0000_0001h	14.7.4/1045
400F_C014	CCM Bus Clock Divider Register (CCM_CBCDR)	32	R/W	000A_8300h	14.7.5/1046
400F_C018	CCM Bus Clock Multiplexer Register (CCM_CBCMR)	32	R/W	2DAE_8324h	14.7.6/1048
400F_C01C	CCM Serial Clock Multiplexer Register 1 (CCM_CSCMR1)	32	R/W	0490_0000h	14.7.7/1051
400F_C020	CCM Serial Clock Multiplexer Register 2 (CCM_CSCMR2)	32	R/W	1319_2F06h	14.7.8/1053

Table continues on the next page...

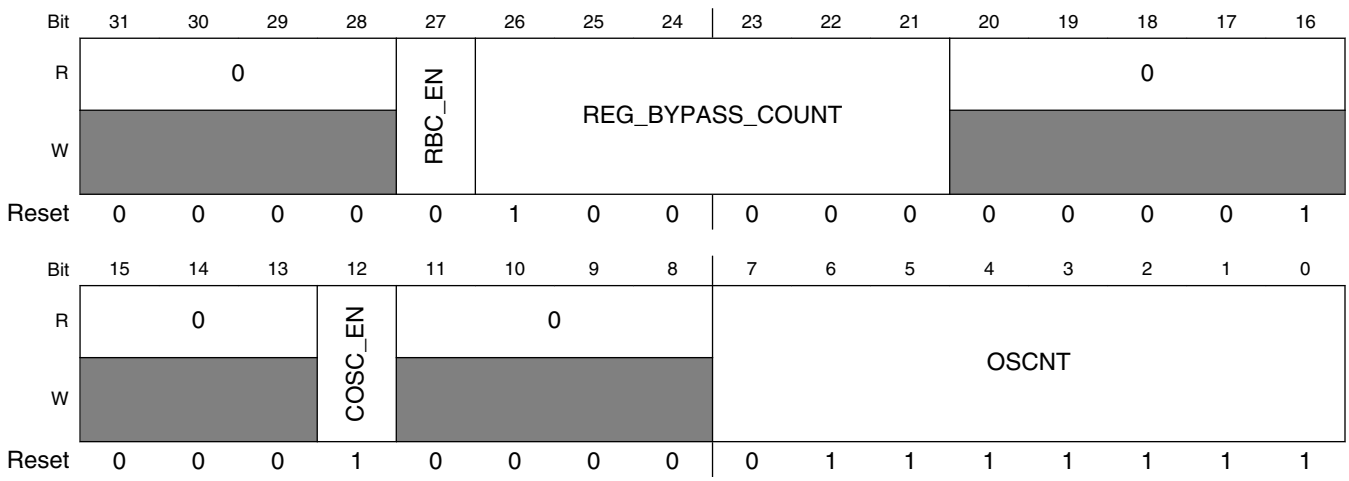
CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400F_C024	CCM Serial Clock Divider Register 1 (CCM_CSCDR1)	32	R/W	0649_0B00h	14.7.9/1054
400F_C028	CCM Clock Divider Register (CCM_CS1CDR)	32	R/W	0EC1_02C1h	14.7.10/ 1056
400F_C02C	CCM Clock Divider Register (CCM_CS2CDR)	32	R/W	0073_36C1h	14.7.11/ 1058
400F_C030	CCM D1 Clock Divider Register (CCM_CDCDR)	32	R/W	33F7_1F92h	14.7.12/ 1059
400F_C038	CCM Serial Clock Divider Register 2 (CCM_CSCDR2)	32	R/W	0002_9150h	14.7.13/ 1061
400F_C03C	CCM Serial Clock Divider Register 3 (CCM_CSCDR3)	32	R/W	0003_0841h	14.7.14/ 1062
400F_C048	CCM Divider Handshake In-Process Register (CCM_CDHIPR)	32	R	0000_0000h	14.7.15/ 1064
400F_C054	CCM Low Power Control Register (CCM_CLPCR)	32	R/W	0000_0079h	14.7.16/ 1067
400F_C058	CCM Interrupt Status Register (CCM_CISR)	32	w1c	0000_0000h	14.7.17/ 1069
400F_C05C	CCM Interrupt Mask Register (CCM_CIMR)	32	R/W	FFFF_FFFFh	14.7.18/ 1072
400F_C060	CCM Clock Output Source Register (CCM_CCOSR)	32	R/W	000A_0001h	14.7.19/ 1074
400F_C064	CCM General Purpose Register (CCM_CGPR)	32	R/W	0000_FE62h	14.7.20/ 1076
400F_C068	CCM Clock Gating Register 0 (CCM_CCGR0)	32	R/W	FFFF_FFFFh	14.7.21/ 1077
400F_C06C	CCM Clock Gating Register 1 (CCM_CCGR1)	32	R/W	FFFF_FFFFh	14.7.22/ 1079
400F_C070	CCM Clock Gating Register 2 (CCM_CCGR2)	32	R/W	FC3F_FFFFh	14.7.23/ 1080
400F_C074	CCM Clock Gating Register 3 (CCM_CCGR3)	32	R/W	FFFF_FFCFh	14.7.24/ 1081
400F_C078	CCM Clock Gating Register 4 (CCM_CCGR4)	32	R/W	FFFF_FFFFh	14.7.25/ 1083
400F_C07C	CCM Clock Gating Register 5 (CCM_CCGR5)	32	R/W	FFFF_FFFFh	14.7.26/ 1084
400F_C080	CCM Clock Gating Register 6 (CCM_CCGR6)	32	R/W	FFFF_FFFFh	14.7.27/ 1085
400F_C084	CCM Clock Gating Register 7 (CCM_CCGR7)	32	R/W	FFFF_FFFFh	14.7.28/ 1087
400F_C088	CCM Module Enable Overide Register (CCM_CMEOR)	32	R/W	FFFF_FFFFh	14.7.29/ 1088

14.7.1 CCM Control Register (CCM_CCR)

The figure below represents the CCM Control Register (CCR), which contains bits to control general operation of CCM. The table below provides its field descriptions.

Address: 400F_C000h base + 0h offset = 400F_C000h



CCM_CCR field descriptions

Field	Description
31–28 Reserved	This read-only field is reserved and always has the value 0.
27 RBC_EN	Enable for REG_BYPASS_COUNTER. If enabled, analog_reg_bypass signal will be asserted after REG_BYPASS_COUNT clocks of CKIL, after standby voltage is requested. If standby voltage is not requested analog_reg_bypass won't be asserted, event if counter is enabled. It is recommended to configure REG_BYPASS_COUNTER as late as possible in the suspend entry sequence. 1 REG_BYPASS_COUNTER enabled. 0 REG_BYPASS_COUNTER disabled
26–21 REG_BYPASS_ COUNT	Counter for analog_reg_bypass signal assertion after standby voltage request by PMIC_STBY_REQ. Should be zeroed and reconfigured after exit from low power mode. REG_BYPASS_COUNT can also be used for holding off interrupts when the PGC unit is sending signals to power gate the core. 000000 no delay 000001 1 CKIL clock period delay 111111 63 CKIL clock periods delay
20–13 Reserved	This read-only field is reserved and always has the value 0.
12 COSC_EN	On chip oscillator enable bit - this bit value is reflected on the output cosc_en. The system will start with on chip oscillator enabled to supply source for the PLLs. Software can change this bit if a transition to the bypass PLL clocks was performed for all the PLLs. In cases that this bit is changed from '0' to '1' then CCM will enable the on chip oscillator and after counting oscnt ckil clock cycles it will notify that on chip

Table continues on the next page...

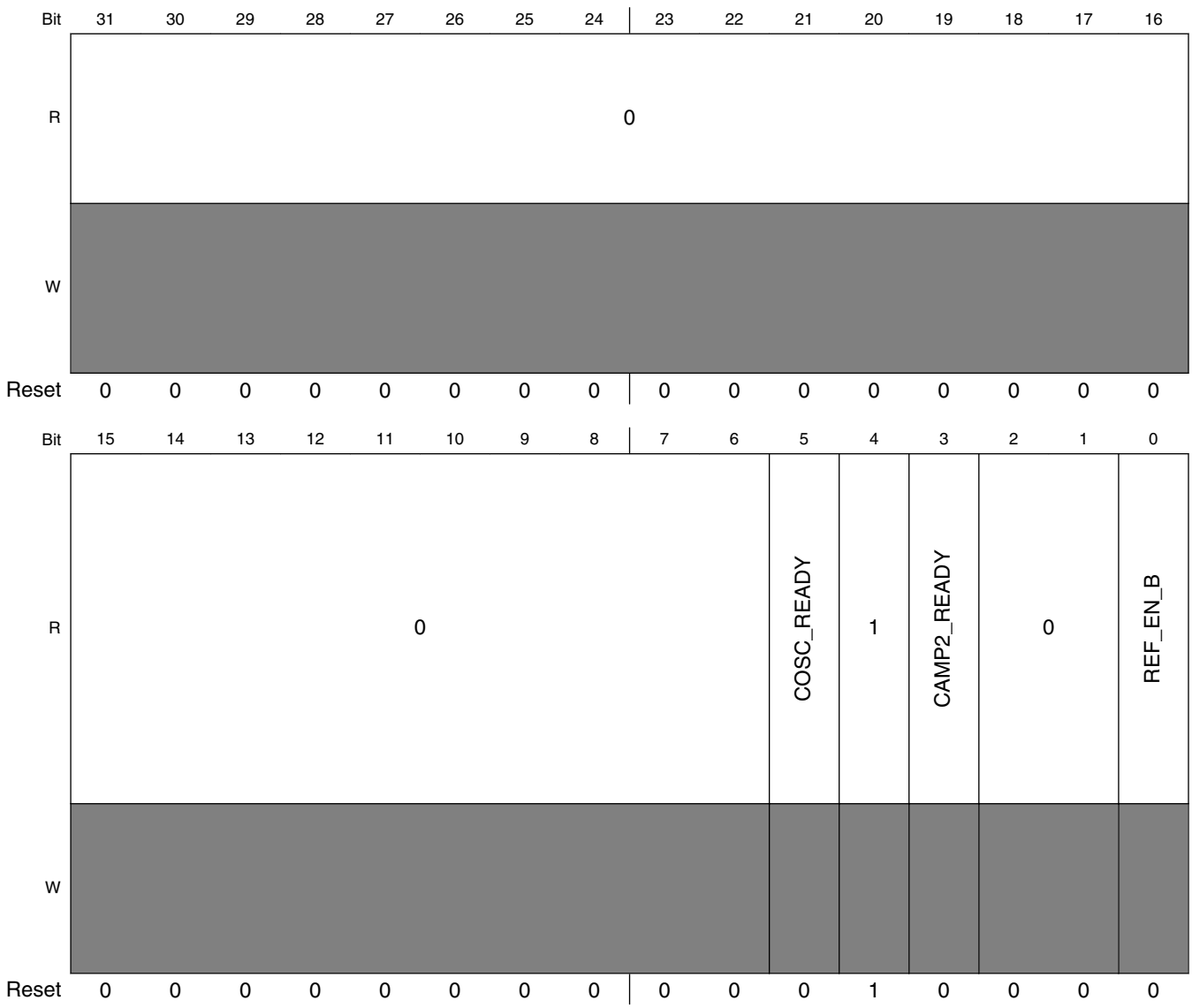
CCM_CCR field descriptions (continued)

Field	Description
	oscillator is ready by an interrupt cosc_ready and by status bit cosc_ready. The cosc_en bit should be changed only when on chip oscillator is not chosen as the clock source. 0 disable on chip oscillator 1 enable on chip oscillator
11–8 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
OSCNT	Oscillator ready counter value. These bits define value of 32KHz counter, that serve as counter for oscillator lock time. This is used for oscillator lock time. Current estimation is ~5ms. This counter will be used in ignition sequence and in wake from stop sequence if sbyos bit was defined, to notify that on chip oscillator output is ready for the dpll_ip to use and only then the gate in dpll_ip can be opened. 00000000 count 1 ckil 11111111 count 256 ckil's

14.7.2 CCM Status Register (CCM_CSR)

The figure below represents the CCM status Register (CSR). The status bits are read-only bits. The table below provides its field descriptions.

Address: 400F_C000h base + 8h offset = 400F_C008h



CCM_CSR field descriptions

Field	Description
31–6 Reserved	This read-only field is reserved and always has the value 0.

Table continues on the next page...

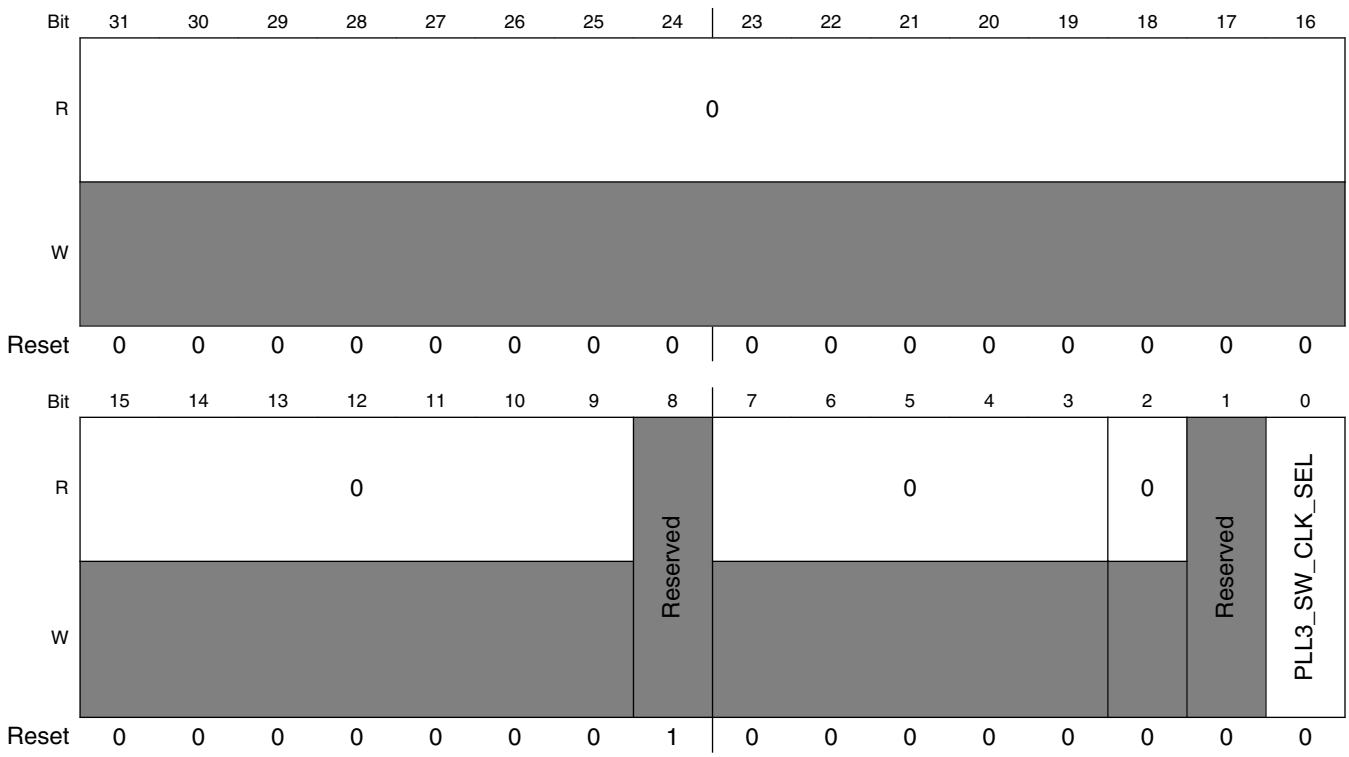
CCM_CSR field descriptions (continued)

Field	Description
5 COSC_READY	Status indication of on board oscillator. This bit will be asserted if on chip oscillator is enabled and on chip oscillator is not powered down, and if oscnt counter has finished counting. 0 on board oscillator is not ready. 1 on board oscillator is ready.
4 Reserved	This read-only field is reserved and always has the value 1.
3 CAMP2_READY	Status indication of CAMP2. 0 CAMP2 is not ready. 1 CAMP2 is ready.
2–1 Reserved	This read-only field is reserved and always has the value 0.
0 REF_EN_B	Status of the value of CCM_REF_EN_B output of ccm 0 value of CCM_REF_EN_B is '0' 1 value of CCM_REF_EN_B is '1'

14.7.3 CCM Clock Switcher Register (CCM_CCSR)

The figure below represents the CCM Clock Switcher register (CCSR). The CCSR register contains bits to control the switcher sub-module dividers and multiplexers. The table below provides its field descriptions.

Address: 400F_C000h base + Ch offset = 400F_C00Ch



CCM_CCSR field descriptions

Field	Description
31–9 Reserved	This read-only field is reserved and always has the value 0.
8 -	This field is reserved. Reserved
7–3 Reserved	This read-only field is reserved and always has the value 0.
2 Reserved	This read-only field is reserved and always has the value 0.
1 -	This field is reserved. Reserved

Table continues on the next page...

CCM_CCSR field descriptions (continued)

Field	Description
0 PLL3_SW_CLK_SEL	Selects source to generate pll3_sw_clk. This bit should only be used for testing purposes. 0 pll3_main_clk 1 pll3 bypass clock

14.7.4 CCM Arm Clock Root Register (CCM_CACRR)

The figure below represents the CCM Arm Clock Root register (CACRR). The CACRR register contains bits to control the Arm clock root generation. The table below provides its field descriptions.

Address: 400F_C000h base + 10h offset = 400F_C010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																													ARM_PODF		
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

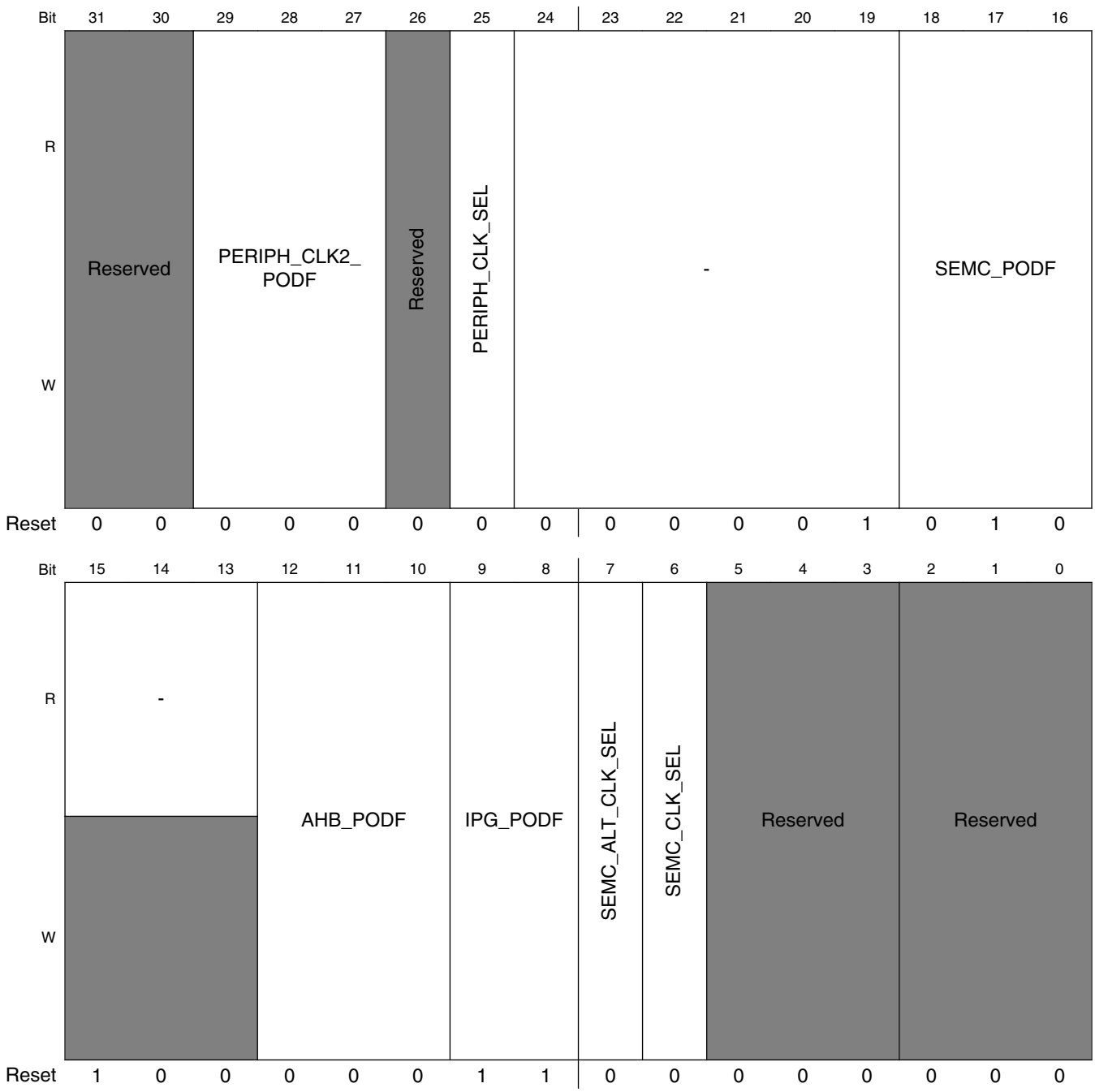
CCM_CACRR field descriptions

Field	Description
31–3 Reserved	This read-only field is reserved and always has the value 0.
ARM_PODF	Divider for Arm clock root. **NOTE:** If arm_freq_shift_divider is set to '1' then any new write to arm_podf will be held until arm_clk_switch_req signal is asserted. 000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8

14.7.5 CCM Bus Clock Divider Register (CCM_CBCDR)

The figure below represents the CCM Bus Clock Divider Register (CBCDR). The CBCDR register contains bits to control the clock generation sub module dividers. The table below provides its field descriptions.

Address: 400F_C000h base + 14h offset = 400F_C014h



CCM_CBCDR field descriptions

Field	Description
31–30 -	This field is reserved. Reserved
29–27 PERIPH_CLK2_PODF	Divider for periph_clk2_podf. 000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
26 -	This field is reserved. Reserved
25 PERIPH_CLK_SEL	Selector for peripheral main clock. **NOTE:** Alternative clock source should be used when PLL is relocked. For PLL relock procedure pls refer to the PLL chapter. **NOTE:** Any change of this sync mux select will involve handshake with the MMDC. Refer to the CCDR and CDHIPR registers for the handshake bypass and busy bits. 0 derive clock from pre_periph_clk_sel 1 derive clock from periph_clk2_clk_divided
24–19 -	Reserved This is a non-zero reserved field and the reset value of this bit field should be maintained.
18–16 SEMC_PODF	Post divider for SEMC clock. **NOTE:** Any change of this divider might involve handshake with EMI. See CDHIPR register for the handshake busy bits. 000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
15–13 -	Reserved
12–10 AHB_PODF	Divider for AHB PODF. **NOTE:** Any change of this divider might involve handshake with EMI. See CDHIPR register for the handshake busy bits. 000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4

Table continues on the next page...

CCM_CBCDR field descriptions (continued)

Field	Description
	100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
9–8 IPG_PODF	Divider for ipg podf. 00 divide by 1 01 divide by 2 10 divide by 3 11 divide by 4
7 SEMC_ALT_ CLK_SEL	SEMC alternative clock select 0 PLL2 PFD2 will be selected as alternative clock for SEMC root clock 396 MHz 1 PLL3 PFD1 will be selected as alternative clock for SEMC root clock 664.62 MHz
6 SEMC_CLK_SEL	SEMC clock source select 0 Periph_clk output will be used as SEMC clock root 1 SEMC alternative clock will be used as SEMC clock root
5–3 -	This field is reserved. Reserved
-	This field is reserved. Reserved

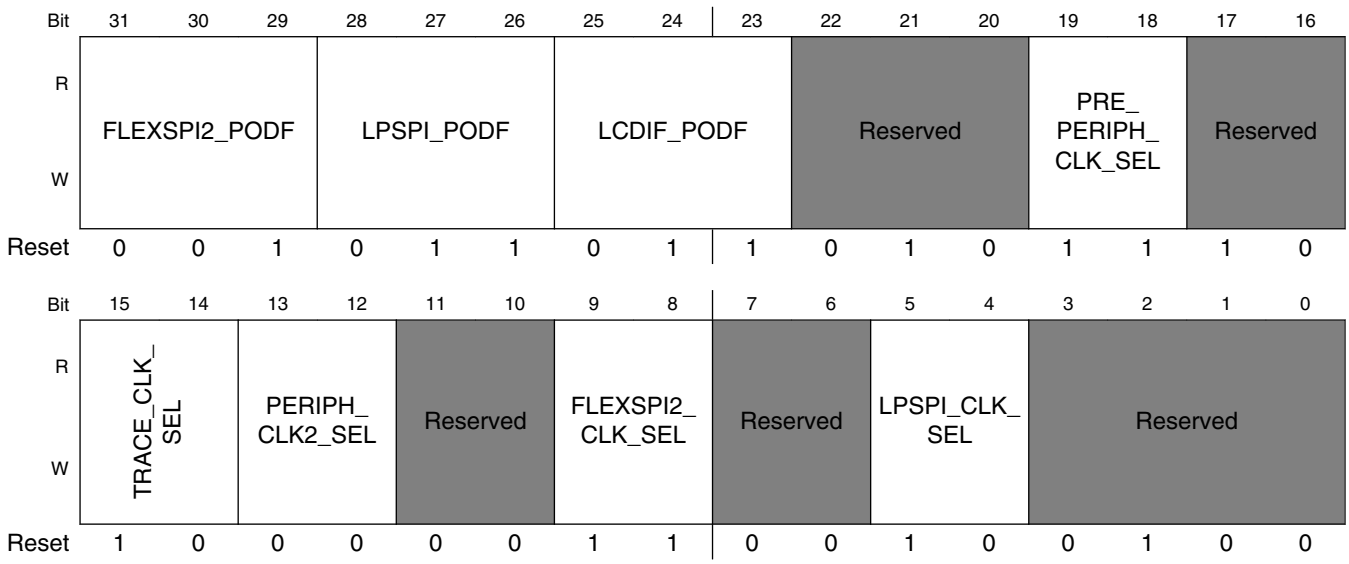
14.7.6 CCM Bus Clock Multiplexer Register (CCM_CBCMR)

The figure below represents the CCM Bus Clock Multiplexer Register (CBCMR). The CBCMR register contains bits to control the multiplexers that generate the bus clocks. The table below provides its field descriptions.

NOTE

Any change on the above multiplexer will have to be done while the module that its clock is affected is not functional and the respective clock is gated in LPCG. If the change will be done during operation of the module, then it is not guaranteed that the modules operation will not be harmed.

Address: 400F_C000h base + 18h offset = 400F_C018h



CCM_CBCMR field descriptions

Field	Description
31–29 FLEXSPI2_ PODF	Divider for flexspi2 clock root. 000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
28–26 LPSPI_PODF	Divider for LPSPI. **NOTE:** Divider should be updated when output clock is gated. 000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
25–23 LCDIF_PODF	Post-divider for LCDIF clock. 000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6

Table continues on the next page...

CCM_CBCMR field descriptions (continued)

Field	Description
	110 divide by 7 111 divide by 8
22–20 -	This field is reserved. Reserved
19–18 PRE_PERIPH_CLK_SEL	Selector for pre_periph clock multiplexer 00 derive clock from PLL2 528 MHz 01 derive clock from PLL2 PFD2 396 MHz 10 derive clock from PLL2 PFD0 352 MHz 11 derive clock from divided PLL1 CPU / CCM_CACRR
17–16 -	This field is reserved. Reserved
15–14 TRACE_CLK_SEL	Selector for Trace clock multiplexer 00 derive clock from PLL2 528 MHz 01 derive clock from PLL2 PFD2 396 MHz 10 derive clock from PLL2 PFD0 352 MHz 11 derive clock from PLL2 PFD1 594 MHz
13–12 PERIPH_CLK2_SEL	Selector for peripheral clk2 clock multiplexer 00 derive clock from pll3_sw_clk 01 derive clock from osc_clk (pll1_ref_clk) 10 derive clock from pll2_bypass_clk 11 reserved
11–10 -	This field is reserved. Reserved
9–8 FLEXSPI2_CLK_SEL	Selector for flexspi2 clock multiplexer 00 derive clock from PLL2 PFD2 396 MHz 01 derive clock from PLL3 PFD0 720 MHz 10 derive clock from PLL3 PFD1 664.62 MHz 11 derive clock from PLL2 (pll2_main_clk) 528 MHz
7–6 -	This field is reserved. Reserved
5–4 LPSPI_CLK_SEL	Selector for lpspi clock multiplexer 00 derive clock from PLL3 PFD1 clk 664.62 MHz 01 derive clock from PLL3 PFD0 720 MHz 10 derive clock from PLL2 528 MHz 11 derive clock from PLL2 PFD2 396 MHz
-	This field is reserved. Reserved

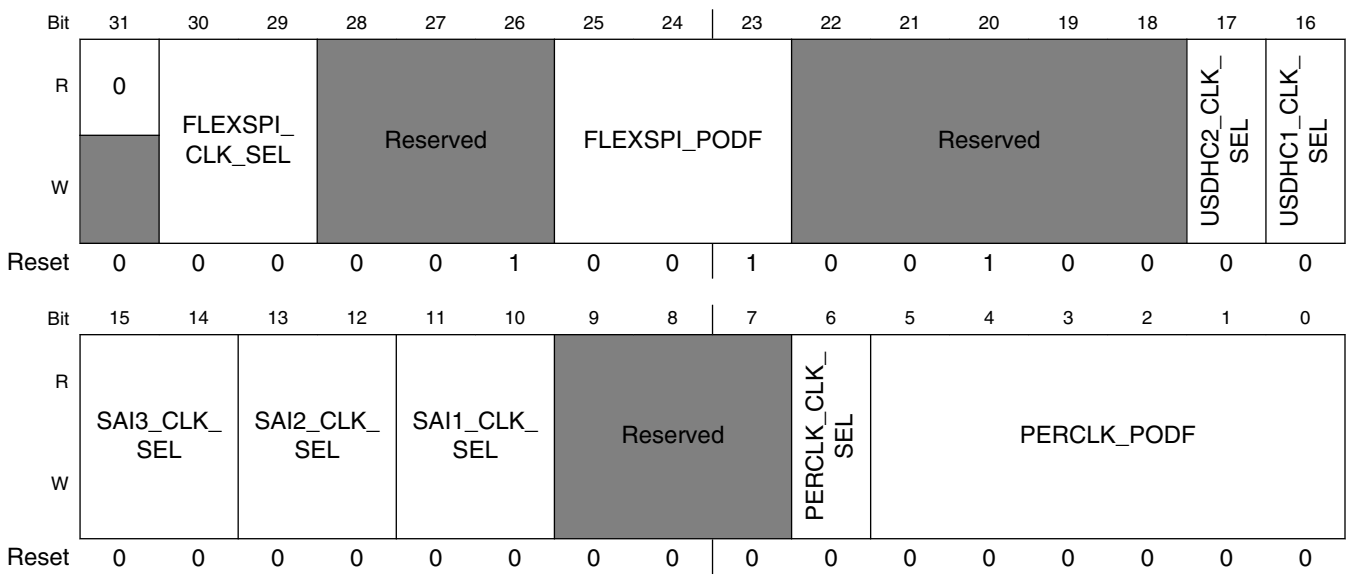
14.7.7 CCM Serial Clock Multiplexer Register 1 (CCM_CSCMR1)

The figure below represents the CCM Serial Clock Multiplexer Register 1 (CSCMR1). The CSCMR1 register contains bits to control the multiplexers that generate the serial clocks. The table below provides its field descriptions.

NOTE

Any change on the above multiplexer will have to be done while the module that its clock is affected is not functional and the clock is gated. If the change will be done during operation of the module, then it is not guaranteed that the modules operation will not be harmed.

Address: 400F_C000h base + 1Ch offset = 400F_C01Ch



CCM_CSCMR1 field descriptions

Field	Description
31 Reserved	This read-only field is reserved and always has the value 0.
30–29 FLEXSPI_CLK_SEL	Selector for flexspi clock multiplexer 00 derive clock from semc_clk_root_pre 01 derive clock from pll3_sw_clk 480 MHz (or 24 MHz) 10 derive clock from PLL2 PFD2 396 MHz 11 derive clock from PLL3 PFD0 720 MHz
28–26 -	This field is reserved. Reserved
25–23 FLEXSPI_PODF	Divider for flexspi clock root.

Table continues on the next page...

CCM_CSCMR1 field descriptions (continued)

Field	Description
	000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
22–18 -	This field is reserved. Reserved
17 USDHC2_CLK_SEL	Selector for usdhc2 clock multiplexer 0 derive clock from PLL2 PFD2 396 MHz 1 derive clock from PLL2 PFD0 352 MHz
16 USDHC1_CLK_SEL	Selector for usdhc1 clock multiplexer 0 derive clock from PLL2 PFD2 396 MHz 1 derive clock from PLL2 PFD0 352 MHz
15–14 SAI3_CLK_SEL	Selector for sai3/adc1/adc2 clock multiplexer 00 derive clock from PLL3 PFD2 508.24 MHz 01 derive clock from PLL5 10 derive clock from PLL4 11 Reserved
13–12 SAI2_CLK_SEL	Selector for sai2 clock multiplexer 00 derive clock from PLL3 PFD2 508.24 MHz 01 derive clock from PLL5 10 derive clock from PLL4 11 Reserved
11–10 SAI1_CLK_SEL	Selector for sai1 clock multiplexer 00 derive clock from PLL3 PFD2 508.24 MHz 01 derive clock from PLL5 10 derive clock from PLL4 11 Reserved
9–7 -	This field is reserved. Reserved
6 PERCLK_CLK_SEL	Selector for the perclk clock multiplexor 0 derive clock from ipg clk root 1 derive clock from osc_clk
PERCLK_PODF	Divider for perclk podf. 000000 divide by 1 000001 divide by 2 000010 divide by 3 000011 divide by 4

Table continues on the next page...

CCM_CSCMR1 field descriptions (continued)

Field	Description
	000100 divide by 5 000101 divide by 6 000110 divide by 7 111111 divide by 64

14.7.8 CCM Serial Clock Multiplexer Register 2 (CCM_CSCMR2)

The figure below represents the CCM Serial Clock Multiplexer Register 2 (CSCMR2). The CSCMR2 register contains bits to control the multiplexers that generate the serial clocks. The table below provides its field descriptions.

NOTE

Any change on the above multiplexer will have to be done while the module that its clock is affected is not functional and the clock is gated. If the change will be done during operation of the module, then it is not guaranteed that the modules operation will not be harmed.

Address: 400F_C000h base + 20h offset = 400F_C020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved											FLEXIO2_CLK_SEL		Reserved		
Reset	0	0	0	1	0	0	1	1	0	0	0	1	1	0	0	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved						CAN_CLK_SEL		CAN_CLK_PODF						Reserved	
Reset	0	0	1	0	1	1	1	1	0	0	0	0	0	1	1	0

CCM_CSCMR2 field descriptions

Field	Description
31–21 -	This field is reserved. Reserved
20–19 FLEXIO2_CLK_SEL	Selector for flexio2/flexio3 clock multiplexer 00 derive clock from PLL4 divided clock 01 derive clock from PLL3 PFD2 clock 508.24 MHz 10 derive clock from PLL5 clock 11 derive clock from pll3_sw_clk 480 MHz (or 24 MHz)
18–10 -	This field is reserved. Reserved

Table continues on the next page...

CCM_CSCMR2 field descriptions (continued)

Field	Description
9–8 CAN_CLK_SEL	Selector for CAN/CANFD clock multiplexer 00 derive clock from pll3_sw_clk divided clock (60M) 01 derive clock from osc_clk (24M) 10 derive clock from pll3_sw_clk divided clock (80M) 11 Disable FlexCAN clock
7–2 CAN_CLK_ PODF	Divider for CAN/CANFD clock podf. 000000 divide by 1 ... 000111 divide by 8 ... 111111 divide by 2^6
-	This field is reserved. Reserved

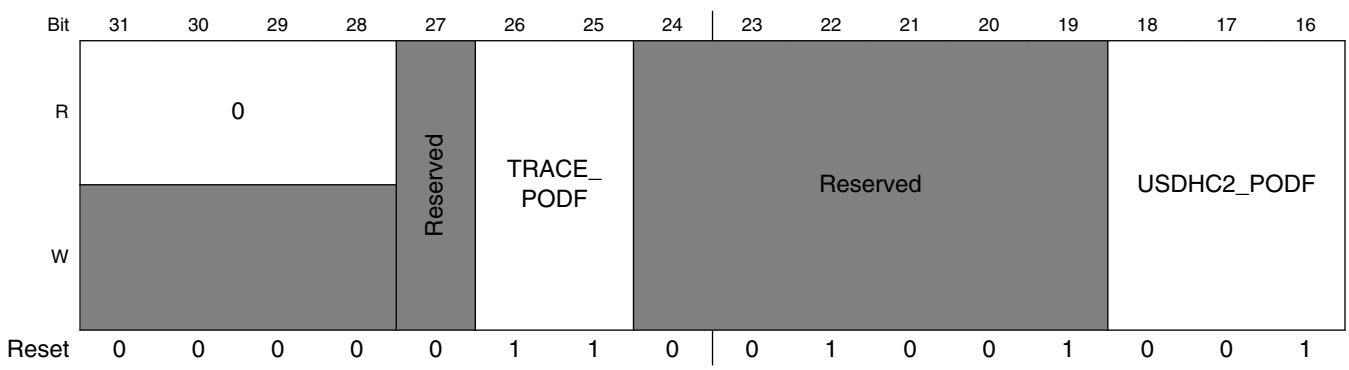
14.7.9 CCM Serial Clock Divider Register 1 (CCM_CSCDR1)

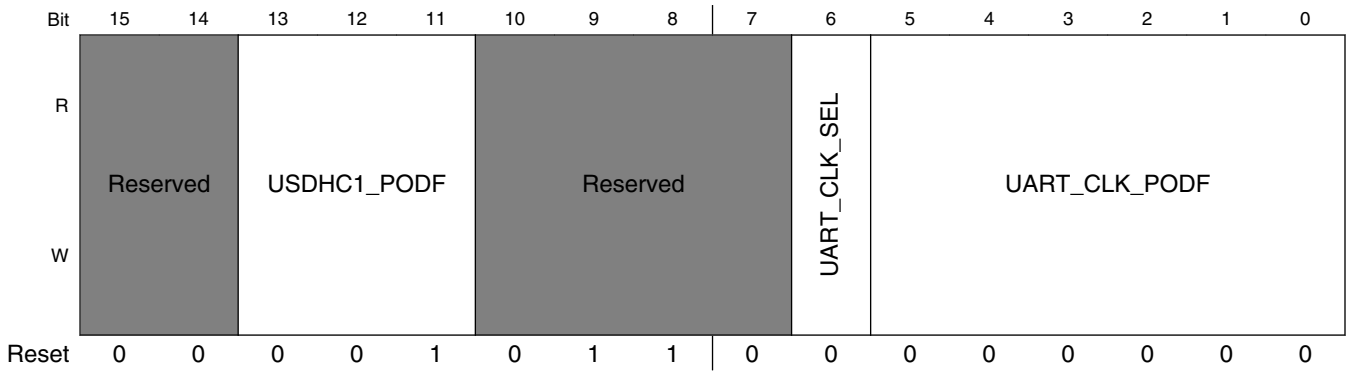
The figure below represents the CCM Serial Clock Divider Register 1 (CSCDR1). The CSCDR1 register contains bits to control the clock generation sub-module dividers. The table below provides its field descriptions.

NOTE

Any change on the above dividers will have to be done while the module that its clock is affected is not functional and the affected clock is gated. If the change will be done during operation of the module, then it is not guaranteed that the modules operation will not be harmed.

Address: 400F_C000h base + 24h offset = 400F_C024h





CCM_CSCDR1 field descriptions

Field	Description
31–28 Reserved	This read-only field is reserved and always has the value 0.
27 -	This field is reserved. Reserved
26–25 TRACE_PODF	Divider for trace clock. **NOTE:** Divider should be updated when output clock is gated. 00 divide by 1 01 divide by 2 10 divide by 3 11 divide by 4
24–19 -	This field is reserved. Reserved
18–16 USDHC2_PODF	Divider for usdhc2 clock. **NOTE:** Divider should be updated when output clock is gated. 000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
15–14 -	This field is reserved. Reserved
13–11 USDHC1_PODF	Divider for usdhc1 clock podf. **NOTE:** Divider should be updated when output clock is gated. 000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5

Table continues on the next page...

CCM_CSCDR1 field descriptions (continued)

Field	Description
	101 divide by 6 110 divide by 7 111 divide by 8
10–7 -	This field is reserved. Reserved.
6 UART_CLK_SEL	Selector for the UART clock multiplexor 0 derive clock from pll3_80m 80 MHz 1 derive clock from osc_clk 24 MHz
UART_CLK_ PODF	Divider for uart clock podf. 000000 divide by 1 111111 divide by 2^6

14.7.10 CCM Clock Divider Register (CCM_CS1CDR)

The figure below represents the CCM SAI1, and SAI3 Clock Divider Register (CS1CDR). The CS1CDR register contains bits to control the SAI1 and SAI3 clock generation dividers. The table below provides its field descriptions.

Address: 400F_C000h base + 28h offset = 400F_C028h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0				FLEXIO2_CLK_PODF			SAI3_CLK_PRED			SAI3_CLK_PODF						0				FLEXIO2_CLK_PRED			SAI1_CLK_PRED			SAI1_CLK_PODF					
W																																
Reset	0	0	0	0	1	1	1	0	1	1	0	0	0	0	0	1	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	1

CCM_CS1CDR field descriptions

Field	Description
31–28 Reserved	This read-only field is reserved and always has the value 0.
27–25 FLEXIO2_CLK_ PODF	Divider for flexio2/flexio3 clock. 000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8

Table continues on the next page...

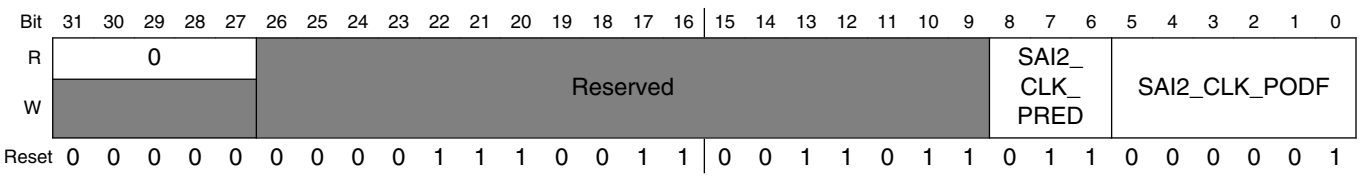
CCM_CS1CDR field descriptions (continued)

Field	Description
24–22 SAI3_CLK_ PRED	Divider for sai3/adc1/adc2 clock pred. 000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
21–16 SAI3_CLK_ PODF	Divider for sai3/adc1/adc2 clock podf. The input clock to this divider should be lower than 300Mhz, the predivider can be used to achieve this. 000000 divide by 1 111111 divide by 2^6
15–12 Reserved	This read-only field is reserved and always has the value 0.
11–9 FLEXIO2_CLK_ PRED	Divider for flexio2/flexio3 clock. 000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
8–6 SAI1_CLK_ PRED	Divider for sai1 clock pred. 000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
SAI1_CLK_ PODF	Divider for sai1 clock podf. The input clock to this divider should be lower than 300Mhz, the predivider can be used to achieve this. 000000 divide by 1 111111 divide by 2^6

14.7.11 CCM Clock Divider Register (CCM_CS2CDR)

The figure below represents the CCM SAI2 Clock Divider Register (CS2CDR). The CS2CDR register contains bits to control the SAI2 clock generation dividers. The table below provides its field descriptions.

Address: 400F_C000h base + 2Ch offset = 400F_C02Ch



CCM_CS2CDR field descriptions

Field	Description
31–27 Reserved	This read-only field is reserved and always has the value 0.
26–9 -	This field is reserved. Reserved
8–6 SAI2_CLK_ PRED	Divider for sai2 clock pred. **NOTE:** Divider should be updated when output clock is gated. 000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
SAI2_CLK_ PODF	Divider for sai2 clock podf. The input clock to this divider should be lower than 300Mhz, the predivider can be used to achieve this. **NOTE:** Divider should be updated when output clock is gated. 000000 divide by 1 111111 divide by 2^6

14.7.12 CCM D1 Clock Divider Register (CCM_CDCDR)

The figure below represents the CCM DI Clock Divider Register (CDCDR). The table below provides its field descriptions.

Address: 400F_C000h base + 30h offset = 400F_C030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved				SPDIF0_CLK_PRED			SPDIF0_CLK_PODF			SPDIF0_ CLK_SEL		Reserved			
Reset	0	0	1	1	0	0	1	1	1	1	1	1	0	1	1	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserv ed	FLEXIO1_CLK_ PRED			FLEXIO1_CLK_ PODF			FLEXIO1_ CLK_SEL	Reserved							
Reset	0	0	0	1	1	1	1	1	1	0	0	1	0	0	1	0

CCM_CDCDR field descriptions

Field	Description
31–28 -	This field is reserved. Reserved 0 derive from pll3_120M clock 1 derive clock from PLL2 PFD2 396 MHz
27–25 SPDIF0_CLK_ PRED	Divider for spdif0 clock pred. **NOTE:** Divider should be updated when output clock is gated. 000 divide by 1 (do not use with high input frequencies) 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
24–22 SPDIF0_CLK_ PODF	Divider for spdif0 clock podf. **NOTE:** Divider should be updated when output clock is gated. 000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6

Table continues on the next page...

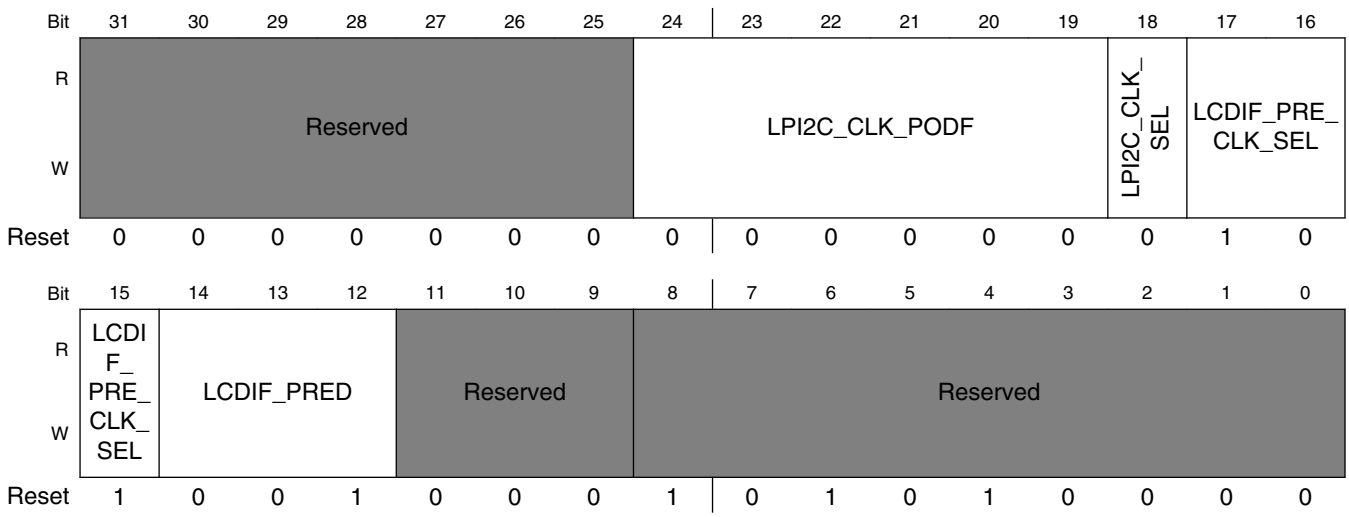
CCM_CDCDR field descriptions (continued)

Field	Description
	110 divide by 7 111 divide by 8
21–20 SPDIF0_CLK_SEL	Selector for spdif0 clock multiplexer 00 derive clock from PLL4 01 derive clock from PLL3 PFD2 508.24 MHz 10 derive clock from PLL5 11 derive clock from pll3_sw_clk
19–15 -	This field is reserved. Reserved
14–12 FLEXIO1_CLK_PRED	Divider for flexio1 clock pred. **NOTE:** Divider should be updated when output clock is gated. 000 divide by 1 (do not use with high input frequencies) 001 divide by 2 010 divide by 3 111 divide by 8
11–9 FLEXIO1_CLK_PODF	Divider for flexio1 clock podf. **NOTE:** Divider should be updated when output clock is gated. 000 divide by 1 111 divide by 8
8–7 FLEXIO1_CLK_SEL	Selector for flexio1 clock multiplexer 00 derive clock from PLL4 01 derive clock from PLL3 PFD2 508.24 MHz 10 derive clock from PLL5 11 derive clock from pll3_sw_clk
-	This field is reserved. Reserved

14.7.13 CCM Serial Clock Divider Register 2 (CCM_CSCDR2)

The figure below represents the CCM Serial Clock Divider Register 2(CSCDR2). The CSCDR2 register contains bits to control the clock generation sub-module dividers. The table below provides its field descriptions.

Address: 400F_C000h base + 38h offset = 400F_C038h



CCM_CSCDR2 field descriptions

Field	Description
31–25 - Reserved	This field is reserved.
24–19 LPI2C_CLK_PODF	Divider for lpi2c clock podf. **NOTE:** Divider should be updated when output clock is gated. **NOTE:** The input clock to this divider should be lower than 300Mhz, the predivider can be used to achieve this. 000000 divide by 1 111111 divide by 2^6
18 LPI2C_CLK_SEL	Selector for the LPI2C clock multiplexor 0 derive clock from pll3_60m 1 derive clock from osc_clk
17–15 LCDIF_PRE_CLK_SEL	Selector for lcdif root clock pre-multiplexer 000 derive clock from PLL2 528 MHz 001 derive clock from PLL3 PFD3 454.74 MHz 010 derive clock from PLL5 PLL5 config 011 derive clock from PLL2 PFD0 352 MHz

Table continues on the next page...

CCM_CSCDR2 field descriptions (continued)

Field	Description
	100 derive clock from PLL2 PFD1 594 MHz 101 derive clock from PLL3 PFD1 664.62 MHz 110-111 Reserved
14–12 LCDIF_PRED	Pre-divider for lcdif clock. **NOTE:** Divider should be updated when output clock is gated. 000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
11–9 -	This field is reserved. Reserved
-	This field is reserved. Reserved

14.7.14 CCM Serial Clock Divider Register 3 (CCM_CSCDR3)

The figure below represents the CCM Serial Clock Divider Register 3(CSCDR3). The CSCDR3 register contains bits to control the clock generation sub-module dividers. The table below provides its field descriptions.

Address: 400F_C000h base + 3Ch offset = 400F_C03Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0													Reserved			CSI_PODF					CSI_CLK_SEL		Reserved								
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1

CCM_CSCDR3 field descriptions

Field	Description
31–19 Reserved	This read-only field is reserved and always has the value 0.
18–14 -	This field is reserved. Reserved
13–11 CSI_PODF	Post divider for csi_mclk. **NOTE:** Divider should be updated when output clock is gated.

Table continues on the next page...

CCM_CSCDR3 field descriptions (continued)

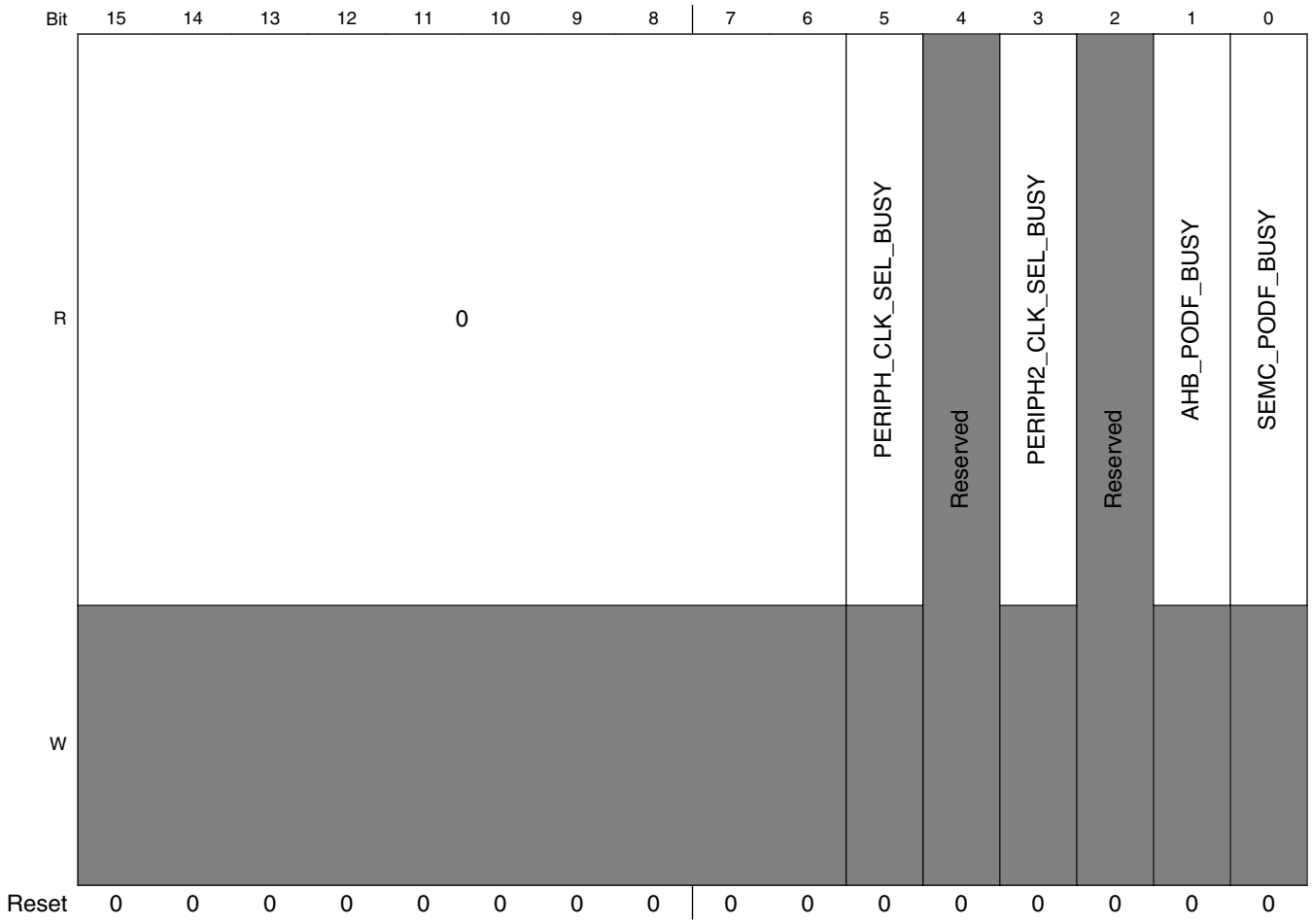
Field	Description
	000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
10–9 CSI_CLK_SEL	Selector for csi_mclk multiplexer 00 derive clock from osc_clk (24M) 01 derive clock from PLL2 PFD2 396 MHz 10 derive clock from pll3_120M 11 derive clock from PLL3 PFD1 664.64 MHz
-	This field is reserved. Reserved

14.7.15 CCM Divider Handshake In-Process Register (CCM_CDHIPR)

The figure below represents the CCM Divider Handshake In-Process Register (CDHIPR). The CDHIPR register contains read-only bits that indicate that CCM is in the process of updating dividers or muxes that might need handshake with modules.

Address: 400F_C000h base + 48h offset = 400F_C048h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															ARM_PODF_BUSY
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



CCM_CDHIPR field descriptions

Field	Description
31–17 Reserved	This read-only field is reserved and always has the value 0.
16 ARM_PODF_BUSY	Busy indicator for arm_podf. 0 divider is not busy and its value represents the actual division. 1 divider is busy with handshake process with module. The value read in the divider represents the previous value of the division factor, and after the handshake the written value of the arm_podf will be applied.
15–6 Reserved	This read-only field is reserved and always has the value 0.
5 PERIPH_CLK_SEL_BUSY	Busy indicator for periph_clk_sel mux control. 0 mux is not busy and its value represents the actual division. 1 mux is busy with handshake process with module. The value read in the periph_clk_sel represents the previous value of select, and after the handshake periph_clk_sel value will be applied.
4 -	This field is reserved. Reserved

Table continues on the next page...

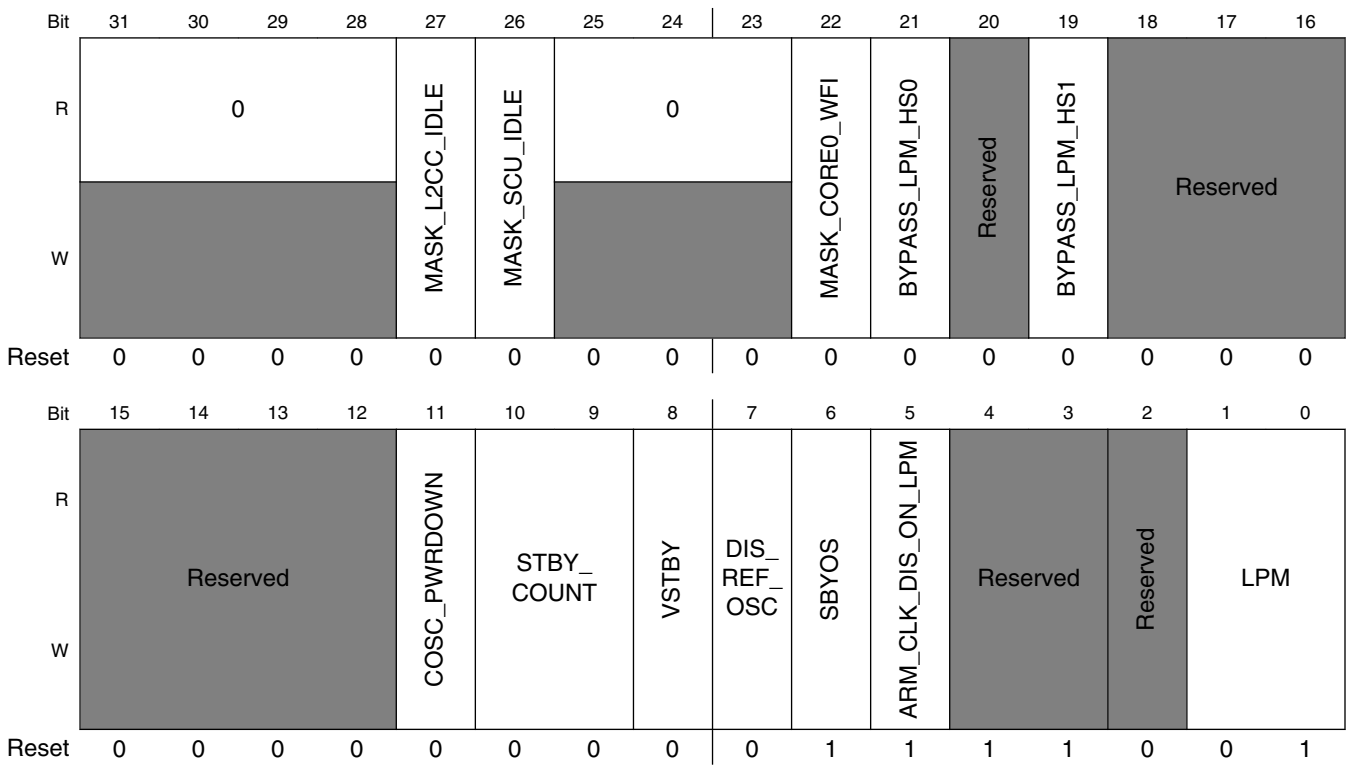
CCM_CDHIPR field descriptions (continued)

Field	Description
3 PERIPH2_CLK_SEL_BUSY	Busy indicator for periph2_clk_sel mux control. 0 mux is not busy and its value represents the actual division. 1 mux is busy with handshake process with module. The value read in the periph2_clk_sel represents the previous value of select, and after the handshake periph2_clk_sel value will be applied.
2 -	This field is reserved. Reserved
1 AHB_PODF_BUSY	Busy indicator for ahb_podf. 0 divider is not busy and its value represents the actual division. 1 divider is busy with handshake process with module. The value read in the divider represents the previous value of the division factor, and after the handshake the written value of the ahb_podf will be applied.
0 SEMC_PODF_BUSY	Busy indicator for semc_podf. 0 divider is not busy and its value represents the actual division. 1 divider is busy with handshake process with module. The value read in the divider represents the previous value of the division factor, and after the handshake the written value of the semc_podf will be applied.

14.7.16 CCM Low Power Control Register (CCM_CLPCR)

The figure below represents the CCM Low Power Control Register (CLPCR). The CLPCR register contains bits to control the low power modes operation.The table below provides its field descriptions.

Address: 400F_C000h base + 54h offset = 400F_C054h



CCM_CLPCR field descriptions

Field	Description
31–28 Reserved	This read-only field is reserved and always has the value 0.
27 MASK_L2CC_ IDLE	Mask L2CC IDLE for entering low power mode. **NOTE:** Assertion of all bits[27:22] will generate low power mode request 1 L2CC IDLE is masked 0 L2CC IDLE is not masked
26 MASK_SCU_ IDLE	Mask SCU IDLE for entering low power mode **NOTE:** Assertion of all bits[27:22] will generate low power mode request 1 SCU IDLE is masked 0 SCU IDLE is not masked

Table continues on the next page...

CCM_CLPCR field descriptions (continued)

Field	Description
25–23 Reserved	This read-only field is reserved and always has the value 0.
22 MASK_CORE0_WFI	Mask WFI of core0 for entering low power mode **NOTE:** Assertion of all bits[27:22] will generate low power mode request 0 WFI of core0 is not masked 1 WFI of core0 is masked
21 BYPASS_LPM_HS0	Bypass low power mode handshake. **NOTE:** This bit should always be set to 1'b1 by software.
20 -	This field is reserved. Reserved
19 BYPASS_LPM_HS1	Bypass low power mode handshake. **NOTE:** This bit should always be set to 1'b1 by software.
18–12 -	This field is reserved. Reserved
11 COSC_PWRDOWN	In run mode, software can manually control powering down of on chip oscillator, i.e. generating '1' on cosc_pwrdown signal. If software manually powered down the on chip oscillator, then sbyos functionality for on chip oscillator will be bypassed. The manual closing of onchip oscillator should be performed only in case the reference oscilator is not the source of all the clocks generation. 0 On chip oscillator will not be powered down, i.e. cosc_pwrdown = '0'. 1 On chip oscillator will be powered down, i.e. cosc_pwrdown = '1'.
10–9 STBY_COUNT	Standby counter definition. These two bits define, in the case of stop exit (if VSTBY bit was set). **NOTE:** Clock cycles ratio depends on pmic_delay_scaler, defined by CGPR[0] bit. 00 CCM will wait (1*pmic_delay_scaler)+1 ckil clock cycles 01 CCM will wait (3*pmic_delay_scaler)+1 ckil clock cycles 10 CCM will wait (7*pmic_delay_scaler)+1 ckil clock cycles 11 CCM will wait (15*pmic_delay_scaler)+1 ckil clock cycles
8 VSTBY	Voltage standby request bit. This bit defines if PMIC_STBY_REQ pin, which notifies external power management IC to move from functional voltage to standby voltage, will be asserted in STOP mode. 0 Voltage will not be changed to standby voltage after next entrance to STOP mode. (PMIC_STBY_REQ will remain negated - '0') 1 Voltage will be requested to change to standby voltage after next entrance to stop mode. (PMIC_STBY_REQ will be asserted - '1').
7 DIS_REF_OSC	dis_ref_osc - in run mode, software can manually control closing of external reference oscillator clock, i.e. generating '1' on CCM_REF_EN_B signal. If software closed manually the external reference clock, then sbyos functionality will be bypassed. The manual closing of external reference oscilator should be performed only in case the reference oscilator is not the source of any clock generation. **NOTE:** When returning from stop mode, the PMIC_STBY_REQ will be deasserted (if it was asserted when entering stop mode). See stby_count bits.

Table continues on the next page...

CCM_CLPCR field descriptions (continued)

Field	Description
	0 external high frequency oscillator will be enabled, i.e. CCM_REF_EN_B = '0'. 1 external high frequency oscillator will be disabled, i.e. CCM_REF_EN_B = '1'
6 SBYOS	Standby clock oscillator bit. This bit defines if cosc_pwrdown, which power down the on chip oscillator, will be asserted in STOP mode. This bit is discarded if cosc_pwrdown='1' for the on chip oscillator. 0 On-chip oscillator will not be powered down, after next entrance to STOP mode. (CCM_REF_EN_B will remain asserted - '0' and cosc_pwrdown will remain de asserted - '0') 1 On-chip oscillator will be powered down, after next entrance to STOP mode. (CCM_REF_EN_B will be deasserted - '1' and cosc_pwrdown will be asserted - '1'). When returning from STOP mode, external oscillator will be enabled again, on-chip oscillator will return to oscillator mode, and after oscnt count, CCM will continue with the exit from the STOP mode process.
5 ARM_CLK_DIS_ON_LPM	Define if Arm clocks (arm_clk, soc_mxclk, soc_pclk, soc_dbg_pclk, vl_wrck) will be disabled on wait mode. This is useful for debug mode, when the user still wants to simulate entering wait mode and still keep Arm clock functioning. **NOTE:** Software should not enable Arm power gating in wait mode if this bit is cleared. 0 Arm clock enabled on wait mode. 1 Arm clock disabled on wait mode. .
4–3 -	This field is reserved. Reserved
2 -	This field is reserved. Reserved
LPM	Setting the low power mode that system will enter on next assertion of dsm_request signal. 00 Remain in run mode 01 Transfer to wait mode 10 Transfer to stop mode 11 Reserved

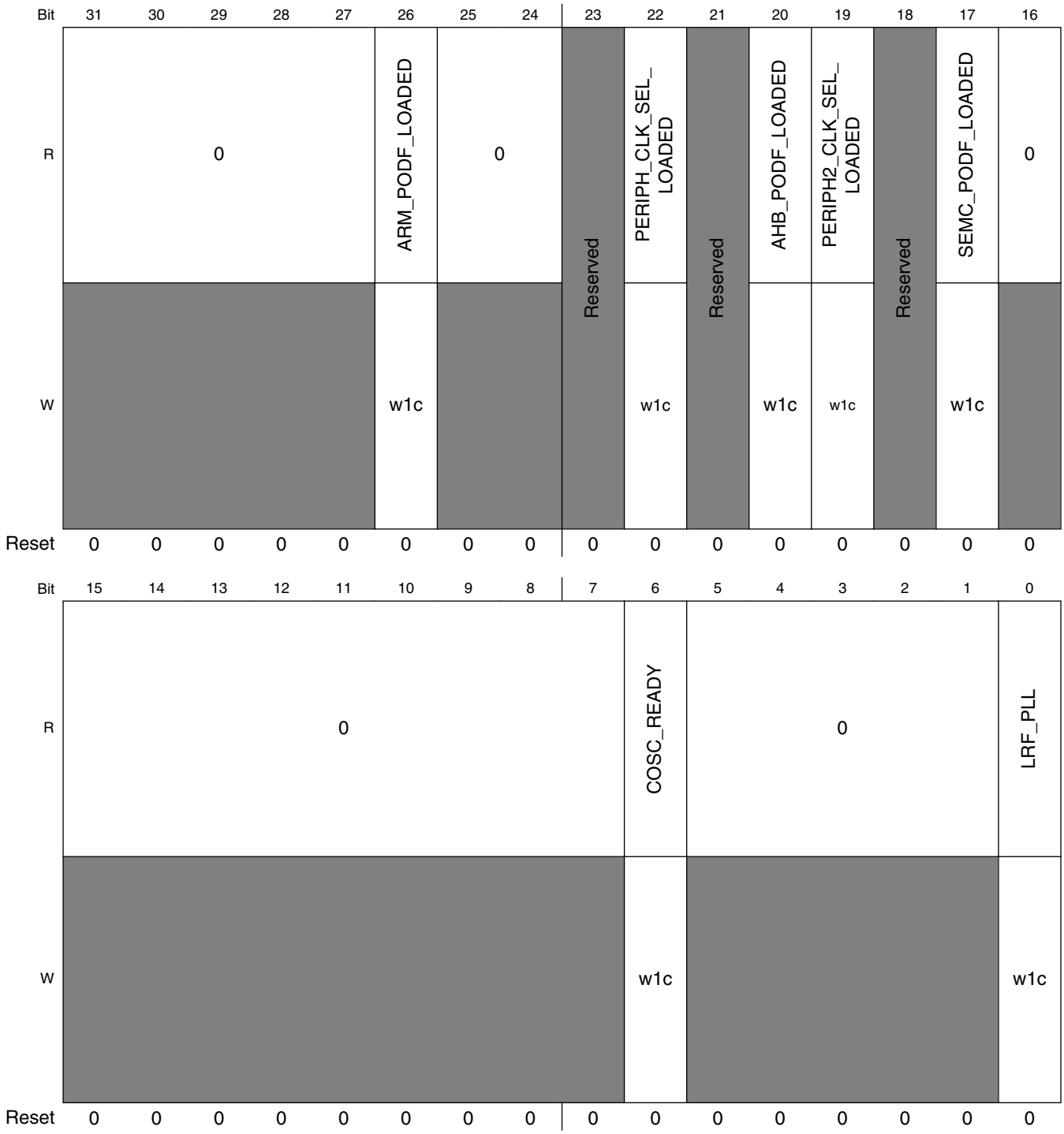
14.7.17 CCM Interrupt Status Register (CCM_CISR)

The figure below represents the CCM Interrupt Status Register (CISR). This is a write one to clear register. After an interrupt is generated, software should write one to clear it. The table below provides its field descriptions.

NOTE

CCM interrupt request 1 can be masked by CCM interrupt request 1 mask bit. CCM interrupt request 2 can be masked by CCM interrupt request 2 mask bit.

Address: 400F_C000h base + 58h offset = 400F_C058h



CCM_CISR field descriptions

Field	Description
31–27 Reserved	This read-only field is reserved and always has the value 0.

Table continues on the next page...

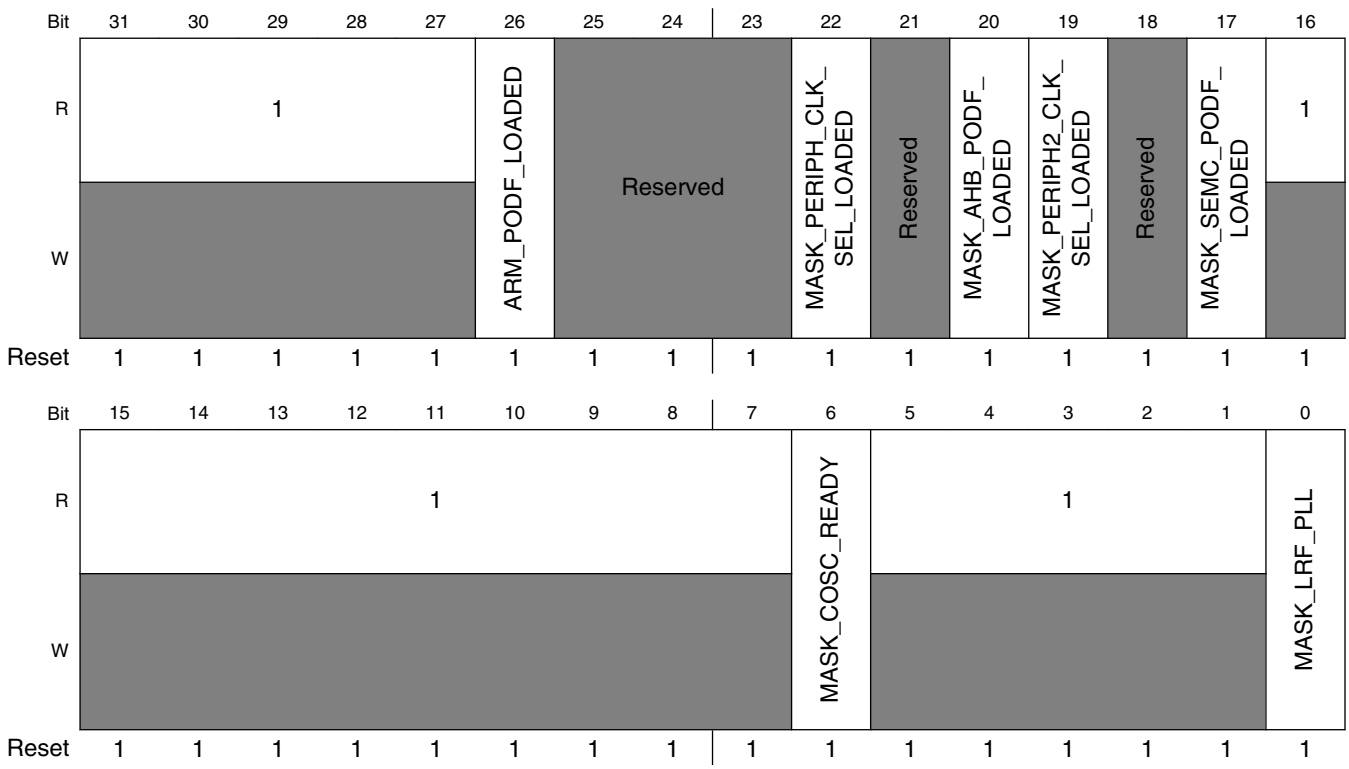
CCM_CISR field descriptions (continued)

Field	Description
26 ARM_PODF_ LOADED	CCM interrupt request 1 generated due to frequency change of arm_podf. The interrupt will commence only if arm_podf is loaded during a DVFS operation. 0 interrupt is not generated due to frequency change of arm_podf 1 interrupt generated due to frequency change of arm_podf
25–24 Reserved	This read-only field is reserved and always has the value 0.
23 -	This field is reserved. Reserved
22 PERIPH_CLK_ SEL_LOADED	CCM interrupt request 1 generated due to update of periph_clk_sel. 0 interrupt is not generated due to update of periph_clk_sel. 1 interrupt generated due to update of periph_clk_sel.
21 -	This field is reserved. Reserved
20 AHB_PODF_ LOADED	CCM interrupt request 1 generated due to frequency change of ahb_podf 0 interrupt is not generated due to frequency change of ahb_podf 1 interrupt generated due to frequency change of ahb_podf
19 PERIPH2_CLK_ SEL_LOADED	CCM interrupt request 1 generated due to frequency change of periph2_clk_sel 0 interrupt is not generated due to frequency change of periph2_clk_sel 1 interrupt generated due to frequency change of periph2_clk_sel
18 -	This field is reserved. Reserved
17 SEMC_PODF_ LOADED	CCM interrupt request 1 generated due to frequency change of semc_podf 0 interrupt is not generated due to frequency change of semc_podf 1 interrupt generated due to frequency change of semc_podf
16–7 Reserved	This read-only field is reserved and always has the value 0.
6 COSC_READY	CCM interrupt request 2 generated due to on board oscillator ready, i.e. oscnt has finished counting. 0 interrupt is not generated due to on board oscillator ready 1 interrupt generated due to on board oscillator ready
5–1 Reserved	This read-only field is reserved and always has the value 0.
0 LRF_PLL	CCM interrupt request 2 generated due to lock of all enabled and not bypaseed PLLs 0 interrupt is not generated due to lock ready of all enabled and not bypaseed PLLs 1 interrupt generated due to lock ready of all enabled and not bypaseed PLLs

14.7.18 CCM Interrupt Mask Register (CCM_CIMR)

The figure below represents the CCM Interrupt Mask Register (CIMR). The table below provides its field descriptions.

Address: 400F_C000h base + 5Ch offset = 400F_C05Ch



CCM_CIMR field descriptions

Field	Description
31–27 Reserved	This read-only field is reserved and always has the value 1.
26 ARM_PODF_LOADED	mask interrupt generation due to frequency change of arm_podf 0 don't mask interrupt due to frequency change of arm_podf - interrupt will be created 1 mask interrupt due to frequency change of arm_podf
25–23 -	This field is reserved. Reserved
22 MASK_PERIPH_CLK_SEL_LOADED	mask interrupt generation due to update of periph_clk_sel. 0 don't mask interrupt due to update of periph_clk_sel - interrupt will be created 1 mask interrupt due to update of periph_clk_sel
21 -	This field is reserved. Reserved

Table continues on the next page...

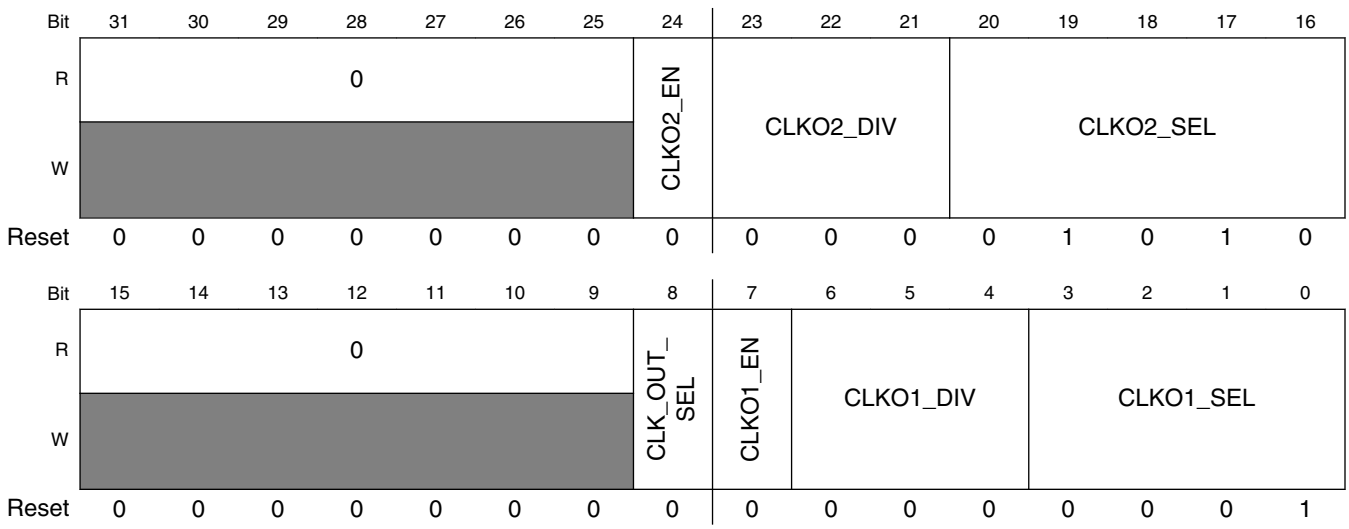
CCM_CIMR field descriptions (continued)

Field	Description
20 MASK_AHB_ PODF_LOADED	mask interrupt generation due to frequency change of ahb_podf 0 don't mask interrupt due to frequency change of ahb_podf - interrupt will be created 1 mask interrupt due to frequency change of ahb_podf
19 MASK_ PERIPH2_CLK_ SEL_LOADED	mask interrupt generation due to update of periph2_clk_sel. 0 don't mask interrupt due to update of periph2_clk_sel - interrupt will be created 1 mask interrupt due to update of periph2_clk_sel
18 -	This field is reserved. Reserved
17 MASK_SEMC_ PODF_LOADED	mask interrupt generation due to frequency change of semc_podf 0 don't mask interrupt due to frequency change of semc_podf - interrupt will be created 1 mask interrupt due to frequency change of semc_podf
16–7 Reserved	This read-only field is reserved and always has the value 1.
6 MASK_COSC_ READY	mask interrupt generation due to on board oscillator ready 0 don't mask interrupt due to on board oscillator ready - interrupt will be created 1 mask interrupt due to on board oscillator ready
5–1 Reserved	This read-only field is reserved and always has the value 1.
0 MASK_LRF_PLL	mask interrupt generation due to lrf of PLLs 0 don't mask interrupt due to lrf of PLLs - interrupt will be created 1 mask interrupt due to lrf of PLLs

14.7.19 CCM Clock Output Source Register (CCM_CCOSR)

The figure below represents the CCM Clock Output Source Register (CCOSR). The CCOSR register contains bits to control the clock that will be generated on the output ipp_do_clko1.The table below provides its field descriptions.

Address: 400F_C000h base + 60h offset = 400F_C060h



CCM_CCOSR field descriptions

Field	Description
31–25 Reserved	This read-only field is reserved and always has the value 0.
24 CLKO2_EN	Enable of CCM_CLKO2 clock 0 CCM_CLKO2 disabled. 1 CCM_CLKO2 enabled.
23–21 CLKO2_DIV	Setting the divider of CCM_CLKO2 000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
20–16 CLKO2_SEL	Selection of the clock to be generated on CCM_CLKO2 00011 usdhc1_clk_root 00110 lpi2c_clk_root

Table continues on the next page...

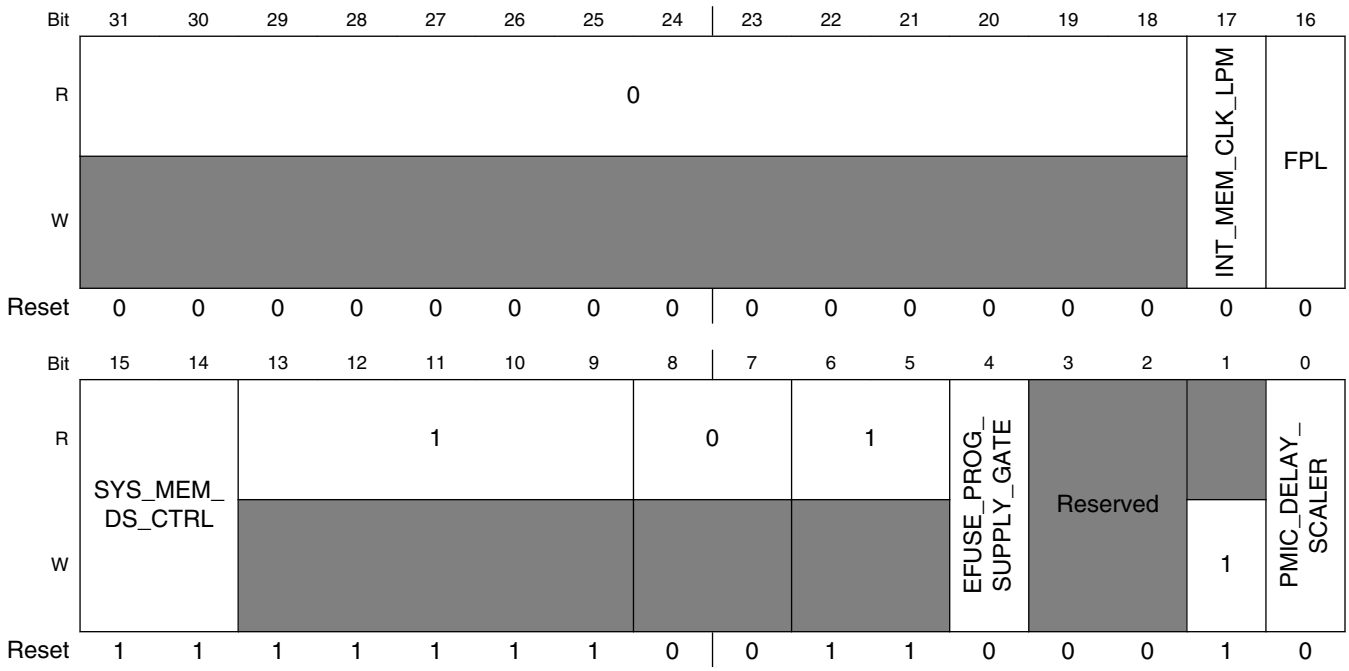
CCM_CCOSR field descriptions (continued)

Field	Description
	01011 csi_clk_root 01110 osc_clk 10001 usdhc2_clk_root 10010 sai1_clk_root 10011 sai2_clk_root 10100 sai3_clk_root (shared with ADC1 and ADC2 alt_clk root) 10111 can_clk_root (FlexCAN, shared with CANFD) 11011 flexspi_clk_root 11100 uart_clk_root 11101 spdif0_clk_root 11111 Reserved
15–9 Reserved	This read-only field is reserved and always has the value 0.
8 CLK_OUT_SEL	CCM_CLKO1 output to reflect CCM_CLKO1 or CCM_CLKO2 clocks 0 CCM_CLKO1 output drives CCM_CLKO1 clock 1 CCM_CLKO1 output drives CCM_CLKO2 clock
7 CLKO1_EN	Enable of CCM_CLKO1 clock 0 CCM_CLKO1 disabled. 1 CCM_CLKO1 enabled.
6–4 CLKO1_DIV	Setting the divider of CCM_CLKO1 000 divide by 1 001 divide by 2 010 divide by 3 011 divide by 4 100 divide by 5 101 divide by 6 110 divide by 7 111 divide by 8
CLKO1_SEL	Selection of the clock to be generated on CCM_CLKO1 0000 USB1 PLL clock (divided by 2) 0001 SYS PLL clock (divided by 2) 0011 VIDEO PLL clock (divided by 2) 0101 semc_clk_root 0110 Reserved 1010 lcdif_pix_clk_root 1011 ahb_clk_root 1100 ipg_clk_root 1101 perclk_root 1110 ckil_sync_clk_root 1111 pll4_main_clk

14.7.20 CCM General Purpose Register (CCM_CGPR)

Fast PLL enable. Can be used to engage PLL faster after STOP mode, if 24MHz OSC was active

Address: 400F_C000h base + 64h offset = 400F_C064h



CCM_CGPR field descriptions

Field	Description
31–18 Reserved	This read-only field is reserved and always has the value 0.
17 INT_MEM_CLK_LPM	Control for the Deep Sleep signal to the Arm Platform memories with additional control logic based on the Arm WFI signal. Used to keep the Arm Platform memory clocks enabled if an interrupt is pending when entering low power mode. **NOTE:** This bit should always bet set when the CCM_CLPCR_LPM bits are set to 01(WAIT Mode) or 10 (STOP mode) without power gating. This bit does not have to be set for STOP mode entry. 0 Disable the clock to the Arm platform memories when entering Low Power Mode 1 Keep the clocks to the Arm platform memories enabled only if an interrupt is pending when entering Low Power Modes (WAIT and STOP without power gating)
16 FPL	Fast PLL enable. 0 Engage PLL enable default way. 1 Engage PLL enable 3 CKIL clocks earlier at exiting low power mode (STOP). Should be used only if 24MHz OSC was active in low power mode.

Table continues on the next page...

CCM_CGPR field descriptions (continued)

Field	Description
15–14 SYS_MEM_DS_CTRL	System memory DS control 00 Disable memory DS mode always 01 Enable memory (outside Arm platform) DS mode when system STOP and PLL are disabled 1x enable memory (outside Arm platform) DS mode when system is in STOP mode
13–9 Reserved	This read-only field is reserved and always has the value 1.
8–7 Reserved	This read-only field is reserved and always has the value 0.
6–5 Reserved	This read-only field is reserved and always has the value 1.
4 EFUSE_PROG_SUPPLY_GATE	Defines the value of the output signal cgpr_dout[4]. Gate of program supply for efuse programing 0 fuse programing supply voltage is gated off to the efuse module 1 allow fuse programing.
3–2 -	This field is reserved. Reserved
1 -	Reserved. Keep default value set to '1' for proper operation.
0 PMIC_DELAY_SCALER	Defines clock dividion of clock for stby_count (pmic delay counter) 0 clock is not divided 1 clock is divided /8

14.7.21 CCM Clock Gating Register 0 (CCM_CCGR0)

CG(i) bits CCGR 0-6

These bits are used to turn on/off the clock to each module independently. The following table details the possible clock activity conditions for each module.

CGR value	Clock Activity Description
00	Clock is off during all modes. Stop enter hardware handshake is disabled.
01	Clock is on in run mode, but off in WAIT and STOP modes
10	Not applicable (Reserved).
11	Clock is on during all modes, except STOP mode.

Module should be stopped, before set its bits to "0"; clocks to the module will be stopped immediately.

The tables above show the register mappings for the different CGRs. The clock connectivity table should be used to match the "CCM output affected" to the actual clocks going into the modules.

The figure below represents the CCM Clock Gating Register 0 (CCM_CCGR0). The clock gating Registers define the clock gating for power reduction of each clock (CG(i) bits). There are 7 CGR registers. The number of registers required is according to the number of peripherals in the system.

Address: 400F_C000h base + 68h offset = 400F_C068h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CG15		CG14		CG13		CG12		CG11		CG10		CG9		CG8	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CG7		CG6		CG5		CG4		CG3		CG2		CG1		CG0	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

CCM_CCGR0 field descriptions

Field	Description
31–30 CG15	gpio2_clocks (gpio2_clk_enable)
29–28 CG14	lpuart2 clock (lpuart2_clk_enable)
27–26 CG13	gpt2 serial clocks (gpt2_serial_clk_enable)
25–24 CG12	gpt2 bus clocks (gpt2_bus_clk_enable)
23–22 CG11	trace clock (trace_clk_enable)
21–20 CG10	can2_serial clock (can2_serial_clk_enable)
19–18 CG9	can2 clock (can2_clk_enable)
17–16 CG8	can1_serial clock (can1_serial_clk_enable)
15–14 CG7	can1 clock (can1_clk_enable)
13–12 CG6	lpuart3 clock (lpuart3_clk_enable)
11–10 CG5	dcp clock (dcp_clk_enable)
9–8 CG4	sim_m or sim_main register access clock (sim_m_mainclk_r_enable)
7–6 CG3	Reserved
5–4 CG2	mqs clock (mqs_hmclk_clock_enable)
3–2 CG1	aips_tz2 clocks (aips_tz2_clk_enable)
CG0	aips_tz1 clocks (aips_tz1_clk_enable)

14.7.22 CCM Clock Gating Register 1 (CCM_CCGR1)

The figure below represents the CCM Clock Gating Register 1(CCM_CCGR1). The clock gating registers define the clock gating for power reduction of each clock (CG(i) bits). There are 8 CGR registers. The number of registers required is determined by the number of peripherals in the system.

Address: 400F_C000h base + 6Ch offset = 400F_C06Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CG15		CG14		CG13		CG12		CG11		CG10		CG9		CG8	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CG7		CG6		CG5		CG4		CG3		CG2		CG1		CG0	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

CCM_CCGR1 field descriptions

Field	Description
31–30 CG15	gpio5 clock (gpio5_clk_enable)
29–28 CG14	csu clock (csu_clk_enable)
27–26 CG13	gpio1 clock (gpio1_clk_enable)
25–24 CG12	lpuart4 clock (lpuart4_clk_enable)
23–22 CG11	gpt1 serial clock (gpt_serial_clk_enable)
21–20 CG10	gpt1 bus clock (gpt_clk_enable)
19–18 CG9	semc_exsc clock (semc_exsc_clk_enable)
17–16 CG8	adc1 clock (adc1_clk_enable)
15–14 CG7	aoi2 clocks (aoi2_clk_enable)
13–12 CG6	pit clocks (pit_clk_enable)
11–10 CG5	enet clock (enet_clk_enable)
9–8 CG4	adc2 clock (adc2_clk_enable)

Table continues on the next page...

CCM_CCGR1 field descriptions (continued)

Field	Description
7–6 CG3	lpspi4 clocks (lpspi4_clk_enable)
5–4 CG2	lpspi3 clocks (lpspi3_clk_enable)
3–2 CG1	lpspi2 clocks (lpspi2_clk_enable)
CG0	lpspi1 clocks (lpspi1_clk_enable)

14.7.23 CCM Clock Gating Register 2 (CCM_CCGR2)

The figure below represents the CCM Clock Gating Register 2 (CCM_CCGR2). The clock gating registers define the clock gating for power reduction of each clock (CG(i) bits). There are 8 CGR registers. The number of registers required is determined by the number of peripherals in the system.

Address: 400F_C000h base + 70h offset = 400F_C070h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CG15		CG14		CG13		CG12		CG11		CG10		CG9		CG8	
Reset	1	1	1	1	1	1	0	0	0	0	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CG7		CG6		CG5		CG4		CG3		CG2		CG1		CG0	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

CCM_CCGR2 field descriptions

Field	Description
31–30 CG15	pxp clocks (pxp_clk_enable)
29–28 CG14	lcd clocks (lcd_clk_enable)
27–26 CG13	gpio3 clock (gpio3_clk_enable)
25–24 CG12	xbar2 clock (xbar2_clk_enable)
23–22 CG11	xbar1 clock (xbar1_clk_enable)
21–20 CG10	ipmux3 clock (ipmux3_clk_enable)
19–18 CG9	ipmux2 clock (ipmux2_clk_enable)

Table continues on the next page...

CCM_CCGR2 field descriptions (continued)

Field	Description
17–16 CG8	ipmux1 clock (ipmux1_clk_enable)
15–14 CG7	xbar3 clock (xbar3_clk_enable)
13–12 CG6	OCOTP_CTRL clock (ocotp_clk_enable)
11–10 CG5	lpi2c3 clock (lpi2c3_clk_enable) Wire1 (Wire2 on MM)
9–8 CG4	lpi2c2 clock (lpi2c2_clk_enable) Wire3 on MM
7–6 CG3	lpi2c1 clock (lpi2c1_clk_enable) Wire
5–4 CG2	iomuxc_snvs clock (iomuxc_snvs_clk_enable)
3–2 CG1	csi clock (csi_clk_enable)
CG0	ocram_exsc clock (ocram_exsc_clk_enable)

14.7.24 CCM Clock Gating Register 3 (CCM_CCGR3)

The figure below represents the CCM Clock Gating Register 3 (CCM_CCGR3). The clock gating Registers define the clock gating for power reduction of each clock (CG(i) bits). There are 8 CGR registers. The number of registers required is determined by the number of peripherals in the system.

NOTE

The OCRAM clock cannot be turned off when CM7 Cache is running. For details, refer to CCM_CCGR3[CG14] bitfield.

Address: 400F_C000h base + 74h offset = 400F_C074h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CG15		CG14		CG13		CG12		CG11		CG10		CG9		CG8	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CG7		CG6		CG5		CG4		CG3		CG2		CG1		CG0	
Reset	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1

CCM_CCGR3 field descriptions

Field	Description
31–30 CG15	iomuxc_snvs_gpr clock (iomuxc_snvs_gpr_clk_enable)
29–28 CG14	The OCRAM clock cannot be turned off when the CM cache is running on this device. ocram clock(ocram_clk_enable)
27–26 CG13	acmp4 clocks (acmp4_clk_enable)
25–24 CG12	acmp3 clocks (acmp3_clk_enable)
23–22 CG11	acmp2 clocks (acmp2_clk_enable)
21–20 CG10	acmp1 clocks (acmp1_clk_enable)
19–18 CG9	flexram clock (flexram_clk_enable)
17–16 CG8	wdog1 clock (wdog1_clk_enable)
15–14 CG7	ewm clocks (ewm_clk_enable)
13–12 CG6	gpio4 clock (gpio4_clk_enable)
11–10 CG5	lcdif pix clock (lcdif_pix_clk_enable)
9–8 CG4	aoi1 clock (aoi1_clk_enable)
7–6 CG3	lpuart6 clock (lpuart6_clk_enable)
5–4 CG2	semc clocks (semc_clk_enable)
3–2 CG1	lpuart5 clock (lpuart5_clk_enable)
CG0	flexio2 clocks (flexio2_clk_enable)

14.7.25 CCM Clock Gating Register 4 (CCM_CCGR4)

The figure below represents the CCM Clock Gating Register 4 (CCM_CCGR4). The clock gating Registers define the clock gating for power reduction of each clock (CG(i) bits). There are 8 CGR registers. The number of registers required is determined by the number of peripherals in the system.

Address: 400F_C000h base + 78h offset = 400F_C078h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CG15		CG14		CG13		CG12		CG11		CG10		CG9		CG8	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CG7		CG6		CG5		CG4		CG3		CG2		CG1		CG0	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

CCM_CCGR4 field descriptions

Field	Description
31–30 CG15	qdc4 clocks (qdc4_clk_enable)
29–28 CG14	qdc3 clocks (qdc3_clk_enable)
27–26 CG13	qdc2 clocks (qdc2_clk_enable)
25–24 CG12	qdc1 clocks (qdc1_clk_enable)
23–22 CG11	pwm4 clocks (pwm4_clk_enable)
21–20 CG10	pwm3 clocks (pwm3_clk_enable)
19–18 CG9	pwm2 clocks (pwm2_clk_enable)
17–16 CG8	pwm1 clocks (pwm1_clk_enable)
15–14 CG7	sim_ems clocks (sim_ems_clk_enable)
13–12 CG6	sim_m clocks (sim_m_clk_enable)
11–10 CG5	tsc_dig clock (tsc_clk_enable)
9–8 CG4	sim_m7 clock (sim_m7_clk_enable)

Table continues on the next page...

CCM_CCGR4 field descriptions (continued)

Field	Description
7–6 CG3	bee clock(bee_clk_enable)
5–4 CG2	iomuxc gpr clock (iomuxc_gpr_clk_enable)
3–2 CG1	iomuxc clock (iomuxc_clk_enable)
CG0	sim_m7 register access clock (sim_m7_mainclk_r_enable)

14.7.26 CCM Clock Gating Register 5 (CCM_CCGR5)

The figure below represents the CCM Clock Gating Register 5 (CCM_CCGR5). The clock gating Registers define the clock gating for power reduction of each clock (CG(i) bits). There are 8 CGR registers. The number of registers required is determined by the number of peripherals in the system.

Address: 400F_C000h base + 7Ch offset = 400F_C07Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CG15		CG14		CG13		CG12		CG11		CG10		CG9		CG8	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CG7		CG6		CG5		CG4		CG3		CG2		CG1		CG0	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

CCM_CCGR5 field descriptions

Field	Description
31–30 CG15	snvs_lp clock (snvs_lp_clk_enable)
29–28 CG14	snvs_hp clock (snvs_hp_clk_enable)
27–26 CG13	lpuart7 clock (lpuart7_clk_enable)
25–24 CG12	lpuart1 clock (lpuart1_clk_enable)
23–22 CG11	sai3 clock (sai3_clk_enable)
21–20 CG10	sai2 clock (sai2_clk_enable)
19–18 CG9	sai1 clock (sai1_clk_enable)

Table continues on the next page...

CCM_CCGR5 field descriptions (continued)

Field	Description
17–16 CG8	sim_main clock (sim_main_clk_enable)
15–14 CG7	spdif clock (spdif_clk_enable)
13–12 CG6	aipstz4 clocks (aips_tz4_clk_enable)
11–10 CG5	wdog2 clock (wdog2_clk_enable)
9–8 CG4	kpp clock (kpp_clk_enable)
7–6 CG3	dma clock (dma_clk_enable)
5–4 CG2	wdog3 clock (wdog3_clk_enable)
3–2 CG1	flexio1 clock (flexio1_clk_enable)
CG0	rom clock (rom_clk_enable)

14.7.27 CCM Clock Gating Register 6 (CCM_CCGR6)

The figure below represents the CCM Clock Gating Register 6 (CCM_CCGR6). The clock gating Registers define the clock gating for power reduction of each clock (CG(i) bits). There are 8 CGR registers. The number of registers required is determined by the number of peripherals in the system.

Address: 400F_C000h base + 80h offset = 400F_C080h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CG15		CG14		CG13		CG12		CG11		CG10		CG9		CG8	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CG7		CG6		CG5		CG4		CG3		CG2		CG1		CG0	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

CCM_CCGR6 field descriptions

Field	Description
31–30 CG15	timer3 clocks (timer3_clk_enable)
29–28 CG14	timer2 clocks (timer2_clk_enable)

Table continues on the next page...

CCM_CCGR6 field descriptions (continued)

Field	Description
27–26 CG13	timer1 clocks (timer1_clk_enable)
25–24 CG12	lpi2c4 serial clock (lpi2c4_serial_clk_enable) Wire2 (Wire1 on MM)
23–22 CG11	anadig clocks (anadig_clk_enable)
21–20 CG10	sim_per clock (sim_per_clk_enable)sim_axbs_p_clk_enable
19–18 CG9	aips_tz3 clock (aips_tz3_clk_enable)
17–16 CG8	timer4 clocks (timer4_clk_enable)
15–14 CG7	lpuart8 clocks (lpuart8_clk_enable)
13–12 CG6	trng clock (trng_clk_enable)
11–10 CG5	flexspi clocks (flexspi_clk_enable) **NOTE:** sim_ems_clk_enable must also be cleared, when flexspi_clk_enable is cleared.
9–8 CG4	ipmux4 clock (ipmux4_clk_enable)
7–6 CG3	dcdc clocks (dcdc_clk_enable)
5–4 CG2	usdhc2 clocks (usdhc2_clk_enable)
3–2 CG1	usdhc1 clocks (usdhc1_clk_enable)
CG0	usboh3 clock (usboh3_clk_enable)

14.7.28 CCM Clock Gating Register 7 (CCM_CCGR7)

The figure below represents the CCM Clock Gating Register 7 (CCM_CCGR7). The clock gating Registers define the clock gating for power reduction of each clock (CG(i) bits). There are 8 CGR registers. The number of registers required is determined by the number of peripherals in the system.

Address: 400F_C000h base + 84h offset = 400F_C084h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved		Reserved		Reserved		Reserved		Reserved		Reserved		Reserved		Reserved	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved		CG6		CG5		CG4		CG3		CG2		CG1		CG0	
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

CCM_CCGR7 field descriptions

Field	Description
31–30 -	This field is reserved. Reserved
29–28 -	This field is reserved. Reserved
27–26 -	This field is reserved. Reserved
25–24 -	This field is reserved. Reserved
23–22 -	This field is reserved. Reserved
21–20 -	This field is reserved. Reserved
19–18 -	This field is reserved. Reserved
17–16 -	This field is reserved. Reserved
15–14 -	This field is reserved. Reserved
13–12 CG6	flexio3_clk_enable
11–10 CG5	aips_lite_clk_enable
9–8 CG4	can3_serial_clk_enable

Table continues on the next page...

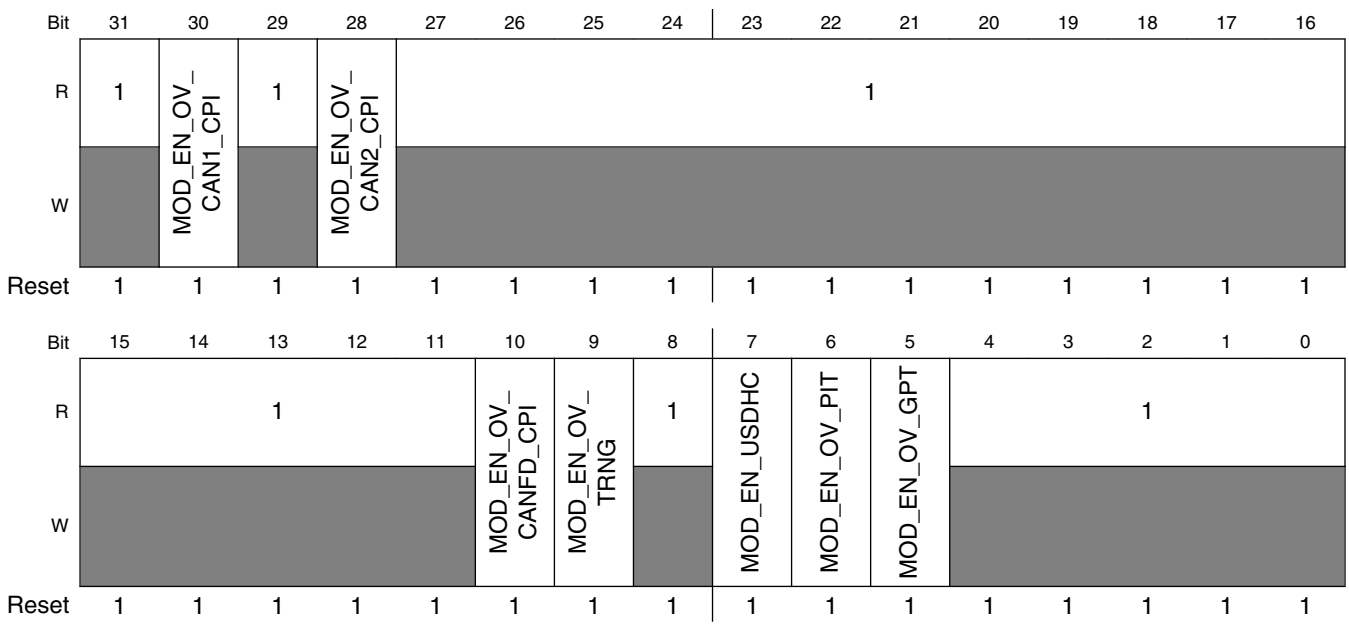
CCM_CCGR7 field descriptions (continued)

Field	Description
7–6 CG3	can3_clk_enable
5–4 CG2	axbs_l_clk_enable
3–2 CG1	flexspi2_clk_enable
CG0	enet2_clk_enable

14.7.29 CCM Module Enable Overide Register (CCM_CMEOR)

The follow figure represents the CCM Module Enable Override Register (CMEOR). The CMEOR register contains bits to override the clock enable signal from the module. This bit will be applicable only for modules whose clock enable signals are used. The following table provides its field descriptions.

Address: 400F_C000h base + 88h offset = 400F_C088h



CCM_CMEOR field descriptions

Field	Description
31 Reserved	This read-only field is reserved and always has the value 1.

Table continues on the next page...

CCM_CMEOR field descriptions (continued)

Field	Description
30 MOD_EN_OV_ CAN1_CPI	Overide clock enable signal from CAN1 - clock will not be gated based on CAN's signal 'enable_clk_cpi'. 0 don't overide module enable signal 1 overide module enable signal
29 Reserved	This read-only field is reserved and always has the value 1.
28 MOD_EN_OV_ CAN2_CPI	Overide clock enable signal from CAN2 - clock will not be gated based on CAN's signal 'enable_clk_cpi'. 0 don't override module enable signal 1 override module enable signal
27–11 Reserved	This read-only field is reserved and always has the value 1.
10 MOD_EN_OV_ CANFD_CPI	Overide clock enable signal from FlexCAN3(CANFD) - clock will not be gated based on CAN's signal 'enable_clk_cpi'. 0 don't override module enable signal 1 override module enable signal
9 MOD_EN_OV_ TRNG	Overide clock enable signal from TRNG 0 don't override module enable signal 1 override module enable signal
8 Reserved	This read-only field is reserved and always has the value 1.
7 MOD_EN_ USDHC	overide clock enable signal from USDHC. 0 don't override module enable signal 1 override module enable signal
6 MOD_EN_OV_ PIT	Overide clock enable signal from PIT - clock will not be gated based on PIT's signal 'ipg_enable_clk' . 0 don't override module enable signal 1 override module enable signal
5 MOD_EN_OV_ GPT	Overide clock enable signal from GPT - clock will not be gated based on GPT's signal 'ipg_enable_clk' . 0 don't override module enable signal 1 override module enable signal
Reserved	This read-only field is reserved and always has the value 1.

14.8 CCM Analog Memory Map/Register Definition

This section describes the registers for the analog PLLs. The registers which have the same description are grouped within {}. The register offsets for the various PLLs are:

- ARM PLL: {0h000, 0h004, 0h008, 0h00C}.
- USB1 PLL: {0h010, 0h014, 0h018, 0h01C}, {0h0F0, 0h0F4, 0h0F8, 0h0FC}.
- System PLL: {0h030, 0h034, 0h038, 0h03C}, 0h040, 0h050, 0h060, {0h100, 0h104, 0h108, 0h10C}.
- Audio / Video PLL: {0h070, 0h074, 0h078, 0h07C}, 0h080, 0h090, {0h0A0, 0h0A4, 0h0A8, 0h0AC}, 0h0B0, 0h0C0

CCM_ANALOG memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400D_8000	Analog ARM PLL control Register (CCM_ANALOG_PLL_ARM)	32	R/W	0001_3063h	14.8.1/1093
400D_8004	Analog ARM PLL control Register (CCM_ANALOG_PLL_ARM_SET)	32	R/W	0001_3063h	14.8.1/1093
400D_8008	Analog ARM PLL control Register (CCM_ANALOG_PLL_ARM_CLR)	32	R/W	0001_3063h	14.8.1/1093
400D_800C	Analog ARM PLL control Register (CCM_ANALOG_PLL_ARM_TOG)	32	R/W	0001_3063h	14.8.1/1093
400D_8010	Analog USB1 480MHz PLL Control Register (CCM_ANALOG_PLL_USB1)	32	R/W	0001_2000h	14.8.2/1095
400D_8014	Analog USB1 480MHz PLL Control Register (CCM_ANALOG_PLL_USB1_SET)	32	R/W	0001_2000h	14.8.2/1095
400D_8018	Analog USB1 480MHz PLL Control Register (CCM_ANALOG_PLL_USB1_CLR)	32	R/W	0001_2000h	14.8.2/1095
400D_801C	Analog USB1 480MHz PLL Control Register (CCM_ANALOG_PLL_USB1_TOG)	32	R/W	0001_2000h	14.8.2/1095
400D_8020	Analog USB2 480MHz PLL Control Register (CCM_ANALOG_PLL_USB2)	32	R/W	0001_2000h	14.8.3/1098
400D_8024	Analog USB2 480MHz PLL Control Register (CCM_ANALOG_PLL_USB2_SET)	32	R/W	0001_2000h	14.8.3/1098
400D_8028	Analog USB2 480MHz PLL Control Register (CCM_ANALOG_PLL_USB2_CLR)	32	R/W	0001_2000h	14.8.3/1098
400D_802C	Analog USB2 480MHz PLL Control Register (CCM_ANALOG_PLL_USB2_TOG)	32	R/W	0001_2000h	14.8.3/1098
400D_8030	Analog System PLL Control Register (CCM_ANALOG_PLL_SYS)	32	R/W	0001_3001h	14.8.4/1100
400D_8034	Analog System PLL Control Register (CCM_ANALOG_PLL_SYS_SET)	32	R/W	0001_3001h	14.8.4/1100
400D_8038	Analog System PLL Control Register (CCM_ANALOG_PLL_SYS_CLR)	32	R/W	0001_3001h	14.8.4/1100

Table continues on the next page...

CCM_ANALOG memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400D_803C	Analog System PLL Control Register (CCM_ANALOG_PLL_SYS_TOG)	32	R/W	0001_3001h	14.8.4/1100
400D_8040	528MHz System PLL Spread Spectrum Register (CCM_ANALOG_PLL_SYS_SS)	32	R/W	0000_0000h	14.8.5/1101
400D_8050	Numerator of 528MHz System PLL Fractional Loop Divider Register (CCM_ANALOG_PLL_SYS_NUM)	32	R/W	0000_0000h	14.8.6/1102
400D_8060	Denominator of 528MHz System PLL Fractional Loop Divider Register (CCM_ANALOG_PLL_SYS_DENOM)	32	R/W	0000_0012h	14.8.7/1103
400D_8070	Analog Audio PLL control Register (CCM_ANALOG_PLL_AUDIO)	32	R/W	0001_1006h	14.8.8/1104
400D_8074	Analog Audio PLL control Register (CCM_ANALOG_PLL_AUDIO_SET)	32	R/W	0001_1006h	14.8.8/1104
400D_8078	Analog Audio PLL control Register (CCM_ANALOG_PLL_AUDIO_CLR)	32	R/W	0001_1006h	14.8.8/1104
400D_807C	Analog Audio PLL control Register (CCM_ANALOG_PLL_AUDIO_TOG)	32	R/W	0001_1006h	14.8.8/1104
400D_8080	Numerator of Audio PLL Fractional Loop Divider Register (CCM_ANALOG_PLL_AUDIO_NUM)	32	R/W	05F5_E100h	14.8.9/1106
400D_8090	Denominator of Audio PLL Fractional Loop Divider Register (CCM_ANALOG_PLL_AUDIO_DENOM)	32	R/W	2964_619Ch	14.8.10/ 1106
400D_80A0	Analog Video PLL control Register (CCM_ANALOG_PLL_VIDEO)	32	R/W	0001_100Ch	14.8.11/ 1108
400D_80A4	Analog Video PLL control Register (CCM_ANALOG_PLL_VIDEO_SET)	32	R/W	0001_100Ch	14.8.11/ 1108
400D_80A8	Analog Video PLL control Register (CCM_ANALOG_PLL_VIDEO_CLR)	32	R/W	0001_100Ch	14.8.11/ 1108
400D_80AC	Analog Video PLL control Register (CCM_ANALOG_PLL_VIDEO_TOG)	32	R/W	0001_100Ch	14.8.11/ 1108
400D_80B0	Numerator of Video PLL Fractional Loop Divider Register (CCM_ANALOG_PLL_VIDEO_NUM)	32	R/W	05F5_E100h	14.8.12/ 1110
400D_80C0	Denominator of Video PLL Fractional Loop Divider Register (CCM_ANALOG_PLL_VIDEO_DENOM)	32	R/W	10A2_4447h	14.8.13/ 1110
400D_80E0	Analog ENET PLL Control Register (CCM_ANALOG_PLL_ENET)	32	R/W	0001_1001h	14.8.14/ 1112
400D_80E4	Analog ENET PLL Control Register (CCM_ANALOG_PLL_ENET_SET)	32	R/W	0001_1001h	14.8.14/ 1112
400D_80E8	Analog ENET PLL Control Register (CCM_ANALOG_PLL_ENET_CLR)	32	R/W	0001_1001h	14.8.14/ 1112
400D_80EC	Analog ENET PLL Control Register (CCM_ANALOG_PLL_ENET_TOG)	32	R/W	0001_1001h	14.8.14/ 1112
400D_80F0	480MHz Clock (PLL3) Phase Fractional Divider Control Register (CCM_ANALOG_PFD_480)	32	R/W	1311_100Ch	14.8.15/ 1114
400D_80F4	480MHz Clock (PLL3) Phase Fractional Divider Control Register (CCM_ANALOG_PFD_480_SET)	32	R/W	1311_100Ch	14.8.15/ 1114

Table continues on the next page...

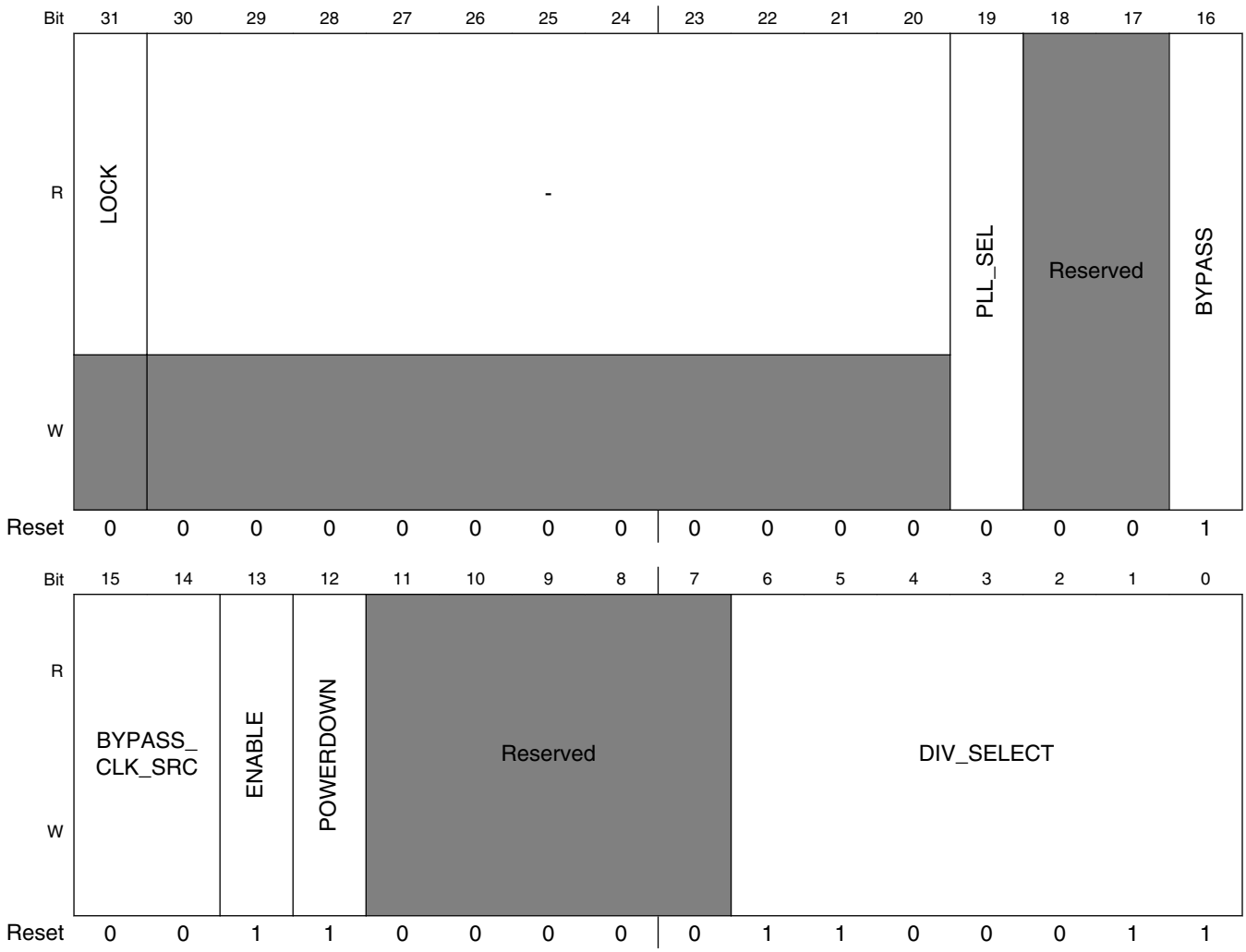
CCM_ANALOG memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400D_80F8	480MHz Clock (PLL3) Phase Fractional Divider Control Register (CCM_ANALOG_PFD_480_CLR)	32	R/W	1311_100Ch	14.8.15/ 1114
400D_80FC	480MHz Clock (PLL3) Phase Fractional Divider Control Register (CCM_ANALOG_PFD_480_TOG)	32	R/W	1311_100Ch	14.8.15/ 1114
400D_8100	528MHz Clock (PLL2) Phase Fractional Divider Control Register (CCM_ANALOG_PFD_528)	32	R/W	1018_101Bh	14.8.16/ 1116
400D_8104	528MHz Clock (PLL2) Phase Fractional Divider Control Register (CCM_ANALOG_PFD_528_SET)	32	R/W	1018_101Bh	14.8.16/ 1116
400D_8108	528MHz Clock (PLL2) Phase Fractional Divider Control Register (CCM_ANALOG_PFD_528_CLR)	32	R/W	1018_101Bh	14.8.16/ 1116
400D_810C	528MHz Clock (PLL2) Phase Fractional Divider Control Register (CCM_ANALOG_PFD_528_TOG)	32	R/W	1018_101Bh	14.8.16/ 1116
400D_8150	Miscellaneous Register 0 (CCM_ANALOG_MISC0)	32	R/W	0400_0000h	14.8.17/ 1119
400D_8154	Miscellaneous Register 0 (CCM_ANALOG_MISC0_SET)	32	R/W	0400_0000h	14.8.17/ 1119
400D_8158	Miscellaneous Register 0 (CCM_ANALOG_MISC0_CLR)	32	R/W	0400_0000h	14.8.17/ 1119
400D_815C	Miscellaneous Register 0 (CCM_ANALOG_MISC0_TOG)	32	R/W	0400_0000h	14.8.17/ 1119
400D_8160	Miscellaneous Register 1 (CCM_ANALOG_MISC1)	32	R/W	0000_0000h	14.8.18/ 1123
400D_8164	Miscellaneous Register 1 (CCM_ANALOG_MISC1_SET)	32	R/W	0000_0000h	14.8.18/ 1123
400D_8168	Miscellaneous Register 1 (CCM_ANALOG_MISC1_CLR)	32	R/W	0000_0000h	14.8.18/ 1123
400D_816C	Miscellaneous Register 1 (CCM_ANALOG_MISC1_TOG)	32	R/W	0000_0000h	14.8.18/ 1123
400D_8170	Miscellaneous Register 2 (CCM_ANALOG_MISC2)	32	R/W	0027_2727h	14.8.19/ 1126
400D_8174	Miscellaneous Register 2 (CCM_ANALOG_MISC2_SET)	32	R/W	0027_2727h	14.8.19/ 1126
400D_8178	Miscellaneous Register 2 (CCM_ANALOG_MISC2_CLR)	32	R/W	0027_2727h	14.8.19/ 1126
400D_817C	Miscellaneous Register 2 (CCM_ANALOG_MISC2_TOG)	32	R/W	0027_2727h	14.8.19/ 1126

14.8.1 Analog ARM PLL control Register (CCM_ANALOG_PLL_ARM*n*)

The control register provides control for the system PLL.

Address: 400D_8000h base + 0h offset + (4d × i), where i=0d to 3d



CCM_ANALOG_PLL_ARM*n* field descriptions

Field	Description
31 LOCK	1 - PLL is currently locked. 0 - PLL is not currently locked.
30–20 -	Always set to zero (0).
19 PLL_SEL	Reserved

Table continues on the next page...

CCM_ANALOG_PLL_ARM*n* field descriptions (continued)

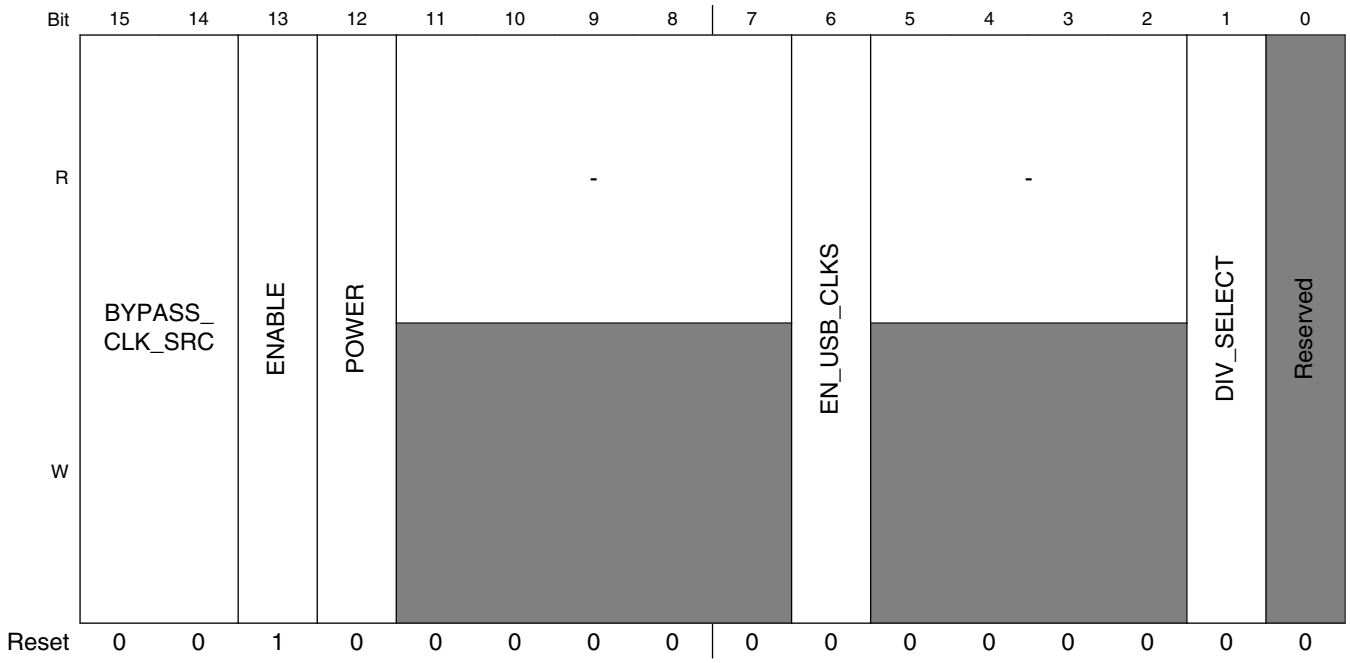
Field	Description
18–17 -	This field is reserved. Reserved
16 BYPASS	Bypass the PLL.
15–14 BYPASS_CLK_SRC	Determines the bypass source. **NOTE:** Changing the Bypass clock source also changes the PLL reference clock source. 0x0 **REF_CLK_24M** — Select the 24MHz oscillator as source. 0x1 **CLK1** — Select the CLK1_N / CLK1_P as source. 0x2 **Reserved1** — 0x3 **Reserved2** —
13 ENABLE	Enable the clock output.
12 POWERDOWN	Powers down the PLL.
11–7 -	This field is reserved. Reserved.
DIV_SELECT	This field controls the PLL loop divider. Valid range for divider value: 54-108. Fout = Fin * div_select/2.0.

14.8.2 Analog USB1 480MHz PLL Control Register (CCM_ANALOG_PLL_USB1*n*)

The control register provides control for USBPHY0 480MHz PLL.

Address: 400D_8000h base + 10h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	LOCK	-														BYPASS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1



CCM_ANALOG_PLL_USB1*n* field descriptions

Field	Description
31 LOCK	1 - PLL is currently locked. 0 - PLL is not currently locked.
30–17 -	Always set to zero (0).
16 BYPASS	Bypass the PLL.
15–14 BYPASS_CLK_SRC	Determines the bypass source. 0x0 **REF_CLK_24M** — Select the 24MHz oscillator as source. 0x1 **CLK1** — Select the CLK1_N / CLK1_P as source.
13 ENABLE	Enable the PLL clock output.
12 POWER	Powers up the PLL. This bit will be set automatically when USBPHY0 remote wakeup event happens.
11–7 -	Always set to zero (0).
6 EN_USB_CLKS	Powers the 9-phase PLL outputs for USBPHYn. Additionally, the UTMI clock gate must be deasserted in the USBPHYn to enable USBn operation (clear CLKGATE bit in USBPHYn_CTRL). This bit will be set automatically when USBPHYn remote wakeup event occurs. 0 PLL outputs for USBPHYn off. 1 PLL outputs for USBPHYn on.
5–2 -	Always set to zero (0).
1 DIV_SELECT	This field controls the PLL loop divider. 0 - Fout=Fref*20; 1 - Fout=Fref*22.

Table continues on the next page...

CCM_ANALOG_PLL_USB1*n* field descriptions (continued)

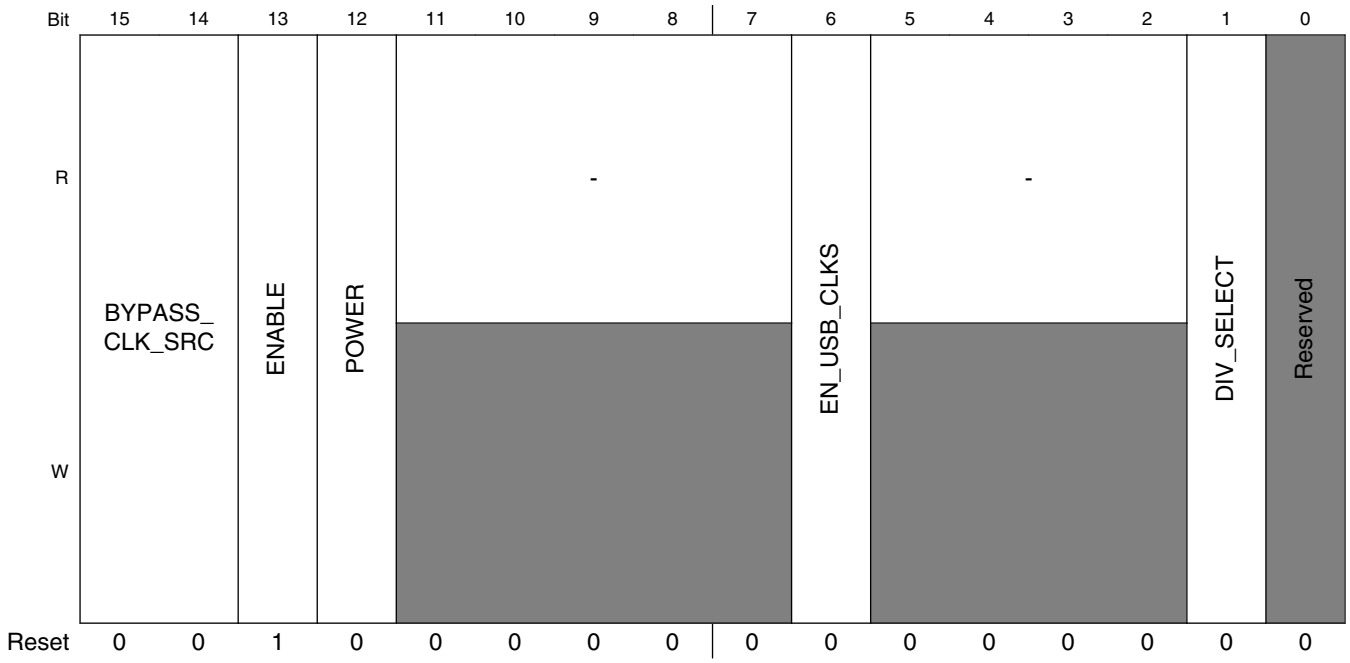
Field	Description
0 -	This field is reserved. Reserved.

14.8.3 Analog USB2 480MHz PLL Control Register (CCM_ANALOG_PLL_USB2*n*)

The control register provides control for USBPHY1 480MHz PLL.

Address: 400D_8000h base + 20h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	LOCK	-														BYPASS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1



CCM_ANALOG_PLL_USB2*n* field descriptions

Field	Description
31 LOCK	1 - PLL is currently locked. 0 - PLL is not currently locked.
30–17 -	Always set to zero (0).
16 BYPASS	Bypass the PLL.
15–14 BYPASS_CLK_ SRC	Determines the bypass source. 0x0 **REF_CLK_24M** — Select the 24MHz oscillator as source. 0x1 **CLK1** — Select the CLK1_N / CLK1_P as source. 0x2 **Reserved1** — 0x3 **Reserved2** —
13 ENABLE	Enable the PLL clock output.
12 POWER	Powers up the PLL. This bit will be set automatically when USBPHY1 remote wakeup event happens.
11–7 -	Always set to zero (0).
6 EN_USB_CLKS	0: 8-phase PLL outputs for USBPHY1 are powered down. If set to 1, 8-phase PLL outputs for USBPHY1 are powered up. Additionally, the utmi clock gate must be deasserted in the USBPHY1 to enable USB0 operation (clear CLKGATE bit in USBPHY1_CTRL).This bit will be set automatically when USBPHY1 remote wakeup event happens.
5–2 -	Always set to zero (0).
1 DIV_SELECT	This field controls the PLL loop divider. 0 - Fout=Fref*20; 1 - Fout=Fref*22.

Table continues on the next page...

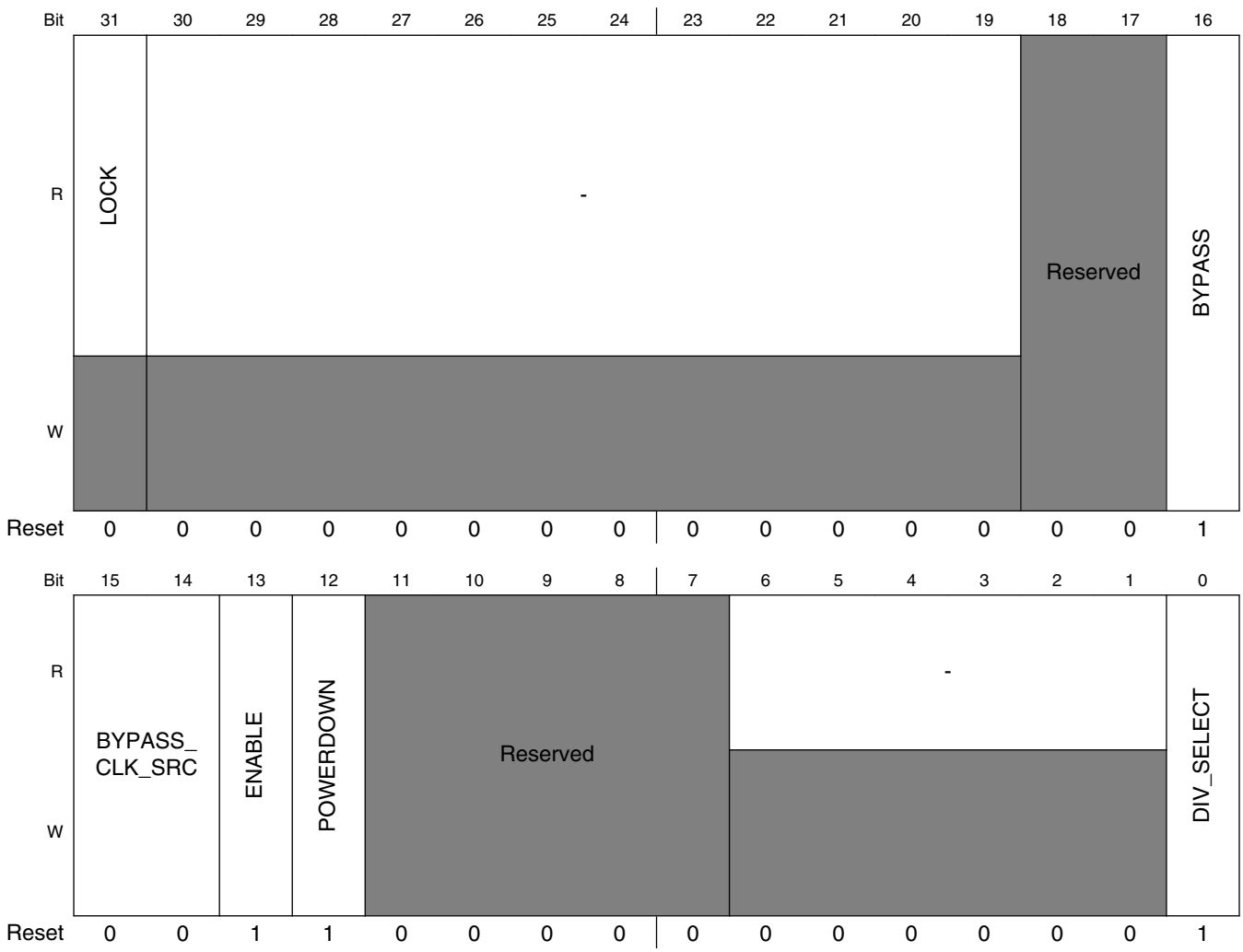
CCM_ANALOG_PLL_USB2*n* field descriptions (continued)

Field	Description
0 -	This field is reserved. Reserved.

14.8.4 Analog System PLL Control Register (CCM_ANALOG_PLL_SYS*n*)

The control register provides control for the 528MHz PLL.

Address: 400D_8000h base + 30h offset + (4d × i), where i=0d to 3d



CCM_ANALOG_PLL_SYS*n* field descriptions

Field	Description
31 LOCK	1 - PLL is currently locked; 0 - PLL is not currently locked.
30–19 -	Always set to zero (0).
18–17 -	This field is reserved. Reserved
16 BYPASS	Bypass the PLL.
15–14 BYPASS_CLK_SRC	Determines the bypass source. 0x0 **REF_CLK_24M** — Select the 24MHz oscillator as source. 0x1 **CLK1** — Select the CLK1_N / CLK1_P as source.
13 ENABLE	Enable PLL output
12 POWERDOWN	Powers down the PLL.
11–7 -	This field is reserved. Reserved.
6–1 -	Always set to zero (0).
0 DIV_SELECT	This field controls the PLL loop divider. 0 - Fout=Fref*20; 1 - Fout=Fref*22.

14.8.5 528MHz System PLL Spread Spectrum Register (CCM_ANALOG_PLL_SYS_SS)

This register contains the 528MHz PLL spread spectrum controls.

Address: 400D_8000h base + 40h offset = 400D_8040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	STOP															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ENABLE	STEP														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CCM_ANALOG_PLL_SYS_SS field descriptions

Field	Description
31–16 STOP	Frequency change = stop/CCM_ANALOG_PLL_SYS_DENOM[B]*24MHz.
15 ENABLE	Enable bit 0 — Spread spectrum modulation disabled 1 — Spread spectrum modulation enabled
STEP	Frequency change step = step/CCM_ANALOG_PLL_SYS_DENOM[B]*24MHz.

14.8.6 Numerator of 528MHz System PLL Fractional Loop Divider Register (CCM_ANALOG_PLL_SYS_NUM)

This register contains the numerator of 528MHz PLL fractional loop divider (signed number).

Absolute value should be less than denominator.

Address: 400D_8000h base + 50h offset = 400D_8050h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-																A															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

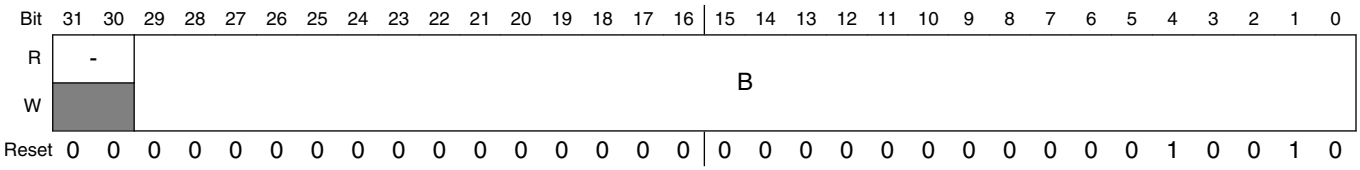
CCM_ANALOG_PLL_SYS_NUM field descriptions

Field	Description
31–30 -	Always set to zero (0).
A	30 bit numerator (A) of fractional loop divider (signed integer).

14.8.7 Denominator of 528MHz System PLL Fractional Loop Divider Register (CCM_ANALOG_PLL_SYS_DENOM)

This register contains the denominator of 528MHz PLL fractional loop divider.

Address: 400D_8000h base + 60h offset = 400D_8060h



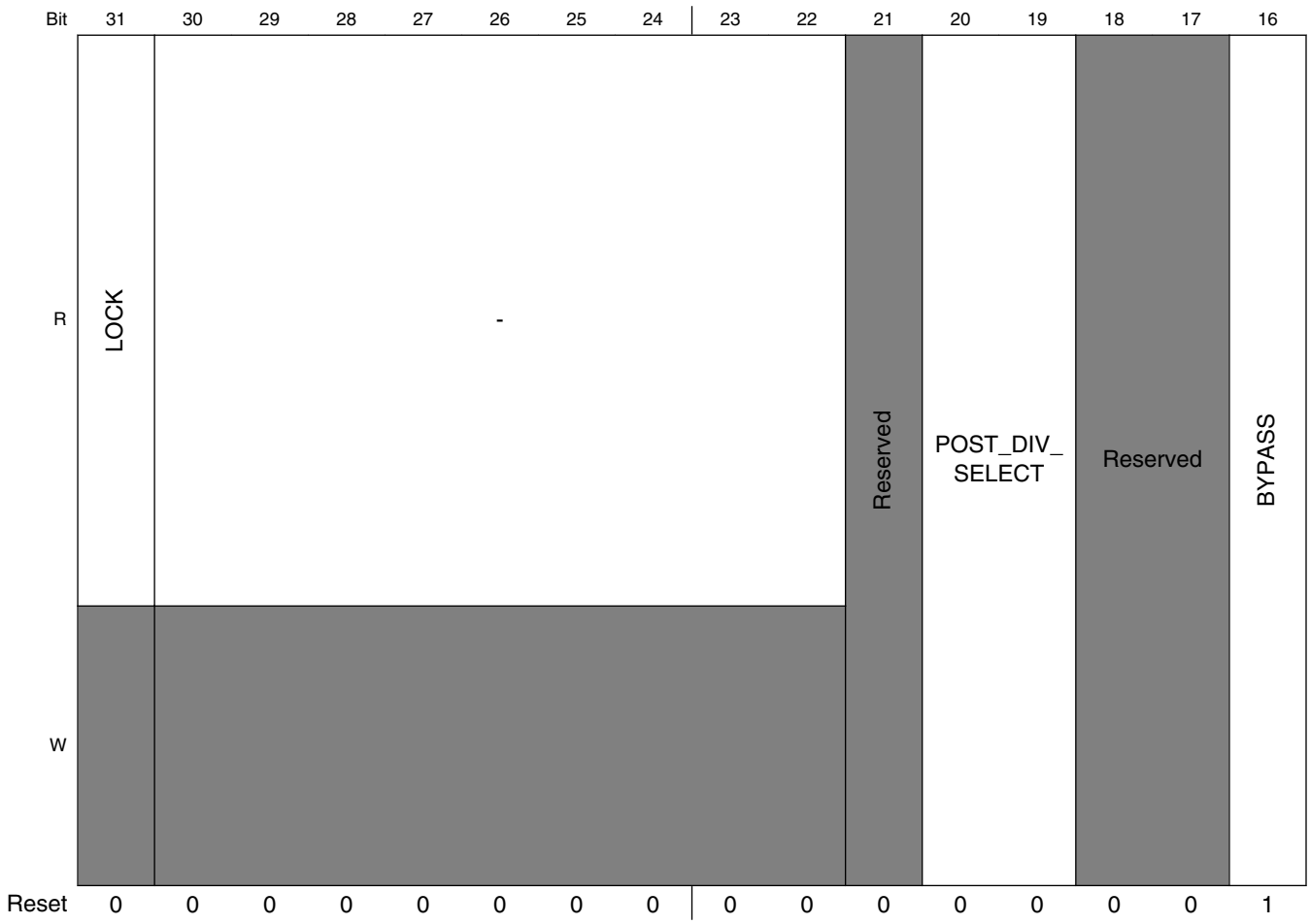
CCM_ANALOG_PLL_SYS_DENOM field descriptions

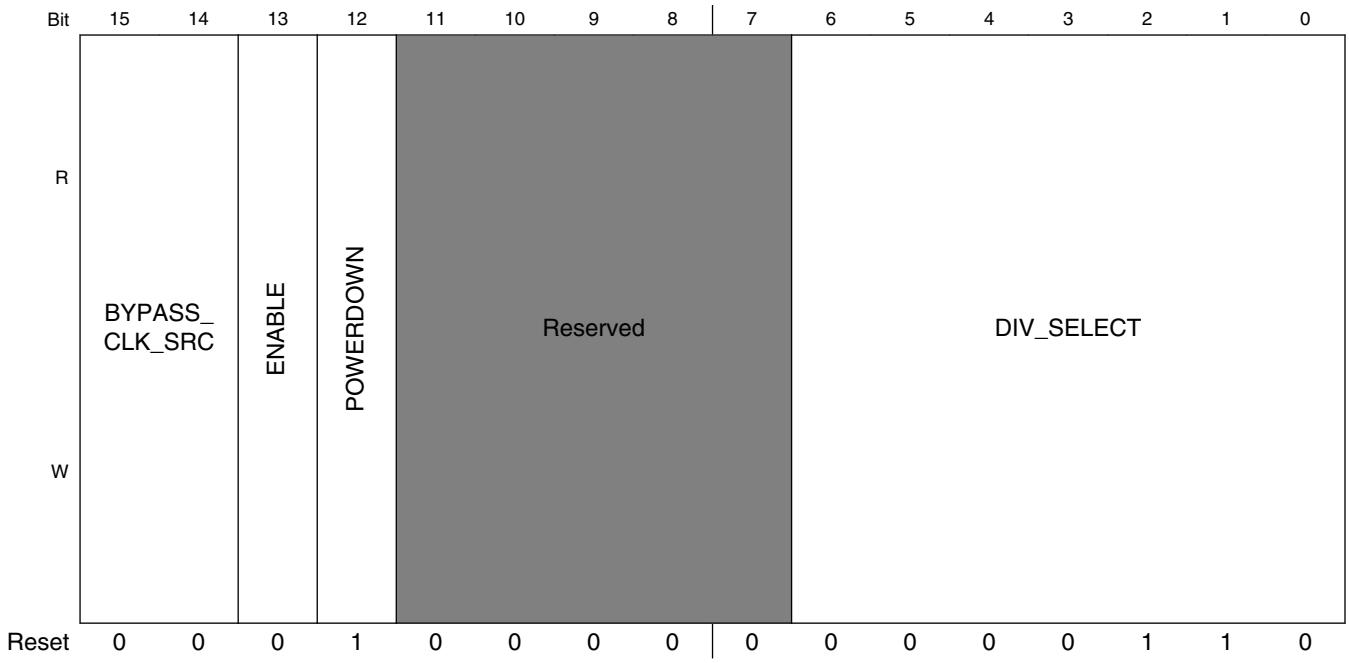
Field	Description
31–30 -	Always set to zero (0).
B	30 bit denominator (B) of fractional loop divider (unsigned integer).

14.8.8 Analog Audio PLL control Register (CCM_ANALOG_PLL_AUDIO*n*)

The control register provides control for the audio PLL.

Address: 400D_8000h base + 70h offset + (4d × i), where i=0d to 3d





CCM_ANALOG_PLL_AUDIO*n* field descriptions

Field	Description
31 LOCK	1 - PLL is currently locked. 0 - PLL is not currently locked.
30–22 -	Always set to zero (0).
21 -	This field is reserved. Reserved
20–19 POST_DIV_ SELECT	These bits implement a divider after the PLL, but before the enable and bypass mux. 00 — Divide by 4. 01 — Divide by 2. 10 — Divide by 1. 11 — Reserved
18–17 -	This field is reserved. Revsered
16 BYPASS	Bypass the PLL.
15–14 BYPASS_CLK_ SRC	Determines the bypass source. 0x0 **REF_CLK_24M** — Select the 24MHz oscillator as source. 0x1 **CLK1** — Select the CLK1_N / CLK1_P as source. 0x2 **Reserved1** — 0x3 **Reserved2** —
13 ENABLE	Enable PLL output
12 POWERDOWN	Powers down the PLL.

Table continues on the next page...

CCM_ANALOG_PLL_AUDIO*n* field descriptions (continued)

Field	Description
11–7 -	This field is reserved. Reserved.
DIV_SELECT	This field controls the PLL loop divider. Valid range for DIV_SELECT divider value: 27~54.

14.8.9 Numerator of Audio PLL Fractional Loop Divider Register (CCM_ANALOG_PLL_AUDIO_NUM)

This register contains the numerator (A) of Audio PLL fractional loop divider (Signed number).

Absolute value should be less than denominator

Address: 400D_8000h base + 80h offset = 400D_8080h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-																A															
W																																
Reset	0	0	0	0	0	1	0	1	1	1	1	1	0	1	0	1	1	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0

CCM_ANALOG_PLL_AUDIO_NUM field descriptions

Field	Description
31–30 -	Always set to zero (0).
A	30 bit numerator of fractional loop divider.

14.8.10 Denominator of Audio PLL Fractional Loop Divider Register (CCM_ANALOG_PLL_AUDIO_DENOM)

This register contains the denominator (B) of Audio PLL fractional loop divider (unsigned number).

Address: 400D_8000h base + 90h offset = 400D_8090h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	-																B															
W																																
Reset	0	0	1	0	1	0	0	1	0	1	1	0	0	1	0	0	0	1	1	0	0	0	0	1	1	0	0	1	1	1	0	0

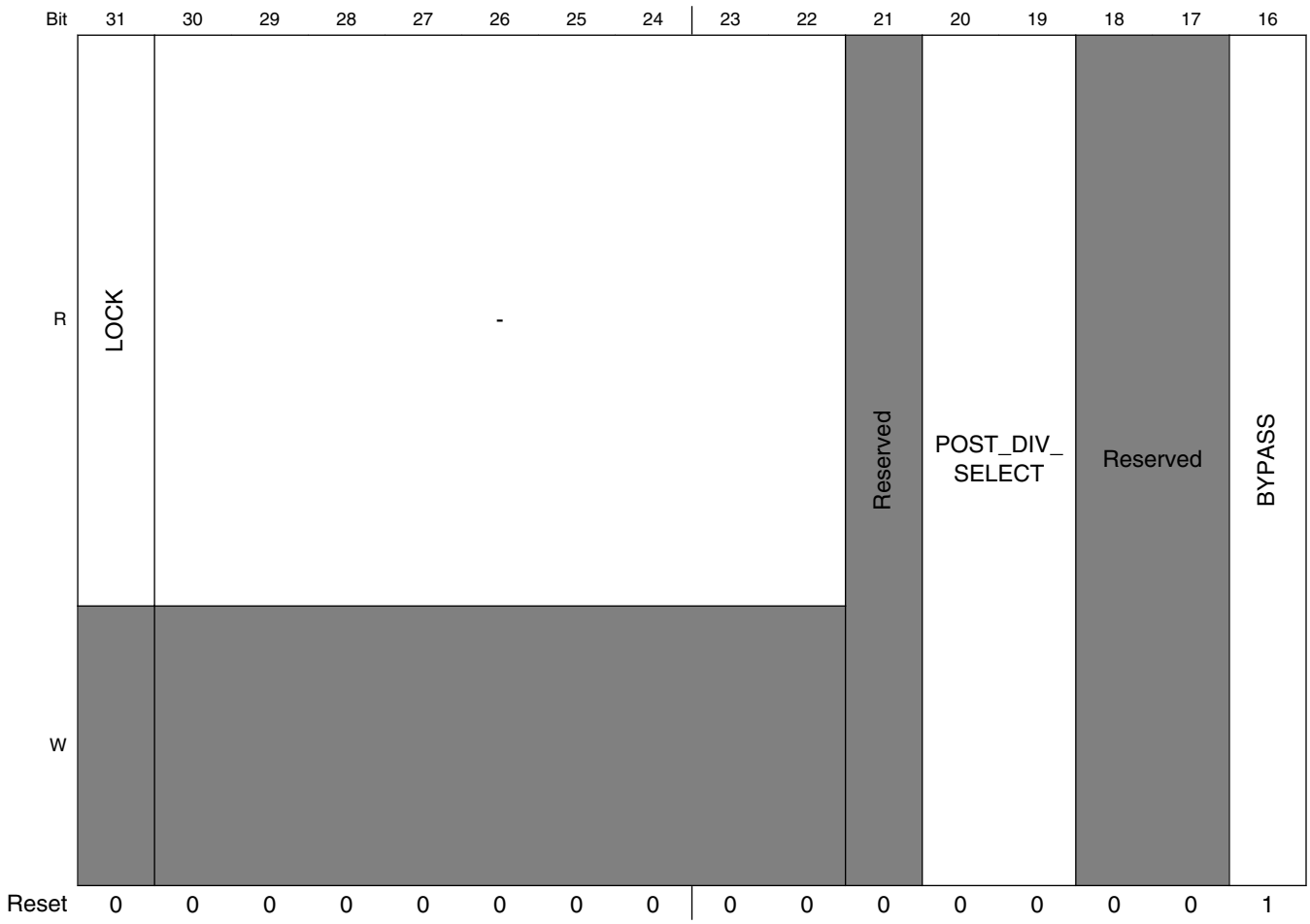
CCM_ANALOG_PLL_AUDIO_DENOM field descriptions

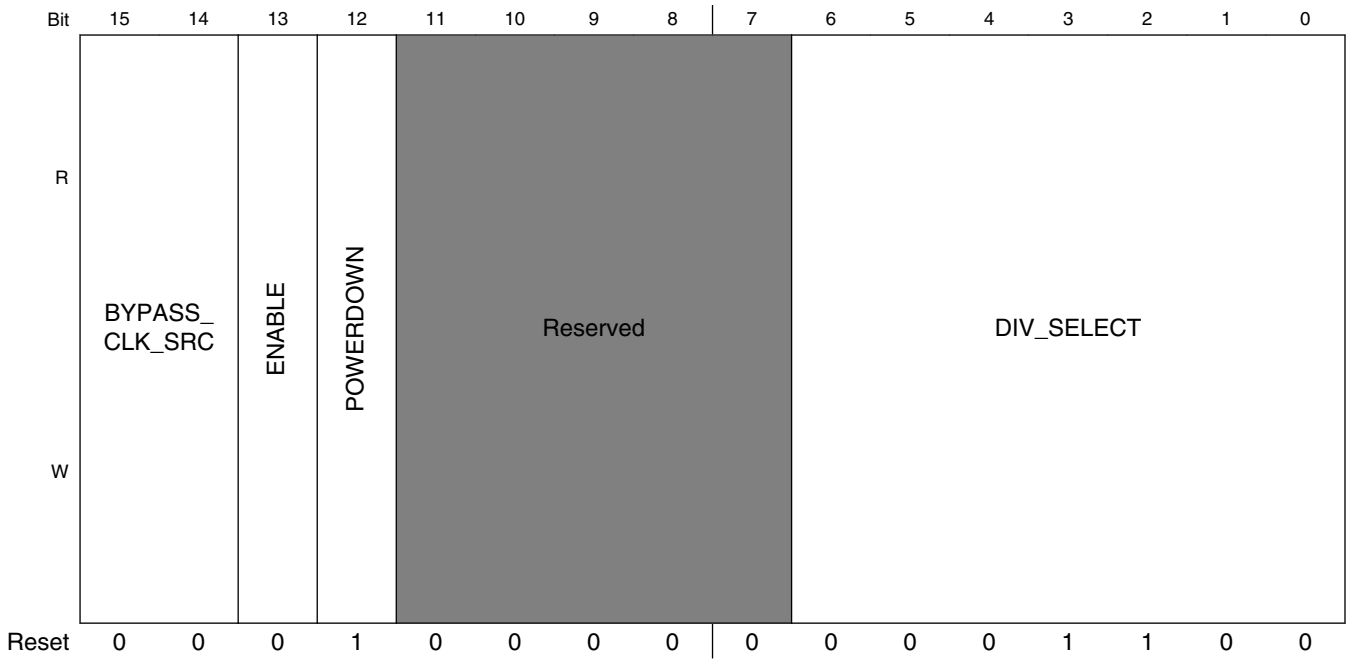
Field	Description
31–30 -	Always set to zero (0).
B	30 bit denominator of fractional loop divider.

14.8.11 Analog Video PLL control Register (CCM_ANALOG_PLL_VIDEO*n*)

The control register provides control for the Video PLL.

Address: 400D_8000h base + A0h offset + (4d × i), where i=0d to 3d





CCM_ANALOG_PLL_VIDEO*n* field descriptions

Field	Description
31 LOCK	1 - PLL is currently locked; 0 - PLL is not currently locked.
30–22 -	Always set to zero (0).
21 -	This field is reserved. Revserved
20–19 POST_DIV_ SELECT	These bits implement a divider after the PLL, but before the enable and bypass mux. 00 — Divide by 4. 01 — Divide by 2. 10 — Divide by 1. 11 — Reserved
18–17 -	This field is reserved. Reserved
16 BYPASS	Bypass the PLL.
15–14 BYPASS_CLK_ SRC	Determines the bypass source. 0x0 **REF_CLK_24M** — Select the 24MHz oscillator as source. 0x1 **CLK1** — Select the CLK1_N / CLK1_P as source. 0x2 **Reserved1** — 0x3 **Reserved2** —
13 ENABLE	Enable PLL output
12 POWERDOWN	Powers down the PLL.

Table continues on the next page...

CCM_ANALOG_PLL_VIDEO*n* field descriptions (continued)

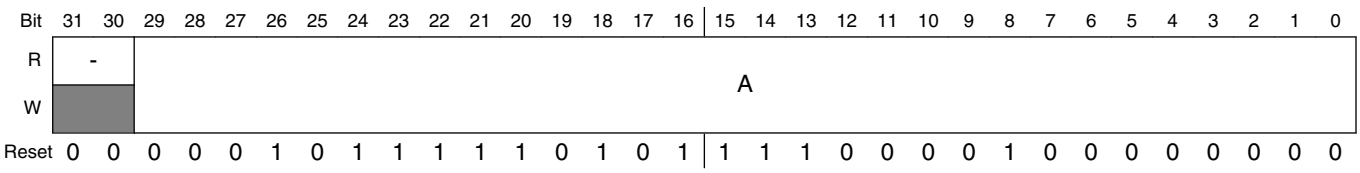
Field	Description
11–7 -	This field is reserved. Reserved.
DIV_SELECT	This field controls the PLL loop divider. Valid range for DIV_SELECT divider value: 27~54.

14.8.12 Numerator of Video PLL Fractional Loop Divider Register (CCM_ANALOG_PLL_VIDEO_NUM)

This register contains the numerator (A) of Video PLL fractional loop divider (signed number).

Absolute value should be less than denominator

Address: 400D_8000h base + B0h offset = 400D_80B0h



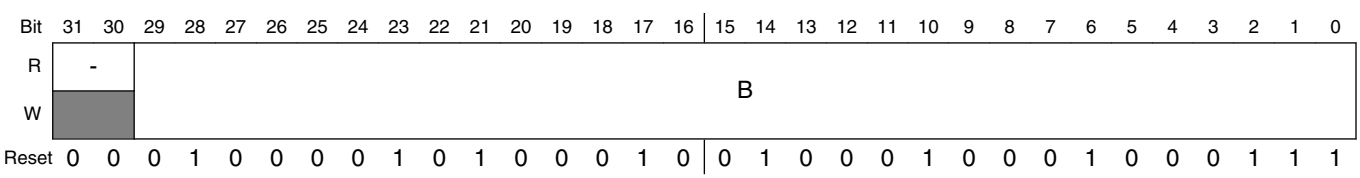
CCM_ANALOG_PLL_VIDEO_NUM field descriptions

Field	Description
31–30 -	Always set to zero (0).
A	30 bit numerator of fractional loop divider (signed number). The absolute value should be less than denominator.

14.8.13 Denominator of Video PLL Fractional Loop Divider Register (CCM_ANALOG_PLL_VIDEO_DENOM)

This register contains the denominator (B) of Video PLL fractional loop divider (unsigned number).

Address: 400D_8000h base + C0h offset = 400D_80C0h



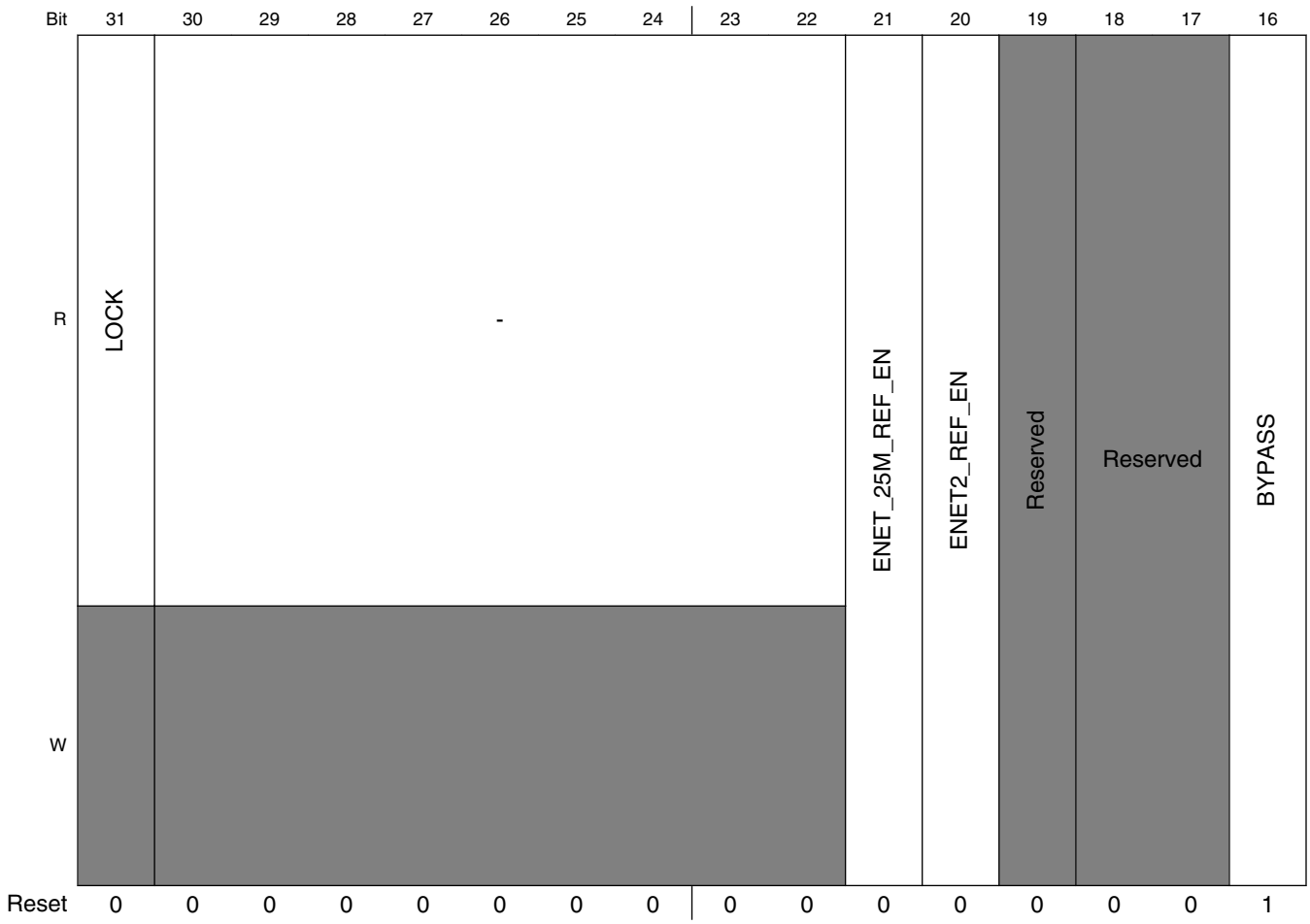
CCM_ANALOG_PLL_VIDEO_DENOM field descriptions

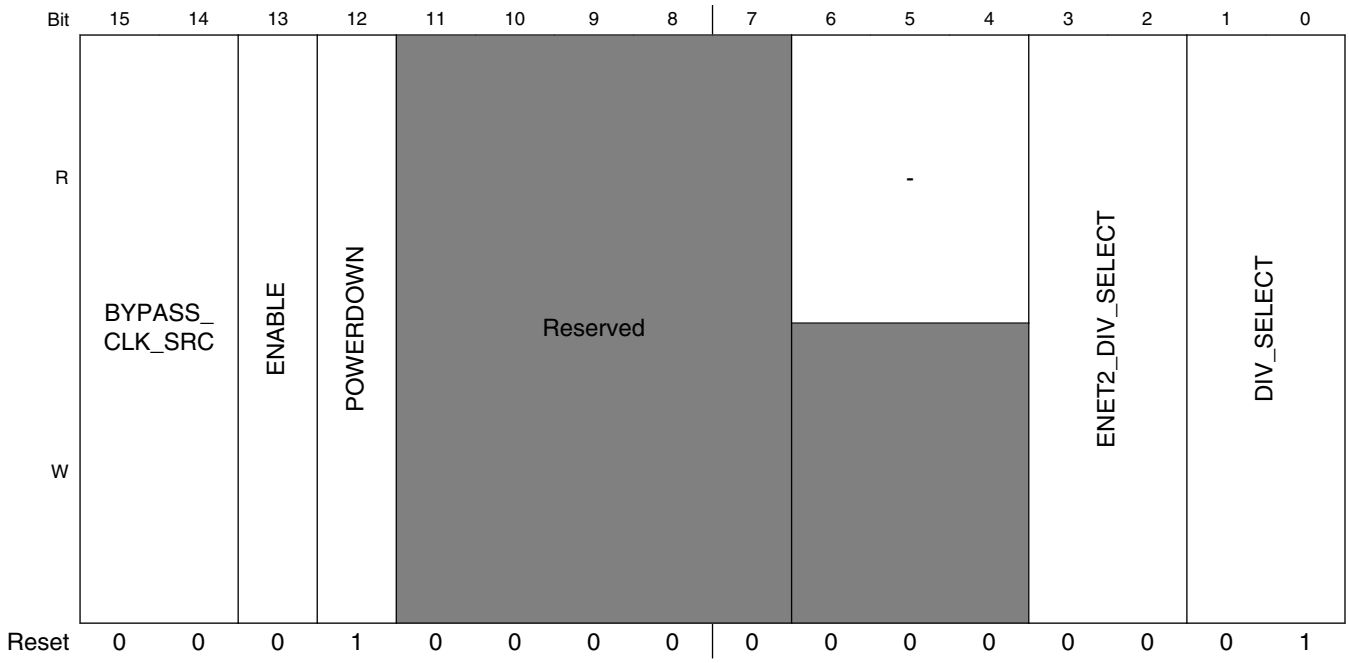
Field	Description
31–30 -	Always set to zero (0).
B	30 bit denominator of fractional loop divider.

14.8.14 Analog ENET PLL Control Register (CCM_ANALOG_PLL_ENET*n*)

The control register provides control for the ENET PLL.

Address: 400D_8000h base + E0h offset + (4d × i), where i=0d to 3d





CCM_ANALOG_PLL_ENET*n* field descriptions

Field	Description
31 LOCK	1 - PLL is currently locked; 0 - PLL is not currently locked.
30–22 -	Always set to zero (0).
21 ENET_25M_ REF_EN	Enable the PLL providing ENET 25 MHz reference clock
20 ENET2_REF_EN	Enable the PLL providing the ENET2 reference clock
19 -	This field is reserved. Reserved
18–17 -	This field is reserved. Reserved
16 BYPASS	Bypass the PLL.
15–14 BYPASS_CLK_ SRC	Determines the bypass source. 0x0 **REF_CLK_24M** — Select the 24MHz oscillator as source. 0x1 **CLK1** — Select the CLK1_N / CLK1_P as source. 0x2 **Reserved1** — 0x3 **Reserved2** —
13 ENABLE	Enable the PLL providing the ENET reference clock.
12 POWERDOWN	Powers down the PLL.
11–7 -	This field is reserved. Reserved.

Table continues on the next page...

CCM_ANALOG_PLL_ENET*n* field descriptions (continued)

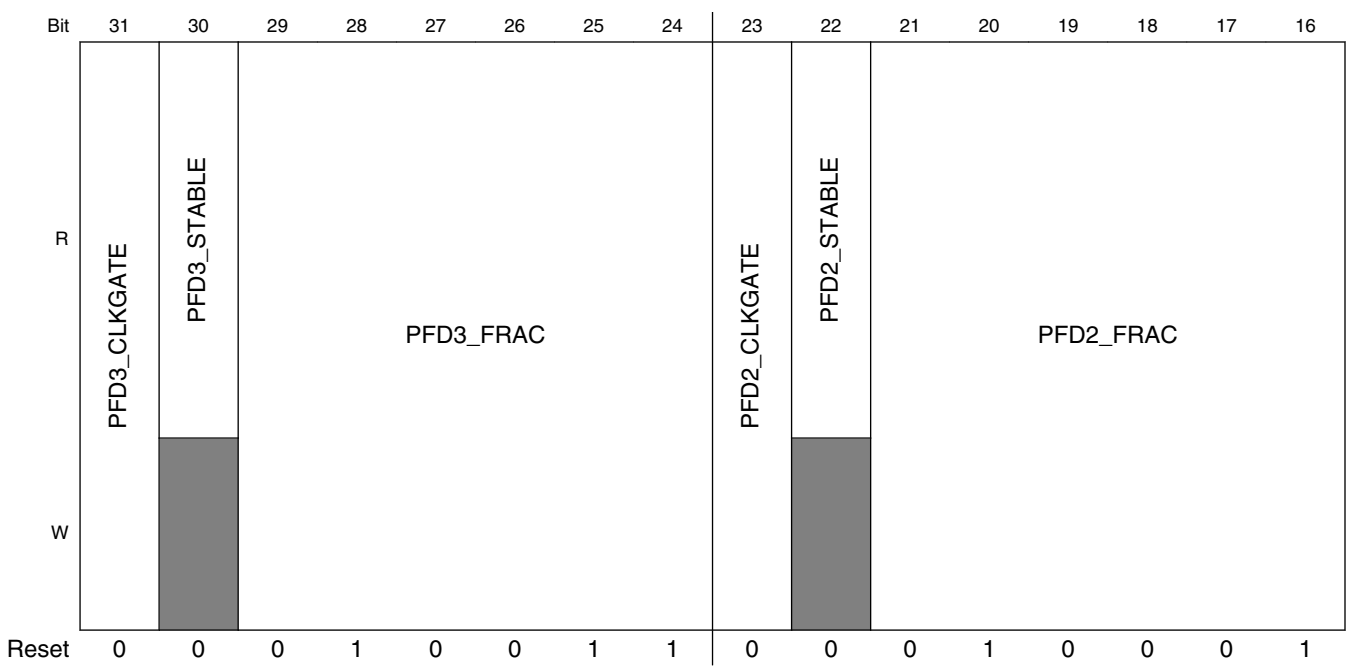
Field	Description
6–4 -	Always set to zero (0).
3–2 ENET2_DIV_SELECT	Controls the frequency of the ENET2 reference clock. 00 25MHz 01 50MHz 10 100MHz (not 50% duty cycle) 11 125MHz
DIV_SELECT	Controls the frequency of the ethernet reference clock.00 - 25MHz; 01 - 50MHz; 10 - 100MHz (not 50% duty cycle); 11 - 125MHz;

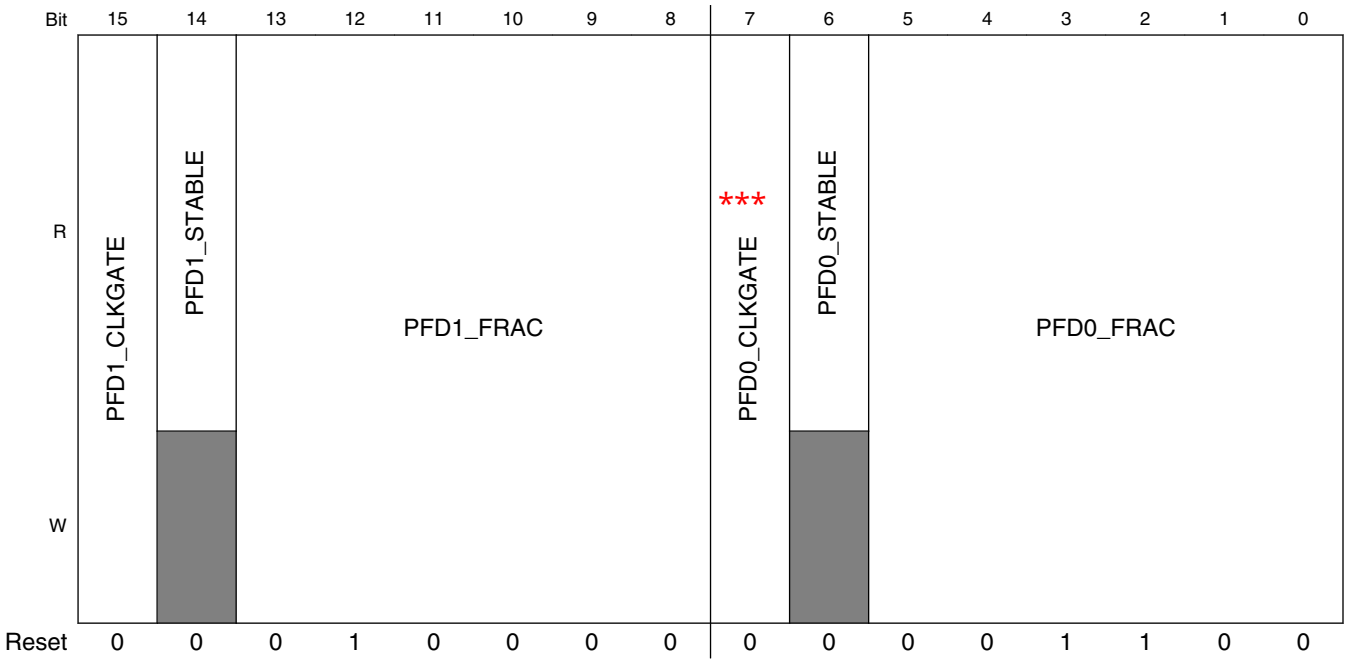
14.8.15 480MHz Clock (PLL3) Phase Fractional Divider Control Register (CCM_ANALOG_PFD_480*n*)

The PFD_480 control register provides control for PFD clock generation.

This register controls the 4-phase fractional clock dividers. The fractional clock frequencies are a product of the values in these registers.

Address: 400D_8000h base + F0h offset + (4d × i), where i=0d to 3d





CCM_ANALOG_PFD_480*n* field descriptions

Field	Description
31 PFD3_CLKGATE	IO Clock Gate. If set to 1, the 3rd fractional divider clock (reference ref_pfd3) is off (power savings). 0: ref_pfd3 fractional divider clock is enabled. Need to assert this bit before PLL is powered down
30 PFD3_STABLE	This read-only bitfield is for DIAGNOSTIC PURPOSES ONLY since the fractional divider should become stable quickly enough that this field will never need to be used by either device driver or application code. The value inverts when the new programmed fractional divide value has taken effect. Read this bit, program the new value, and when this bit inverts, the phase divider clock output is stable. Note that the value will not invert when the fractional divider is taken out of or placed into clock-gated state.
29–24 PFD3_FRAC	This field controls the fractional divide value. The resulting frequency shall be 480*18/PFD3_FRAC where PFD3_FRAC is in the range 12-35.
23 PFD2_CLKGATE	IO Clock Gate. If set to 1, the IO fractional divider clock (reference ref_pfd2) is off (power savings). 0: ref_pfd2 fractional divider clock is enabled. Need to assert this bit before PLL is powered down
22 PFD2_STABLE	This read-only bitfield is for DIAGNOSTIC PURPOSES ONLY since the fractional divider should become stable quickly enough that this field will never need to be used by either device driver or application code. The value inverts when the new programmed fractional divide value has taken effect. Read this bit, program the new value, and when this bit inverts, the phase divider clock output is stable. Note that the value will not invert when the fractional divider is taken out of or placed into clock-gated state.
21–16 PFD2_FRAC	This field controls the fractional divide value. The resulting frequency shall be 480*18/PFD2_FRAC where PFD2_FRAC is in the range 12-35.
15 PFD1_CLKGATE	IO Clock Gate. If set to 1, the IO fractional divider clock (reference ref_pfd1) is off (power savings). 0: ref_pfd1 fractional divider clock is enabled. Need to assert this bit before PLL is powered down
14 PFD1_STABLE	This read-only bitfield is for DIAGNOSTIC PURPOSES ONLY since the fractional divider should become stable quickly enough that this field will never need to be used by either device driver or application code. The value inverts when the new programmed fractional divide value has taken effect. Read this bit,

Table continues on the next page...

CCM_ANALOG_PFD_480*n* field descriptions (continued)

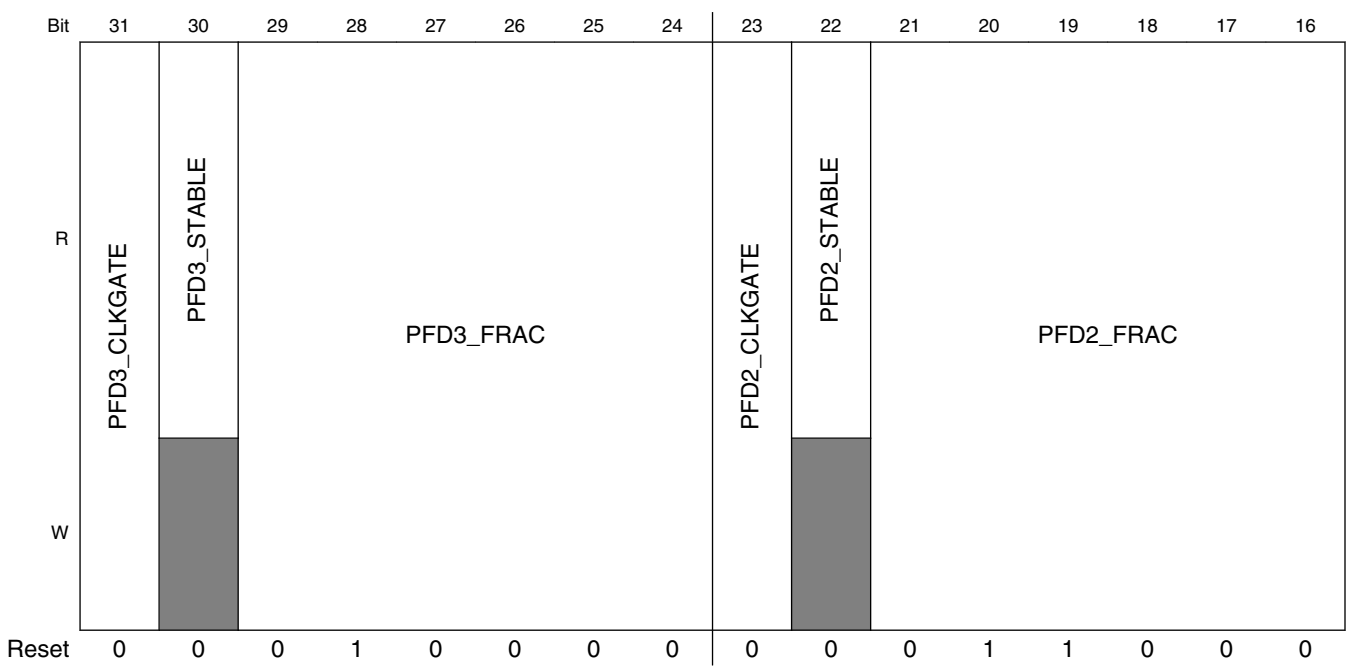
Field	Description
	program the new value, and when this bit inverts, the phase divider clock output is stable. Note that the value will not invert when the fractional divider is taken out of or placed into clock-gated state.
13–8 PFD1_FRAC	This field controls the fractional divide value. The resulting frequency shall be 480*18/PFD1_FRAC where PFD1_FRAC is in the range 12-35.
7 PFD0_CLKGATE	If set to 1, the IO fractional divider clock (reference ref_pfd0) is off (power savings). 0: ref_pfd0 fractional divider clock is enabled. CAUTION: setting this bit may cause problems! Need to assert this bit before PLL is powered down
6 PFD0_STABLE	This read-only bitfield is for DIAGNOSTIC PURPOSES ONLY since the fractional divider should become stable quickly enough that this field will never need to be used by either device driver or application code. The value inverts when the new programmed fractional divide value has taken effect. Read this bit, program the new value, and when this bit inverts, the phase divider clock output is stable. Note that the value will not invert when the fractional divider is taken out of or placed into clock-gated state.
PFD0_FRAC	This field controls the fractional divide value. The resulting frequency shall be 480*18/PFD0_FRAC where PFD0_FRAC is in the range 12-35.

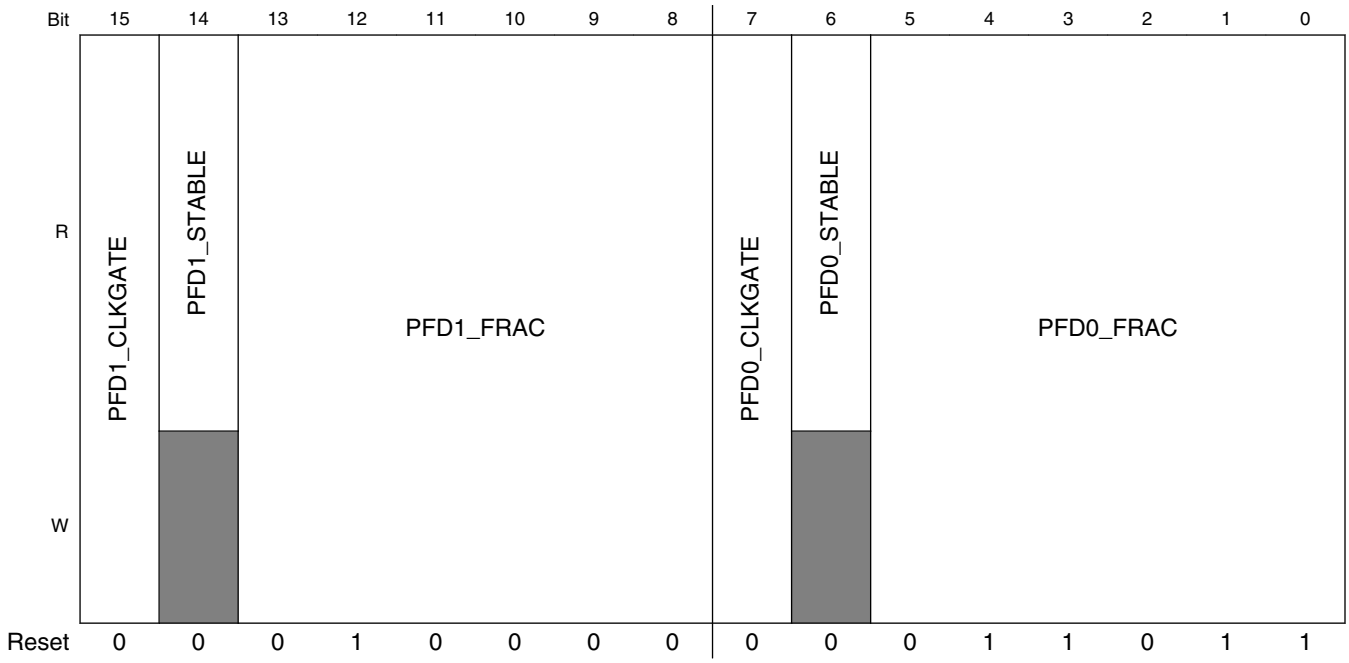
14.8.16 528MHz Clock (PLL2) Phase Fractional Divider Control Register (CCM_ANALOG_PFD_528*n*)

The PFD_528 control register provides control for PFD clock generation.

This register controls the 3-phase fractional clock dividers. The fractional clock frequencies are a product of the values in these registers.

Address: 400D_8000h base + 100h offset + (4d × i), where i=0d to 3d





CCM_ANALOG_PFD_528*n* field descriptions

Field	Description
31 PFD3_CLKGATE	IO Clock Gate. If set to 1, the 3rd fractional divider clock (reference ref_pfd3) is off (power savings). 0: ref_pfd3 fractional divider clock is enabled. Need to assert this bit before PLL powered down
30 PFD3_STABLE	This read-only bitfield is for DIAGNOSTIC PURPOSES ONLY since the fractional divider should become stable quickly enough that this field will never need to be used by either device driver or application code. The value inverts when the new programmed fractional divide value has taken effect. Read this bit, program the new value, and when this bit inverts, the phase divider clock output is stable. Note that the value will not invert when the fractional divider is taken out of or placed into clock-gated state.
29–24 PFD3_FRAC	This field controls the fractional divide value. The resulting frequency shall be 528*18/PFD3_FRAC where PFD3_FRAC is in the range 12-35.
23 PFD2_CLKGATE	IO Clock Gate. If set to 1, the IO fractional divider clock (reference ref_pfd2) is off (power savings). 0: ref_pfd2 fractional divider clock is enabled. Need to assert this bit before PLL powered down
22 PFD2_STABLE	This read-only bitfield is for DIAGNOSTIC PURPOSES ONLY since the fractional divider should become stable quickly enough that this field will never need to be used by either device driver or application code. The value inverts when the new programmed fractional divide value has taken effect. Read this bit, program the new value, and when this bit inverts, the phase divider clock output is stable. Note that the value will not invert when the fractional divider is taken out of or placed into clock-gated state.
21–16 PFD2_FRAC	This field controls the fractional divide value. The resulting frequency shall be 528*18/PFD2_FRAC where PFD2_FRAC is in the range 12-35.
15 PFD1_CLKGATE	IO Clock Gate. If set to 1, the IO fractional divider clock (reference ref_pfd1) is off (power savings). 0: ref_pfd1 fractional divider clock is enabled. Need to assert this bit before PLL powered down
14 PFD1_STABLE	This read-only bitfield is for DIAGNOSTIC PURPOSES ONLY since the fractional divider should become stable quickly enough that this field will never need to be used by either device driver or application code. The value inverts when the new programmed fractional divide value has taken effect. Read this bit,

Table continues on the next page...

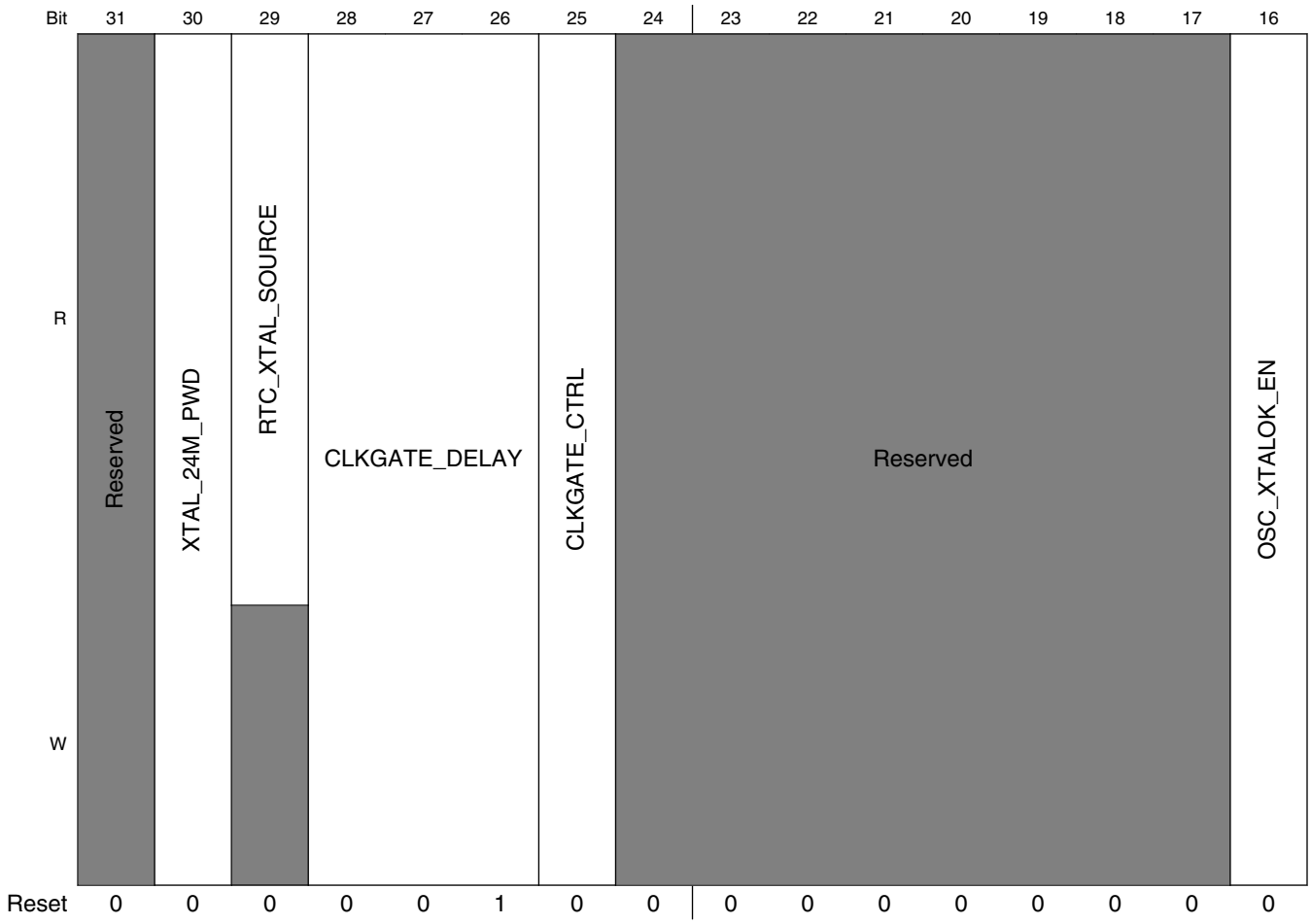
CCM_ANALOG_PFD_528*n* field descriptions (continued)

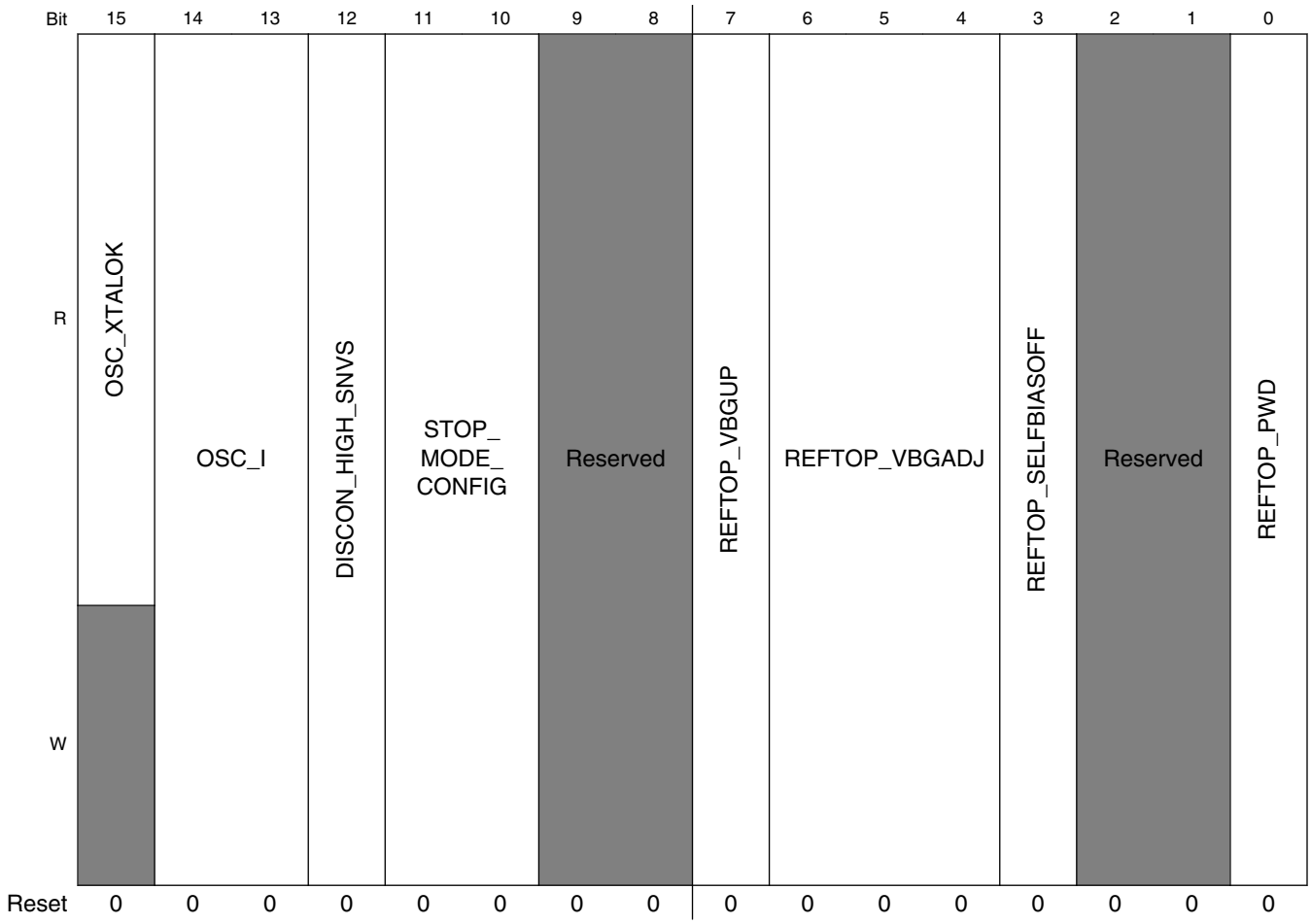
Field	Description
	program the new value, and when this bit inverts, the phase divider clock output is stable. Note that the value will not invert when the fractional divider is taken out of or placed into clock-gated state.
13–8 PFD1_FRAC	This field controls the fractional divide value. The resulting frequency shall be 528*18/PFD1_FRAC where PFD1_FRAC is in the range 12-35.
7 PFD0_CLKGATE	If set to 1, the IO fractional divider clock (reference ref_pfd0) is off (power savings). 0: ref_pfd0 fractional divider clock is enabled. Need to assert this bit before PLL powered down
6 PFD0_STABLE	This read-only bitfield is for DIAGNOSTIC PURPOSES ONLY since the fractional divider should become stable quickly enough that this field will never need to be used by either device driver or application code. The value inverts when the new programmed fractional divide value has taken effect. Read this bit, program the new value, and when this bit inverts, the phase divider clock output is stable. Note that the value will not invert when the fractional divider is taken out of or placed into clock-gated state.
PFD0_FRAC	This field controls the fractional divide value. The resulting frequency shall be 528*18/PFD0_FRAC where PFD0_FRAC is in the range 12-35.

14.8.17 Miscellaneous Register 0 (CCM_ANALOG_MISC0*n*)

This register defines the control and status bits for miscellaneous analog blocks.

Address: 400D_8000h base + 150h offset + (4d × i), where i=0d to 3d





CCM_ANALOG_MISC0*n* field descriptions

Field	Description
31 -	This field is reserved.
30 XTAL_24M_PWD	This field powers down the 24M crystal oscillator if set true. **NOTE:** Not related to CCM. See Crystal Oscillator (XTALOSC)
29 RTC_XTAL_ SOURCE	This field indicates which chip source is being used for the rtc clock. **NOTE:** Not related to CCM. See Crystal Oscillator (XTALOSC) 0 Internal ring oscillator 1 RTC_XTAL
28–26 CLKGATE_ DELAY	This field specifies the delay between powering up the XTAL 24MHz clock and releasing the clock to the digital logic inside the analog block. **NOTE:** Do not change the field during a low power event. This is not a field that the user would normally need to modify. **NOTE:** Not related to CCM. See Crystal Oscillator (XTALOSC) 000 0.5ms

Table continues on the next page...

CCM_ANALOG_MISC0*n* field descriptions (continued)

Field	Description
	001 1.0ms 010 2.0ms 011 3.0ms 100 4.0ms 101 5.0ms 110 6.0ms 111 7.0ms
25 CLKGATE_CTRL	This bit allows disabling the clock gate (always ungated) for the xtal 24MHz clock that clocks the digital logic in the analog block. **NOTE:** Do not change the field during a low power event. This is not a field that the user would normally need to modify. **NOTE:** Not related to CCM. See Crystal Oscillator (XTALOSC) 0 **ALLOW_AUTO_GATE** — Allow the logic to automatically gate the clock when the XTAL is powered down. 1 **NO_AUTO_GATE** — Prevent the logic from ever gating off the clock.
24–17 -	This field is reserved. Always set to zero.
16 OSC_XTALOK_EN	This bit enables the detector that signals when the 24MHz crystal oscillator is stable. **NOTE:** Not related to CCM. See Crystal Oscillator (XTALOSC)
15 OSC_XTALOK	Status bit that signals that the output of the 24-MHz crystal oscillator is stable. Generated from a timer and active detection of the actual frequency. **NOTE:** Not related to CCM. See Crystal Oscillator (XTALOSC)
14–13 OSC_I	This field determines the bias current in the 24MHz oscillator. The aim is to start up with the highest bias current, which can be decreased after startup if it is determined to be acceptable. **NOTE:** Not related to CCM. See Crystal Oscillator (XTALOSC) 00 **NOMINAL** — Nominal 01 **MINUS_12_5_PERCENT** — Decrease current by 12.5% 10 **MINUS_25_PERCENT** — Decrease current by 25.0% 11 **MINUS_37_5_PERCENT** — Decrease current by 37.5%
12 DISCON_HIGH_SNVS	This bit controls a switch from VDD_HIGH_IN to VDD_SNVS_IN. 0 Turn on the switch 1 Turn off the switch
11–10 STOP_MODE_CONFIG	Configure the analog behavior in stop mode. 00 All analog except RTC powered down on stop mode assertion. 01 Beside RTC, analog bandgap, 1p1 and 2p5 regulators are also on. 10 Beside RTC, 1p1 and 2p5 regulators are also on, low-power bandgap is selected so that the normal analog bandgap together with the rest analog is powered down. 11 Beside RTC, low-power bandgap is selected and the rest analog is powered down.

Table continues on the next page...

CCM_ANALOG_MISC0*n* field descriptions (continued)

Field	Description
9–8 - Reserved	This field is reserved.
7 REFTOP_ VBGUP	Status bit that signals the analog bandgap voltage is up and stable. 1 - Stable. **NOTE:** Not related to CCM. See Power Management Unit (PMU)
6–4 REFTOP_ VBGADJ	**NOTE:** Not related to CCM. See Power Management Unit (PMU) 000 Nominal VBG 001 VBG+0.78% 010 VBG+1.56% 011 VBG+2.34% 100 VBG-0.78% 101 VBG-1.56% 110 VBG-2.34% 111 VBG-3.12%
3 REFTOP_ SELFBIASOFF	Control bit to disable the self-bias circuit in the analog bandgap. The self-bias circuit is used by the bandgap during startup. This bit should be set after the bandgap has stabilized and is necessary for best noise performance of analog blocks using the outputs of the bandgap. **NOTE:** Value should be returned to zero before removing vddhigh_in or asserting bit 0 of this register (REFTOP_PWD) to assure proper restart of the circuit. **NOTE:** Not related to CCM. See Power Management Unit (PMU) 0 Uses coarse bias currents for startup 1 Uses bandgap-based bias currents for best performance.
2–1 - Reserved	This field is reserved.
0 REFTOP_PWD	Control bit to power-down the analog bandgap reference circuitry. **NOTE:** A note of caution, the bandgap is necessary for correct operation of most of the LDO, PLL, and other analog functions on the die. **NOTE:** Not related to CCM. See Power Management Unit (PMU)

14.8.18 Miscellaneous Register 1 (CCM_ANALOG_MISC1*n*)

This register defines the control and status bits for miscellaneous analog blocks. The LVDS1 and LVDS2 controls below control the behavior of the anaclk1/1b LVDS IO.

Address: 400D_8000h base + 160h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	IRQ_DIG_BO	IRQ_ANA_BO	IRQ_TEMPHIGH	IRQ_TEMPLOW	IRQ_TEMPPANIC	Reserved									PFD_528_AUTOGATE_EN	PFD_480_AUTOGATE_EN
W	w1c	w1c	w1c	w1c	w1c											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved		Reserved	LVDSCLK1_IBEN	Reserved	LVDSCLK1_OBEN	Reserved					LVDS1_CLK_SEL				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CCM_ANALOG_MISC1*n* field descriptions

Field	Description
31 IRQ_DIG_BO	This status bit is set to one when when any of the digital regulator brownout interrupts assert. Check the regulator status bits to discover which regulator interrupt asserted. **NOTE:** Not related to CCM. See Power Management Unit (PMU)
30 IRQ_ANA_BO	This status bit is set to one when when any of the analog regulator brownout interrupts assert. Check the regulator status bits to discover which regulator interrupt asserted. **NOTE:** Not related to CCM. See Power Management Unit (PMU)
29 IRQ_TEMPHIGH	This status bit is set to one when the temperature sensor high interrupt asserts for high temperature. **NOTE:** Not related to CCM. See Temperature Monitor (TEMPMON)
28 IRQ_TEMPLOW	This status bit is set to one when the temperature sensor low interrupt asserts for low temperature. **NOTE:** Not related to CCM. See Temperature Monitor (TEMPMON)
27 IRQ_ TEMPPANIC	This status bit is set to one when the temperature sensor panic interrupt asserts for a panic high temperature. **NOTE:** Not related to CCM. See Temperature Monitor (TEMPMON)
26–18 -	This field is reserved. Reserved
17 PFD_528_ AUTOGATE_EN	This enables a feature that will clkgate (reset) all PFD_528 clocks anytime the PLL_528 is unlocked or powered off.
16 PFD_480_ AUTOGATE_EN	This enables a feature that will clkgate (reset) all PFD_480 clocks anytime the USB1_PLL_480 is unlocked or powered off.
15–14 -	This field is reserved. Reserved
13 -	This field is reserved. Reserved
12 LVDSCLK1_ IBEN	This enables the LVDS input buffer for anaclk1/1b. Do not enable input and output buffers simultaneously.
11 -	This field is reserved. Reserved
10 LVDSCLK1_ OBEN	This enables the LVDS output buffer for anaclk1/1b. Do not enable input and output buffers simultaneously.
9–5 -	This field is reserved. Reserved
LVDS1_CLK_ SEL	This field selects the clk to be routed to anaclk1/1b. 00000 **ARM_PLL** — Arm PLL 00001 **SYS_PLL** — System PLL 00010 **PFD4** — ref_pfd4_clk == pll2_pfd0_clk

Table continues on the next page...

CCM_ANALOG_MISC1*n* field descriptions (continued)

Field	Description	
	00011	**PFD5** — ref_pfd5_clk == pll2_pfd1_clk
	00100	**PFD6** — ref_pfd6_clk == pll2_pfd2_clk
	00101	**PFD7** — ref_pfd7_clk == pll2_pfd3_clk
	00110	**AUDIO_PLL** — Audio PLL
	00111	**VIDEO_PLL** — Video PLL
	01001	**ETHERNET_REF** — ethernet ref clock (ENET_PLL)
	01100	**USB1_PLL** — USB1 PLL clock
	01101	**USB2_PLL** — USB2 PLL clock
	01110	**PFD0** — ref_pfd0_clk == pll3_pfd0_clk
	01111	**PFD1** — ref_pfd1_clk == pll3_pfd1_clk
	10000	**PFD2** — ref_pfd2_clk == pll3_pfd2_clk
	10001	**PFD3** — ref_pfd3_clk == pll3_pfd3_clk
	10010	**XTAL** — xtal (24M)
	10101 to 11111	ref_pfd7_clk == pll2_pfd3_clk

14.8.19 Miscellaneous Register 2 (CCM_ANALOG_MISC2*n*)

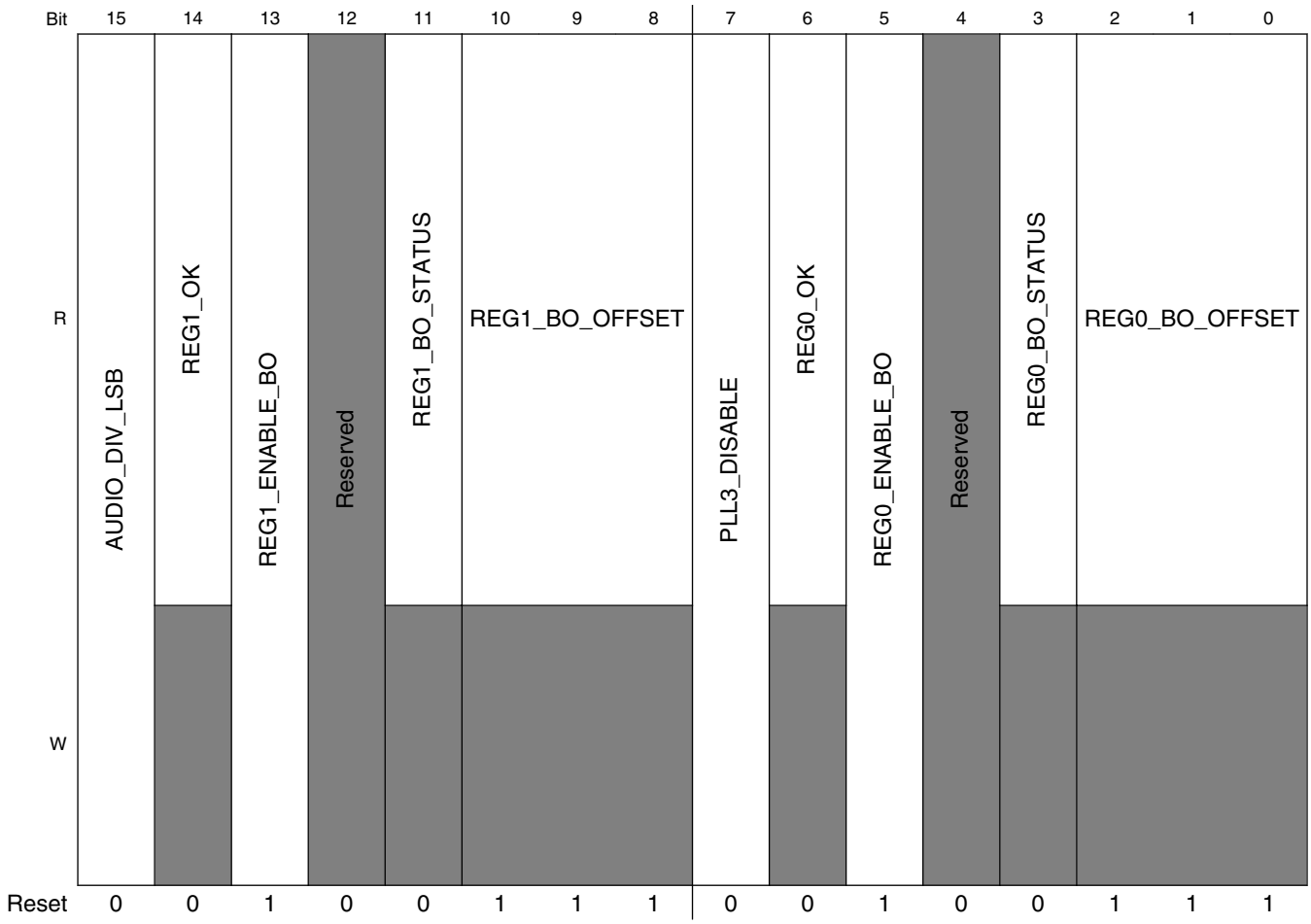
This register defines the control for miscellaneous analog blocks.

NOTE

This register is shared with PMU.

Address: 400D_8000h base + 170h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	VIDEO_DIV		REG2_STEP_TIME		REG1_STEP_TIME		REG0_STEP_TIME		AUDIO_DIV_MSB	REG2_OK	REG2_ENABLE_BO	Reserved	REG2_BO_STATUS	REG2_BO_OFFSET		
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1



CCM_ANALOG_MISC2*n* field descriptions

Field	Description
31–30 VIDEO_DIV	Post-divider for video. The output clock of the video PLL should be gated prior to changing this divider to prevent glitches. This divider is feed by PLL_VIDEOn[POST_DIV_SELECT] to achieve division ratios of /1, /2, /4. 00 divide by 1 (Default) 01 divide by 2 10 divide by 1 11 divide by 4
29–28 REG2_STEP_ TIME	Number of clock periods (24MHz clock). **NOTE:** Not related to CCM. See Power Management Unit (PMU) 00 **64_CLOCKS** — 64 01 **128_CLOCKS** — 128 10 **256_CLOCKS** — 256 11 **512_CLOCKS** — 512
27–26 REG1_STEP_ TIME	Number of clock periods (24MHz clock).

Table continues on the next page...

CCM_ANALOG_MISC2*n* field descriptions (continued)

Field	Description
	NOTE: Not related to CCM. See Power Management Unit (PMU) 00 **64_CLOCKS** — 64 01 **128_CLOCKS** — 128 10 **256_CLOCKS** — 256 11 **512_CLOCKS** — 512
25–24 REG0_STEP_TIME	Number of clock periods (24MHz clock). **NOTE:** Not related to CCM. See Power Management Unit (PMU) 00 **64_CLOCKS** — 64 01 **128_CLOCKS** — 128 10 **256_CLOCKS** — 256 11 **512_CLOCKS** — 512
23 AUDIO_DIV_MSB	MSB of Post-divider for Audio PLL. The output clock of the video PLL should be gated prior to changing this divider to prevent glitches. This divider is feed by PLL_AUDIOn[POST_DIV_SELECT] to achieve division ratios of /1, /2, /4. **NOTE:** MSB bit value pertains to the first bit, please program the LSB bit (bit 15) as well to change divider value for more information. 00 divide by 1 (Default) 01 divide by 2 10 divide by 1 11 divide by 4
22 REG2_OK	Signals that the voltage is above the brownout level for the SOC supply. 1 = regulator output > brownout_target **NOTE:** Not related to CCM. See Power Management Unit (PMU)
21 REG2_ENABLE_BO	Enables the brownout detection. **NOTE:** Not related to CCM. See Power Management Unit (PMU)
20 -	This field is reserved.
19 REG2_BO_STATUS	Reg2 brownout status bit. **NOTE:** Not related to CCM. See Power Management Unit (PMU)
18–16 REG2_BO_OFFSET	This field defines the brown out voltage offset for the xPU power domain. IRQ_DIG_BO is also asserted. Single-bit increments reflect 25mV brownout voltage steps. The reset brown-offset is 175mV below the programmed target code. Brownout target = OUTPUT_TRG - BO_OFFSET. Some steps may be irrelevant because of input supply limitations or load operation. **NOTE:** Not related to CCM. See Power Management Unit (PMU) 100 Brownout offset = 0.100V 111 Brownout offset = 0.175V

Table continues on the next page...

CCM_ANALOG_MISC2*n* field descriptions (continued)

Field	Description
15 AUDIO_DIV_LSB	LSB of Post-divider for Audio PLL. The output clock of the video PLL should be gated prior to changing this divider to prevent glitches. This divider is feed by PLL_AUDIOn[POST_DIV_SELECT] to achieve division ratios of /1, /2, /4. **NOTE:** LSB bit value pertains to the last bit, please program the MSB bit (bit 23) as well, to change divider value for more information. 00 divide by 1 (Default) 01 divide by 2 10 divide by 1 11 divide by 4
14 REG1_OK	GPU/VPU supply **NOTE:** Not related to CCM. See Power Management Unit (PMU)
13 REG1_ENABLE_BO	Enables the brownout detection. **NOTE:** Not related to CCM. See Power Management Unit (PMU)
12 -	This field is reserved.
11 REG1_BO_STATUS	Reg1 brownout status bit. **NOTE:** Not related to CCM. See Power Management Unit (PMU) 1 Brownout, supply is below target minus brownout offset.
10–8 REG1_BO_OFFSET	This field defines the brown out voltage offset for the xPU power domain. IRQ_DIG_BO is also asserted. Single-bit increments reflect 25mV brownout voltage steps. The reset brown-offset is 175mV below the programmed target code. Brownout target = OUTPUT_TRG - BO_OFFSET. Some steps may be irrelevant because of input supply limitations or load operation. **NOTE:** Not related to CCM. See Power Management Unit (PMU) 100 Brownout offset = 0.100V 111 Brownout offset = 0.175V
7 PLL3_DISABLE	When USB is in low power suspend mode this Control bit is used to indicate if other system peripherals require the USB PLL3 clock when the SoC is not in low power mode. A user needs to set this bit if they want to optionally disable PLL3 while the SoC is not in any low power mode to save power. When the system does go into low power mode this bit setting would not have any affect. **NOTE:** When USB is in low power suspend mode users would need to ensure PLL3 is not being used before setting this bit in RUN mode. Please refer to the correct PLL disabling procedure in Disabling / Enabling PLLs 0 PLL3 is being used by peripherals and is enabled when SoC is not in any low power mode 1 PLL3 can be disabled when the SoC is not in any low power mode
6 REG0_OK	Arm supply **NOTE:** Not related to CCM. See Power Management Unit (PMU)

Table continues on the next page...

CCM_ANALOG_MISC2*n* field descriptions (continued)

Field	Description
5 REG0_ENABLE_BO	Enables the brownout detection. **NOTE:** Not related to CCM. See Power Management Unit (PMU)
4 -	This field is reserved.
3 REG0_BO_STATUS	Reg0 brownout status bit. **NOTE:** Not related to CCM. See Power Management Unit (PMU) 1 Brownout, supply is below target minus brownout offset.
REG0_BO_OFFSET	This field defines the brown out voltage offset for the CORE power domain. IRQ_DIG_BO is also asserted. Single-bit increments reflect 25mV brownout voltage steps. Some steps may be irrelevant because of input supply limitations or load operation. **NOTE:** Not related to CCM. See Power Management Unit (PMU) 100 Brownout offset = 0.100V 111 Brownout offset = 0.175V

Chapter 15
Crystal Oscillator (XTALOSC)

15.1 Chip-specific XTALOSC information

Table 15-1. Reference links to related information

Topic	Related module(s)	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
Interrupts, DMA Events and XBAR Assignments	-	Interrupts, DMA Events and XBAR Assignments

15.2 Overview

This block comprises both the 24 MHz and 32 kHz implementation of a biased amplifier that when combined with a suitable external quartz crystal and external load capacitors, implements an oscillator.

The block includes means to:

- Accept an external clock source.
- Detect if the crystal frequency is close to 24 MHz or 32 kHz.
- Reduce the operating current via software after the oscillator has started (24 MHz specific feature)
- Supply another ~32 kHz clock source based off an independent internal oscillator if there is no oscillation sensed on the RTC_XTAL bumps(contacts) (32 kHz specific

feature). The internal oscillator will provide clocks to the same on-chip modules as the external 32 kHz oscillator.

- Automatically switch to the external oscillation source when sensed on the RTC_XTAL bumps(contacts) (32 kHz specific feature).

15.3 External Signals

The table found here describes the external signals of XTALOSC:

Table 15-2. XTALOSC External Signals

Signal	Description	Pad	Mode	Direction
REF_CLK_32K	32 kHz reference clock	GPIO_AD_B0_00	ALT2	O
REF_CLK_24M	24 MHz reference clock	GPIO_AD_B0_01	ALT2	O
		GPIO_AD_B0_03 pin 0	ALT6	
		GPIO_AD_B0_13 pin 25	ALT7	
XTALI	Crystal oscillator input signal	XTALI	No Muxing	O
XTALO	Crystal oscillator output signal	XTALO	No Muxing	O

15.4 Crystal Oscillator 24 MHz

15.4.1 Oscillator Configuration (24 MHz)

The basic block diagram of the 24 MHz module configured as a crystal oscillator is shown below.

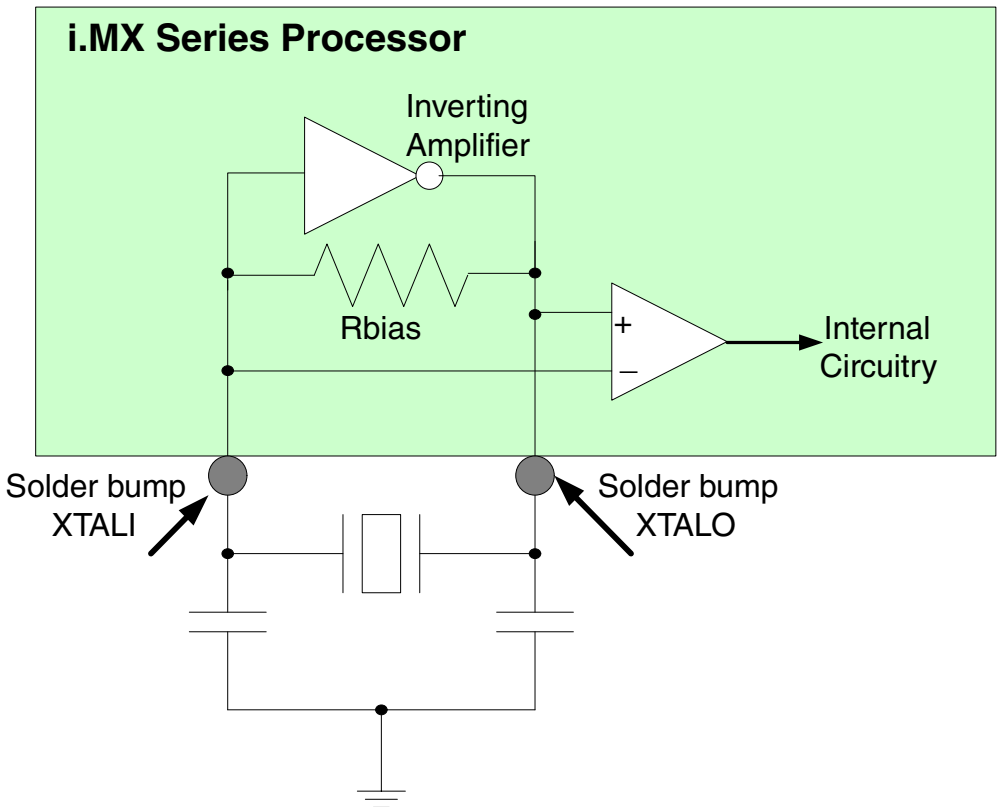


Figure 15-1. Oscillator Configuration (24 MHz)

This integrated biased amplifier can be used to create different frequency oscillators with different external component selection. However, care should be taken as many of the serial IO modules depend on the fixed frequency of 24 MHz. Please consult the sections of the document pertaining to the USB, ENET interfaces, for example. After a healthy oscillation is established, then the bias current of the oscillator can generally be reduced to save power. This is accomplished through the XTALOSC24M_MISC0[OSC_I] bits, defined in the MISC0 register later in this chapter. Restore the XTALOSC24M_MISC0[OSC_I] bits before going into a power mode where the XTALOSC24 is powered down or oscillator startup may become an issue. The power down of the XTALOSC24 module is controlled by the CCM. See this section of the manual for more details.

15.4.2 Bypass Configuration (24 MHz)

If it is desired to drive the chip with an external clock source, then the 24 MHz oscillator could be driven in one of three configurations using a nominal 1.1V source.

1. A single ended external clock source can be used to overdrive the output of the amplifier (XTALO). Since the oscillation sensing amplifier is differential, the

XTALI pin should be externally floating and capacitively loaded. The combination of the internal biasing resistor and the external capacitor will filter the signal applied to the XTALO pin and develop a rough reference for the sensing amplifier to compare to.

2. A single ended external clock source can be used to drive XTALI. In this configuration, XTALO should be left externally floating.
3. A differential external clock source can be used to drive both XTALI and XTALO.

Generally, configuration 2 is anticipated to be the most used configuration, but all three configurations may be utilized.

15.4.3 RC Oscillator (24 MHz)

A lower-power RC oscillator module is available on-chip as a possible alternative to the 24 MHz crystal oscillator after a successful power-up sequence.

The 24 MHz RC oscillator is a self-tuning circuit that will output the programmed frequency value by using the RTC clock as it's reference. This oscillator is intended for normal operation and not fast boot.

While the power consumption of this RC oscillator is much lower than the 24 MHz crystal oscillator, one limitation of this RC oscillator module is that its clock frequency is not as accurate. Therefore, care should be observed when using this oscillator as the reference for the on-chip PLLs as their output clock frequency will be lower/higher than when using the 24 MHz crystal oscillator clock.

15.4.4 Crystal Frequency Detection(24 MHz)

A submodule exists that gives a fairly crude (relative to the accuracy of a crystal) estimation of whether the clock frequency is correct.

This function may be enabled by setting the XTALOSC24M_MISC0[OSC_XTALOK_EN] bit. It is disabled at system reset. When the oscillator is stable and the correct frequency is detected, the XTALOSC24M_MISC0[OSC_XTALOK] bit will be set. Note that the correct frequency will be observed before the oscillator fully blooms(the oscillation waveform build-up is completed).

15.5 Crystal Oscillator 32 kHz

15.5.1 Oscillator Configuration (32 kHz)

The basic block diagram of the 32 kHz module configured as a crystal oscillator is shown below.

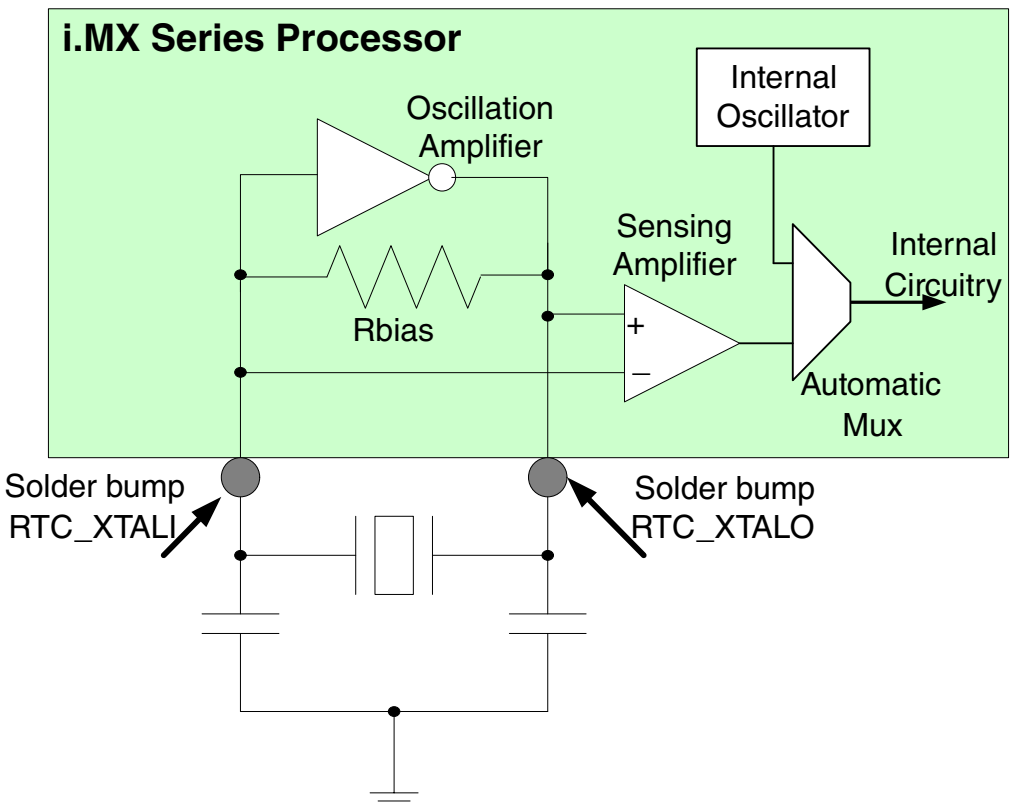


Figure 15-2. Oscillator Configuration (32 kHz)

This integrated biased amplifier can be used to create different frequency oscillators with different external component selection. Generally, RTC oscillators are either implemented with 32 kHz or 32.768 kHz crystals. Please consult the Security Reference Manual for appropriate frequency selection and configuration. Care must be taken to limit external leakage as this may debias the amplifier and degrade the gain.

The internal oscillator is automatically multiplexed in the clocking system when the system detects a loss of clock. The internal oscillator will provide clocks to the same on-chip modules as the external 32 kHz oscillator. The internal oscillator is not precise relative to a crystal. While it will provide a clock to the system, it generally will not be precise enough for long term time keeping. The internal oscillator is anticipated to be useful for quicker startup times and tampering prevention, but should not be used as the exclusive source for the 32 kHz clocks.

15.5.2 Bypass Configuration (32 kHz)

If it is desired to drive the chip with an external clock source, then the 32 kHz oscillator could be driven in one of three configurations using a nominal 1.1V source.

1. A single ended external clock source can be used to overdrive the output of the amplifier (RTC_XTALO). Since the oscillation sensing amplifier is differential, the RTC_XTALI pin should be externally floating and capacitively loaded. The combination of the internal biasing resistor and the external capacitor will filter the signal applied to the RTC_XTALO pin and develop a rough reference for the sensing amplifier to compare to.
2. A single ended external clock source can be used to drive RTC_XTALI. In this configuration, RTC_XTALO should be left externally floating.
3. A differential external clock source can be used to drive both RTC_XTALI and RTC_XTALO.

Generally, configuration 2 is anticipated to be the most used configuration, but all three configurations may be utilized.

15.6 XTALOSC 24MHz Memory Map/Register Definition

NOTE

The register content is mixed with analog functions not related to the oscillator function. These bits are noted.

XTALOSC24M memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400D_8150	Miscellaneous Register 0 (XTALOSC24M_MISC0)	32	R/W	0400_0000h	15.6.1/1138
400D_8154	Miscellaneous Register 0 (XTALOSC24M_MISC0_SET)	32	R/W	0400_0000h	15.6.1/1138
400D_8158	Miscellaneous Register 0 (XTALOSC24M_MISC0_CLR)	32	R/W	0400_0000h	15.6.1/1138
400D_815C	Miscellaneous Register 0 (XTALOSC24M_MISC0_TOG)	32	R/W	0400_0000h	15.6.1/1138
400D_8270	XTAL OSC (LP) Control Register (XTALOSC24M_LOWPWR_CTRL)	32	R/W	See section	15.6.2/1142
400D_8274	XTAL OSC (LP) Control Register (XTALOSC24M_LOWPWR_CTRL_SET)	32	R/W	See section	15.6.2/1142
400D_8278	XTAL OSC (LP) Control Register (XTALOSC24M_LOWPWR_CTRL_CLR)	32	R/W	See section	15.6.2/1142

Table continues on the next page...

XTALOSC24M memory map (continued)

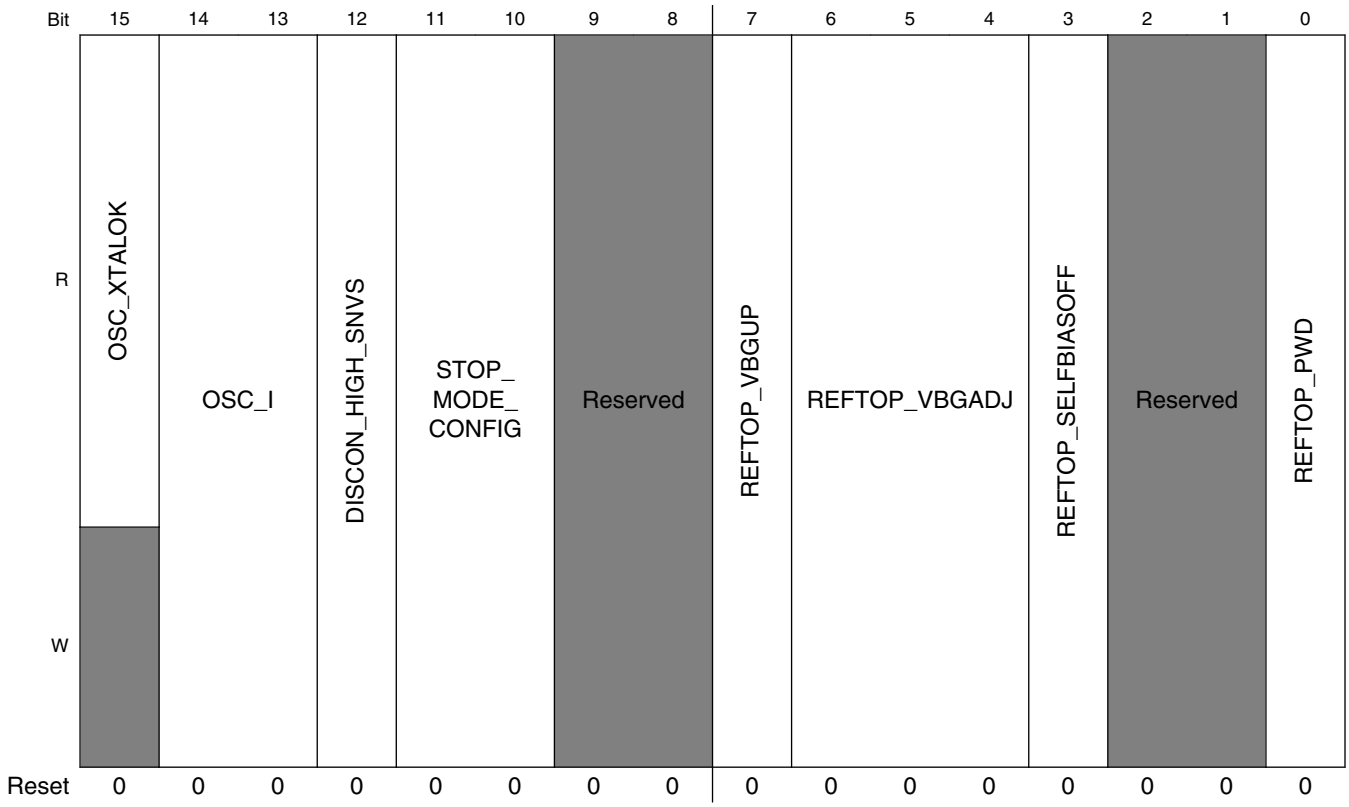
Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400D_827C	XTAL OSC (LP) Control Register (XTALOSC24M_LOWPWR_CTRL_TOG)	32	R/W	See section	15.6.2/1142
400D_82A0	XTAL OSC Configuration 0 Register (XTALOSC24M_OSC_CONFIG0)	32	R/W	0000_1020h	15.6.3/1145
400D_82A4	XTAL OSC Configuration 0 Register (XTALOSC24M_OSC_CONFIG0_SET)	32	R/W	0000_1020h	15.6.3/1145
400D_82A8	XTAL OSC Configuration 0 Register (XTALOSC24M_OSC_CONFIG0_CLR)	32	R/W	0000_1020h	15.6.3/1145
400D_82AC	XTAL OSC Configuration 0 Register (XTALOSC24M_OSC_CONFIG0_TOG)	32	R/W	0000_1020h	15.6.3/1145
400D_82B0	XTAL OSC Configuration 1 Register (XTALOSC24M_OSC_CONFIG1)	32	R/W	0000_02EEh	15.6.4/1146
400D_82B4	XTAL OSC Configuration 1 Register (XTALOSC24M_OSC_CONFIG1_SET)	32	R/W	0000_02EEh	15.6.4/1146
400D_82B8	XTAL OSC Configuration 1 Register (XTALOSC24M_OSC_CONFIG1_CLR)	32	R/W	0000_02EEh	15.6.4/1146
400D_82BC	XTAL OSC Configuration 1 Register (XTALOSC24M_OSC_CONFIG1_TOG)	32	R/W	0000_02EEh	15.6.4/1146
400D_82C0	XTAL OSC Configuration 2 Register (XTALOSC24M_OSC_CONFIG2)	32	R/W	0001_02E2h	15.6.5/1147
400D_82C4	XTAL OSC Configuration 2 Register (XTALOSC24M_OSC_CONFIG2_SET)	32	R/W	0001_02E2h	15.6.5/1147
400D_82C8	XTAL OSC Configuration 2 Register (XTALOSC24M_OSC_CONFIG2_CLR)	32	R/W	0001_02E2h	15.6.5/1147
400D_82CC	XTAL OSC Configuration 2 Register (XTALOSC24M_OSC_CONFIG2_TOG)	32	R/W	0001_02E2h	15.6.5/1147

15.6.1 Miscellaneous Register 0 (XTALOSC24M_MISC0*n*)

This register defines the control and status bits for miscellaneous analog blocks.

Address: 400D_8000h base + 150h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	VID_PLL_PREDIV	XTAL_24M_PWD	RTC_XTAL_SOURCE	CLKGATE_DELAY			CLKGATE_CTRL	Reserved								OSC_XTALOK_EN
W																
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0



XTALOSC24M_MISC0*n* field descriptions

Field	Description
31 VID_PLL_ PREDIV	Predivider for the source clock of the PLL's. **NOTE:** Not related to oscillator. 0 Divide by 1 1 Divide by 2
30 XTAL_24M_PWD	This field powers down the 24M crystal oscillator if set true.
29 RTC_XTAL_ SOURCE	This field indicates which chip source is being used for the rtc clock. 0 Internal ring oscillator 1 RTC_XTAL
28–26 CLKGATE_ DELAY	This field specifies the delay between powering up the XTAL 24MHz clock and releasing the clock to the digital logic inside the analog block. **NOTE:** Do not change the field during a low power event. This is not a field that the user would normally need to modify. 000 0.5ms 001 1.0ms 010 2.0ms 011 3.0ms 100 4.0ms

Table continues on the next page...

XTALOSC24M_MISC0*n* field descriptions (continued)

Field	Description
	101 5.0ms 110 6.0ms 111 7.0ms
25 CLKGATE_CTRL	This bit allows disabling the clock gate (always ungated) for the xtal 24MHz clock that clocks the digital logic in the analog block. **NOTE:** Do not change the field during a low power event. This is not a field that the user would normally need to modify. 0 **ALLOW_AUTO_GATE** — Allow the logic to automatically gate the clock when the XTAL is powered down. 1 **NO_AUTO_GATE** — Prevent the logic from ever gating off the clock.
24–17 -	This field is reserved. Always set to zero.
16 OSC_XTALOK_EN	This bit enables the detector that signals when the 24MHz crystal oscillator is stable.
15 OSC_XTALOK	Status bit that signals that the output of the 24-MHz crystal oscillator is stable. Generated from a timer and active detection of the actual frequency.
14–13 OSC_I	This field determines the bias current in the 24MHz oscillator. The aim is to start up with the highest bias current, which can be decreased after startup if it is determined to be acceptable. 00 **NOMINAL** — Nominal 01 **MINUS_12_5_PERCENT** — Decrease current by 12.5% 10 **MINUS_25_PERCENT** — Decrease current by 25.0% 11 **MINUS_37_5_PERCENT** — Decrease current by 37.5%
12 DISCON_HIGH_SNVS	This bit controls a switch from VDD_HIGH_IN to VDD_SNVS_IN. 0 Turn on the switch 1 Turn off the switch
11–10 STOP_MODE_CONFIG	Configure the analog behavior in stop mode. **NOTE:** Not related to oscillator. 00 All analog except rtc powered down on stop mode assertion. XtalOsc=on, RCOsc=off; 01 Certain analog functions such as certain regulators left up. XtalOsc=on, RCOsc=off; 10 XtalOsc=off, RCOsc=on, Old BG=on, New BG=off. 11 XtalOsc=off, RCOsc=on, Old BG=off, New BG=on.
9–8 -	This field is reserved. Reserved
7 REFTOP_VBGUP	Status bit that signals the analog bandgap voltage is up and stable. 1 - Stable. **NOTE:** Not related to oscillator.
6–4 REFTOP_VBGADJ	**NOTE:** Not related to oscillator. 000 Nominal VBG

Table continues on the next page...

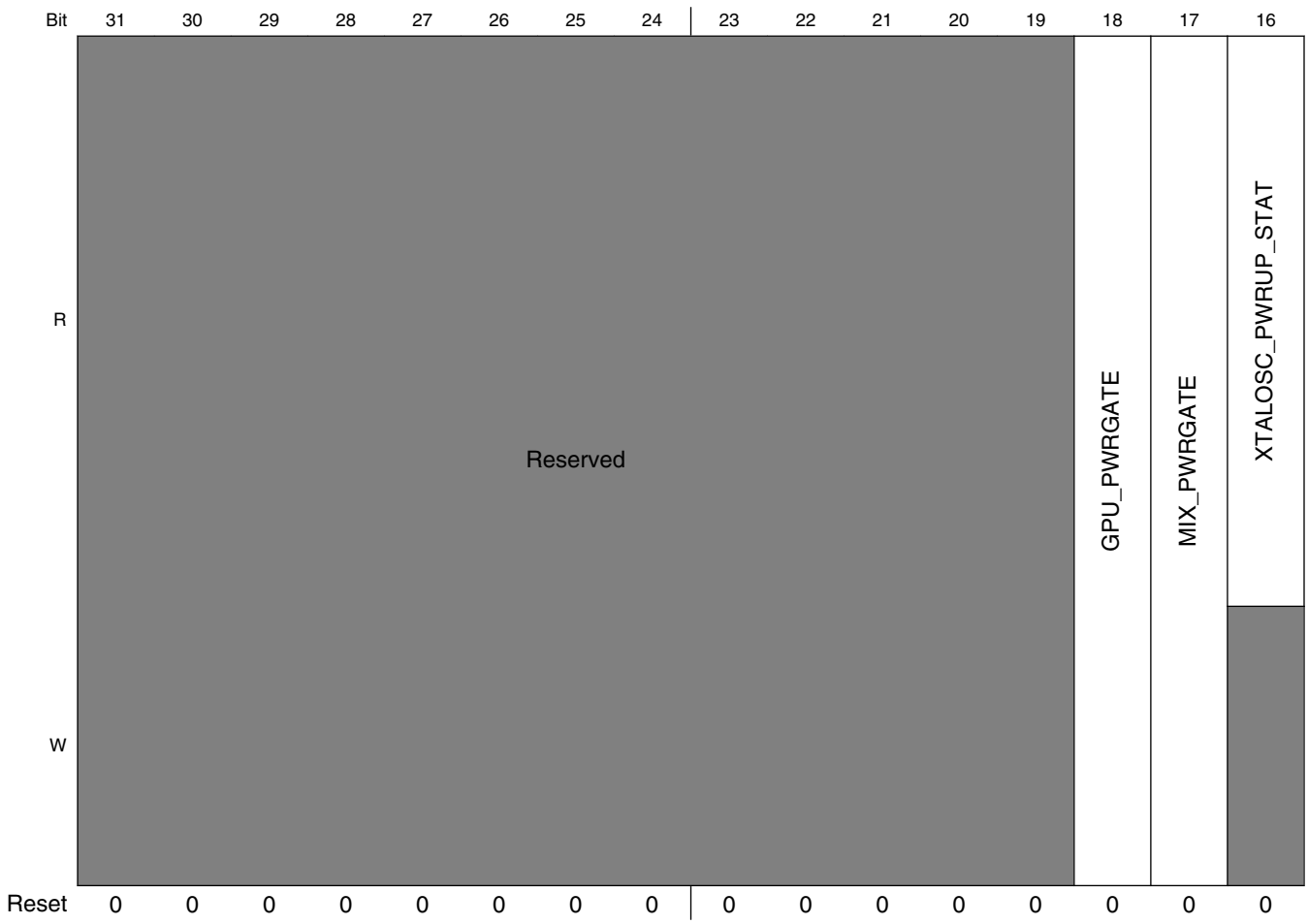
XTALOSC24M_MISC0*n* field descriptions (continued)

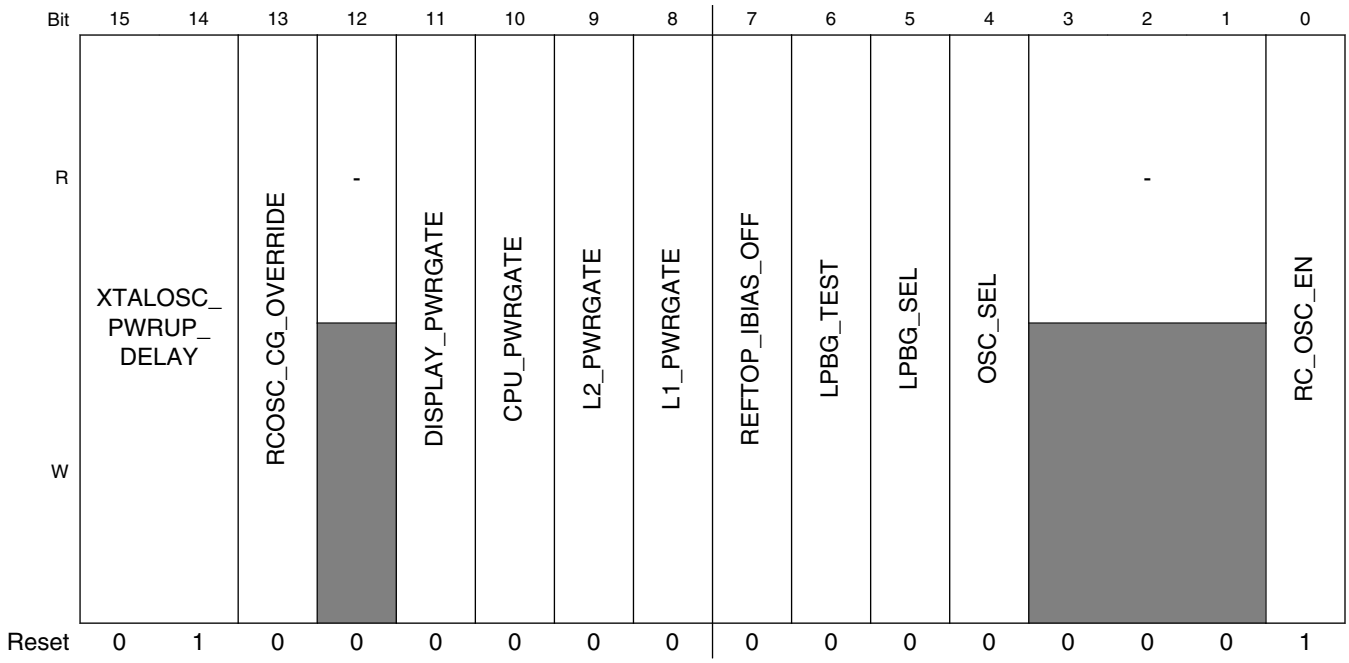
Field	Description
	001 VBG+0.78% 010 VBG+1.56% 011 VBG+2.34% 100 VBG-0.78% 101 VBG-1.56% 110 VBG-2.34% 111 VBG-3.12%
3 REFTOP_ SELFBIASOFF	Control bit to disable the self-bias circuit in the analog bandgap. The self-bias circuit is used by the bandgap during startup. This bit should be set after the bandgap has stabilized and is necessary for best noise performance of analog blocks using the outputs of the bandgap. **NOTE:** Value should be returned to zero before removing vddhigh_in or asserting bit 0 of this register (REFTOP_PWD) to assure proper restart of the circuit. **NOTE:** Not related to oscillator. 0 Uses coarse bias currents for startup 1 Uses bandgap-based bias currents for best performance.
2–1 -	This field is reserved.
0 REFTOP_PWD	Control bit to power-down the analog bandgap reference circuitry. **NOTE:** A note of caution, the bandgap is necessary for correct operation of most of the LDO, pll, and other analog functions on the die. **NOTE:** Not related to oscillator.

15.6.2 XTAL OSC (LP) Control Register (XTALOSC24M_LOWPWR_CTRL*n*)

This register defines xtal osc and low power configuration.

Address: 400D_8000h base + 270h offset + (4d × i), where i=0d to 3d





XTALOSC24M_LOWPWR_CTRL*n* field descriptions

Field	Description
31–19 -	This field is reserved.
18 GPU_PWRGATE	GPU power gate control. Used as software mask. Set to zero to force ungated.
17 MIX_PWRGATE	Display power gate control. Used as software mask. Set to zero to force ungated.
16 XTALOSC_ PWRUP_STAT	Status of the 24MHz xtal oscillator. 0 Not stable 1 Stable and ready to use
15–14 XTALOSC_ PWRUP_DELAY	Specifies the time delay between when the 24MHz xtal is powered up until it is stable and ready to use. 00 0.25ms 01 0.5ms 10 1ms 11 2ms
13 RCOSC_CG_ OVERRIDE	For debug purposes only. This bit effects clock gating of certain digital logic clocked by the 24MHz clk.
12 -	Reserved
11 DISPLAY_ PWRGATE	Display logic power gate control. Used as software override. **NOTE:** Not related to oscillator.
10 CPU_PWRGATE	CPU power gate control. Used as software override.

Table continues on the next page...

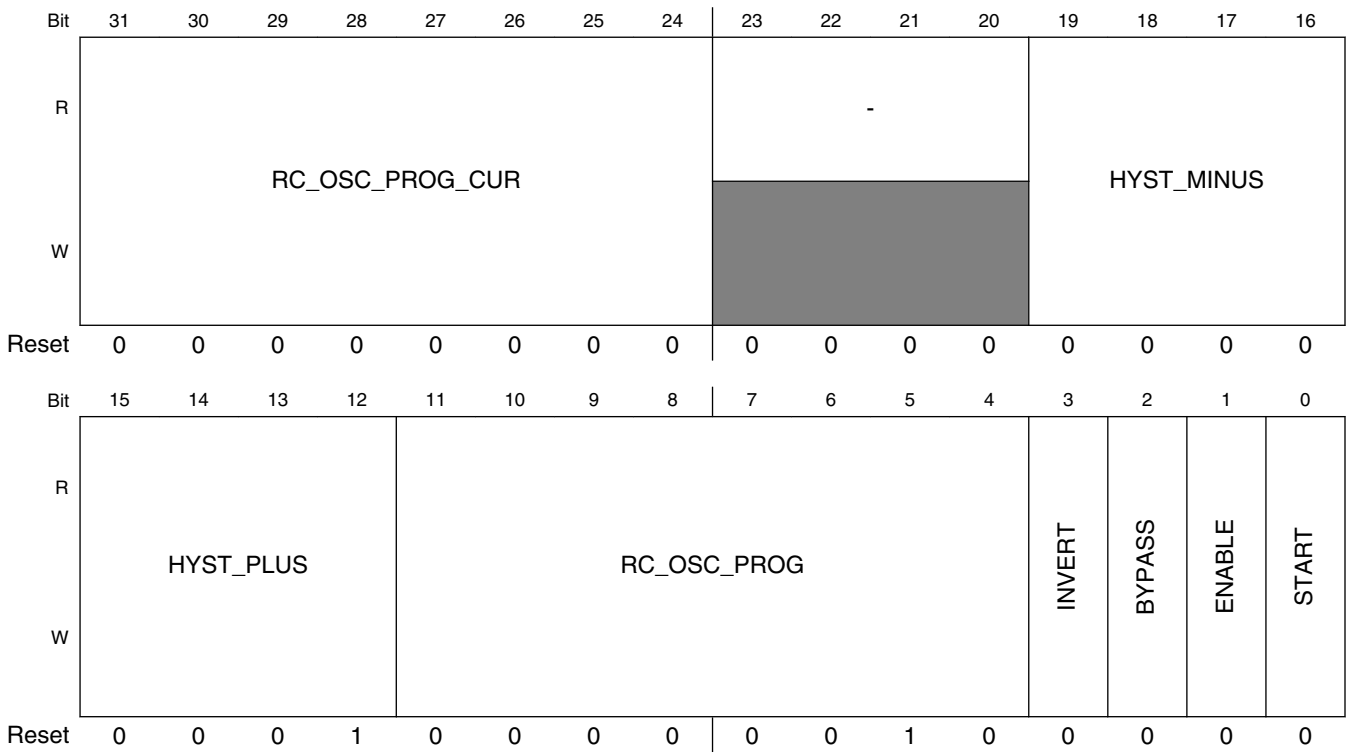
XTALOSC24M_LOWPWR_CTRL*n* field descriptions (continued)

Field	Description
	Attention: Test purpose only **NOTE:** Not related to oscillator.
9 L2_PWRGATE	L2 power gate control. Used as software override. **NOTE:** Not related to oscillator.
8 L1_PWRGATE	L1 power gate control. Used as software override. **NOTE:** Not related to oscillator.
7 REFTOP_IBIAS_OFF	Low power reftop ibias disable. **NOTE:** Not related to oscillator.
6 LPBG_TEST	Low power bandgap test bit. **NOTE:** Not related to oscillator.
5 LPBG_SEL	Bandgap select. **NOTE:** Not related to oscillator. 0 Normal power bandgap 1 Low power bandgap
4 OSC_SEL	Select the source for the 24MHz clock. 0 XTAL OSC 1 RC OSC
3–1 -	Reserved
0 RC_OSC_EN	RC Osc. enable control. 0 Use XTAL OSC to source the 24MHz clock 1 Use RC OSC

15.6.3 XTAL OSC Configuration 0 Register (XTALOSC24M_OSC_CONFIG0*n*)

This register is used to configure the 24MHz RC oscillator.

Address: 400D_8000h base + 2A0h offset + (4d × i), where i=0d to 3d



XTALOSC24M_OSC_CONFIG0*n* field descriptions

Field	Description
31–24 RC_OSC_ PROG_CUR	The current tuning value in use.
23–20 -	Reserved.
19–16 HYST_MINUS	Negative hysteresis value. Subtracted from target value before comparison. This, along with HYST_PLUS creates a range.
15–12 HYST_PLUS	Positive hysteresis value. Added to target value before comparison. This, along with HYST_MINUS creates a range.
11–4 RC_OSC_PROG	RC osc. tuning values.
3 INVERT	Invert the stepping of the calculated RC tuning value.

Table continues on the next page...

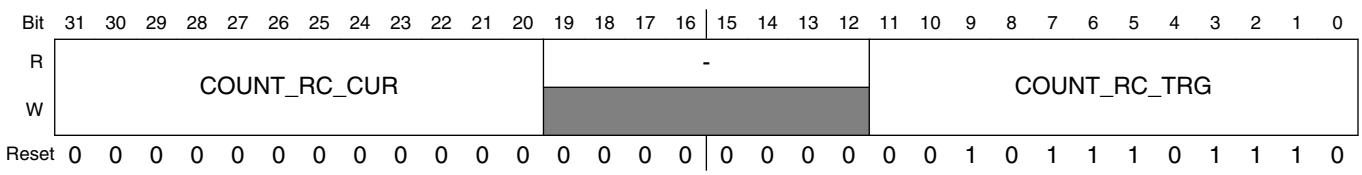
XTALOSC24M_OSC_CONFIG0*n* field descriptions (continued)

Field	Description
2 BYPASS	Bypasses any calculated RC tuning value and uses the programmed register value.
1 ENABLE	Enables the tuning logic to calculate new RC tuning values. Disabling essentially freezes the state of the calculation.
0 START	Start/stop bit for the RC tuning calculation logic. If stopped the tuning logic is reset.

15.6.4 XTAL OSC Configuration 1 Register (XTALOSC24M_OSC_CONFIG1*n*)

This register is used to configure the 24MHz RC oscillator.

Address: 400D_8000h base + 2B0h offset + (4d × i), where i=0d to 3d



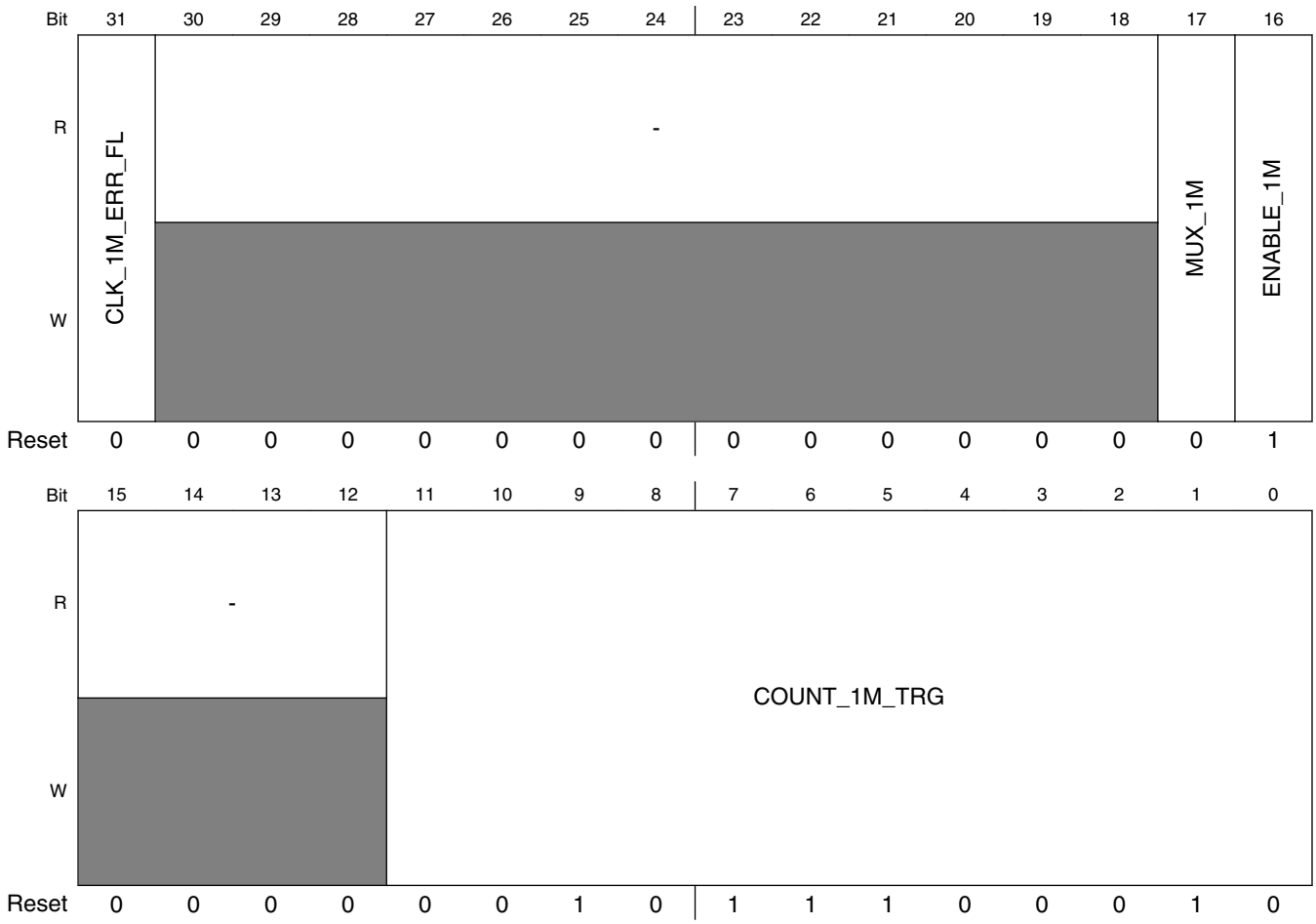
XTALOSC24M_OSC_CONFIG1*n* field descriptions

Field	Description
31–20 COUNT_RC_ CUR	The current tuning value in use.
19–12 -	Reserved
COUNT_RC_ TRG	The target count used to tune the RC OSC frequency. Essentially the number of desired RC OSC clock cycles is within one 32kHz clock cycle.

15.6.5 XTAL OSC Configuration 2 Register (XTALOSC24M_OSC_CONFIG2*n*)

This register is used to configure the 1MHz clock generated from the 24MHz RC oscillator.

Address: 400D_8000h base + 2C0h offset + (4d × i), where i=0d to 3d



XTALOSC24M_OSC_CONFIG2*n* field descriptions

Field	Description
31 CLK_1M_ERR_FL	Flag indicates that the count_1m count wasn't reached within 1 32kHz period. This is intended as feedback to software that the HW_ANADIG_OSC_CONFIG2_COUNT_1M_TRG value is too high for the RC Osc frequency.
30–18 -	Reserved.
17 MUX_1M	Mux the corrected or uncorrected 1MHz clock to the output. 0 - free 1MHz clock; 1 - locked 1MHz clock.

Table continues on the next page...

XTALOSC24M_OSC_CONFIG2*n* field descriptions (continued)

Field	Description
16 ENABLE_1M	Enable the 1MHz clock output. 0 - disabled; 1 - enabled.
15–12 -	Reserved
COUNT_1M_ TRG	The target count used to tune the 1MHz clock frequency. Essentially the number of desired RC OSC clock cycles used to generate the 1MHz clock is within one 32kHz clock cycle.

Chapter 16
Power Management Unit (PMU)

16.1 Chip-specific PMU information

Table 16-1. Reference links to related information

Topic	Related module(s)	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
Interrupts, DMA Events and XBAR Assignments	-	Interrupts, DMA Events and XBAR Assignments

NOTE

The following bitfields are not applicable on this device, and those corresponding bits are **Reserved**:

- in PMU_REG_CORE*n* register:
  - RAMP_RATE
  - REG2_TARG
  - REG1_TARG
  - REG0_TARG
- in PMU_MISC1*n* register: IRQ_DIG_BO.
- in PMU_MISC2*n* register:
  - REG2_STEP_TIME
  - REG1_STEP_TIME
  - REG0_STEP_TIME
  - REG2_OK
  - REG2_ENABLE_BO
  - REG1_ENABLE_BO

- REG0_ENABLE_BO
- REG2_BO_STATUS
- REG1_BO_STATUS
- REG0_BO_STATUS
- REG2_BO_OFFSET
- REG1_BO_OFFSET
- REG0_BO_OFFSET

16.2 Overview

The power management unit (PMU) is designed to simplify the external power interface. The power system can be split into the input power sources and their characteristics, the integrated power transforming and controlling elements, and the final load interconnection and requirements.

A typical power system uses the PMU is depicted in the following diagram.

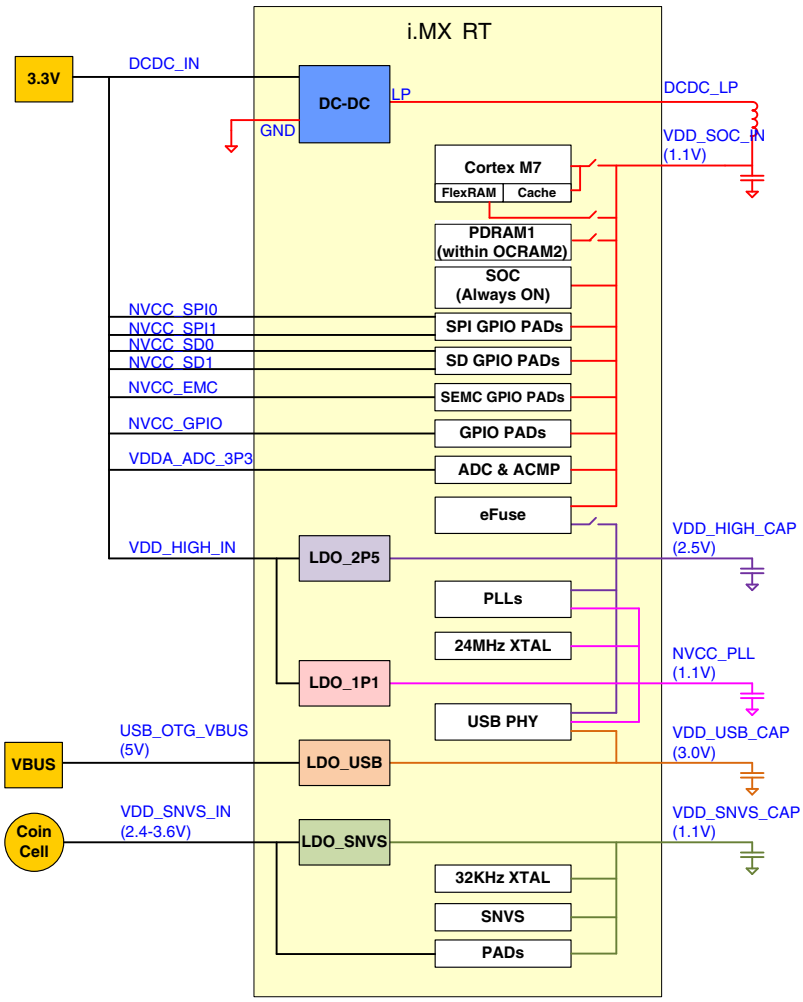


Figure 16-1. Power system overview

Using four LDO regulators, the number of external supplies is greatly reduced. Not counting the backup coin and USB inputs, the number of external supplies is reduced to two. Missing from this external supply total is the number of necessary external supplies to power the desired memory interface; that number varies depending on the type of external memory selected. Other supplies may also be necessary to supply the voltage to the different I/O power segments if their I/O voltages have to be different from what is provided above.

16.3 Analog LDO Regulators

There are two analog regulators described here.

16.3.1 LDO 1P1

The LDO_1P1 module on the chip implements a programmable linear-regulator function from a higher analog supply voltage (2.8 V–3.3 V) to produce a nominal 1.1 V output voltage.

The output of the regulator can be programmed in 25 mV steps from 0.8 V to 1.4 V. The regulator has been designed to be stable with a minimum external low-ESR decoupling capacitance, though the actual capacitance required should be determined by the application. A programmable brownout detector is included in the regulator which can be used by the system to determine when the load capability of the regulator is being exceeded, so the necessary steps can be taken.

Current limiting can be enabled by setting the PMU_REG_1P1[ENABLE_ILIMIT] bit to allow for in-rush current requirements during startup if needed. Active pulldown can also be enabled by setting the PMU_REG_1P1[ENABLE_PULLDOWN] bit for systems requiring this feature.

16.3.2 LDO 2P5

The LDO_2P5 module on the chip implements a programmable linear-regulator function from a higher analog supply voltage (2.8V-3.3V) to produce a nominal 2.5V output voltage.

The output of the regulator can be programmed in 25mV steps from 2.0V to 2.75V. The regulator has been designed to be stable with a minimum external low-ESR decoupling capacitance, though the actual capacitance required should be determined by the application. A programmable brown-out detector is included in the regulator which can be used by the system to determine when the load capability of the regulator is being exceeded to take the necessary steps.

Current-limiting can be enabled by setting the REG_PMU_2P5[ENABLE_ILIMIT] bit to allow for in-rush current requirements during start-up if needed. Active-pulldown can also be enabled by setting the REG_PMU_2P5[ENABLE_PULLDOWN] bit for systems requiring this feature.

16.3.3 Low Power Operation

The 1.1 V and 2.5 V LDO includes an alternate, self-biased, low-precision, weak regulator which can be enabled for applications needing to keep the 1.1 V and 2.5 V output voltage alive during low-power modes where the main regulator and its associated global bandgap reference module are disabled.

The output of this weak regulator is not programmable and is a function of its input power supply as well as load current. The low-power mode is enabled by setting high the PMU_REG_1P1[ENABLE_WEAK_LINREG] and PMU_REG_2P5[ENABLE_WEAK_LINREG] bit of the regulator. It is recommended that the following sequence be followed to enable this mode:

1. Throttle down the 1.1 V / 2.5 V attached load to its low-power maintain state.
2. Disable the main 1.1 V / 2.5 V regulator driver by clearing the PMU_REG_1P1[ENABLE_LINREG] / PMU_REG_2P5[ENABLE_LINREG] bit.
3. Enable the weak 1.1 V / 2.5 V regulator by setting the PMU_REG_1P1[ENABLE_WEAK_LINREG] / PMU_REG_2P5[ENABLE_WEAK_LINREG] bit.

To go back to full-power operation, reverse the steps outlined above. Note that the external decoupling cap is supporting the power supply between steps 2 and 3. Therefore step 3 should happen appropriately in time relative to the discharge of the supporting capacitor.

16.4 USB LDO Regulator

The USB_LDO module on the chip implements a programmable linear-regulator function from the USB VBUS voltages (typically 5 V) to produce a nominal 3.0 V output voltage.

The output of the regulator can be programmed in 25 mV steps, from 2.625V to 3.4 V . The regulator has been designed to be stable with a minimum external low-ESR decoupling capacitor of 4.7 μF, though the actual capacitance required should be determined by the application. A programmable brownout detector is included in the regulator which can be used by the system to determine when the load capability of the regulator is being exceeded, so the necessary steps can be taken. This regulator has a built-in power mux which allows the user to choose to run the regulator from either VBUS supply when both are present. If only one of the VBUS voltages is present, then the regulator automatically selects this supply. Current limit is also included to help the system meet in-rush current targets.

Upon attachment of VBUS, this regulator starts up in a low-power, self-preservation mode to prevent over-voltage conditions on the chip. It is expected that the user transition to full regulation by enabling the regulator and disabling the in-rush current limits via its control registers. Upon VBUS removal, it is further expected that the regulator controls are returned to their reset state.

16.5 SNVS Regulator

The SNVS regulator takes the SNVS_IN supply and generates the SNVS_CAP supply, which powers the real time clock and SNVS blocks.

The SNVS_LDO is a non-programmable linear-regulator function, producing a nominal 1.1V output voltage.

16.6 PMU Memory Map/Register Definition

The register definitions that affect the behavior of the digital LDO regulators follow.

NOTE

Some of the registers are collections of bits that affect multiple components on the chip. Those that are not pertinent to this chapter have comments in the related register bitfields.

If a full description is desired, please consult the full register programming reference in the related block.

PMU memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400D_8110	Regulator 1P1 Register (PMU_REG_1P1)	32	R/W	0000_1073h	16.6.1/1156
400D_8114	Regulator 1P1 Register (PMU_REG_1P1_SET)	32	R/W	0000_1073h	16.6.1/1156
400D_8118	Regulator 1P1 Register (PMU_REG_1P1_CLR)	32	R/W	0000_1073h	16.6.1/1156
400D_811C	Regulator 1P1 Register (PMU_REG_1P1_TOG)	32	R/W	0000_1073h	16.6.1/1156
400D_8120	Regulator 3P0 Register (PMU_REG_3P0)	32	R/W	0000_0F74h	16.6.2/1159
400D_8124	Regulator 3P0 Register (PMU_REG_3P0_SET)	32	R/W	0000_0F74h	16.6.2/1159
400D_8128	Regulator 3P0 Register (PMU_REG_3P0_CLR)	32	R/W	0000_0F74h	16.6.2/1159
400D_812C	Regulator 3P0 Register (PMU_REG_3P0_TOG)	32	R/W	0000_0F74h	16.6.2/1159
400D_8130	Regulator 2P5 Register (PMU_REG_2P5)	32	R/W	0000_1073h	16.6.3/1161
400D_8134	Regulator 2P5 Register (PMU_REG_2P5_SET)	32	R/W	0000_1073h	16.6.3/1161
400D_8138	Regulator 2P5 Register (PMU_REG_2P5_CLR)	32	R/W	0000_1073h	16.6.3/1161
400D_813C	Regulator 2P5 Register (PMU_REG_2P5_TOG)	32	R/W	0000_1073h	16.6.3/1161
400D_8140	Digital Regulator Core Register (PMU_REG_CORE)	32	R/W	0048_2012h	16.6.4/1163
400D_8144	Digital Regulator Core Register (PMU_REG_CORE_SET)	32	R/W	0048_2012h	16.6.4/1163
400D_8148	Digital Regulator Core Register (PMU_REG_CORE_CLR)	32	R/W	0048_2012h	16.6.4/1163
400D_814C	Digital Regulator Core Register (PMU_REG_CORE_TOG)	32	R/W	0048_2012h	16.6.4/1163
400D_8150	Miscellaneous Register 0 (PMU_MISC0)	32	R/W	0400_0000h	16.6.5/1167

Table continues on the next page...

PMU memory map (continued)

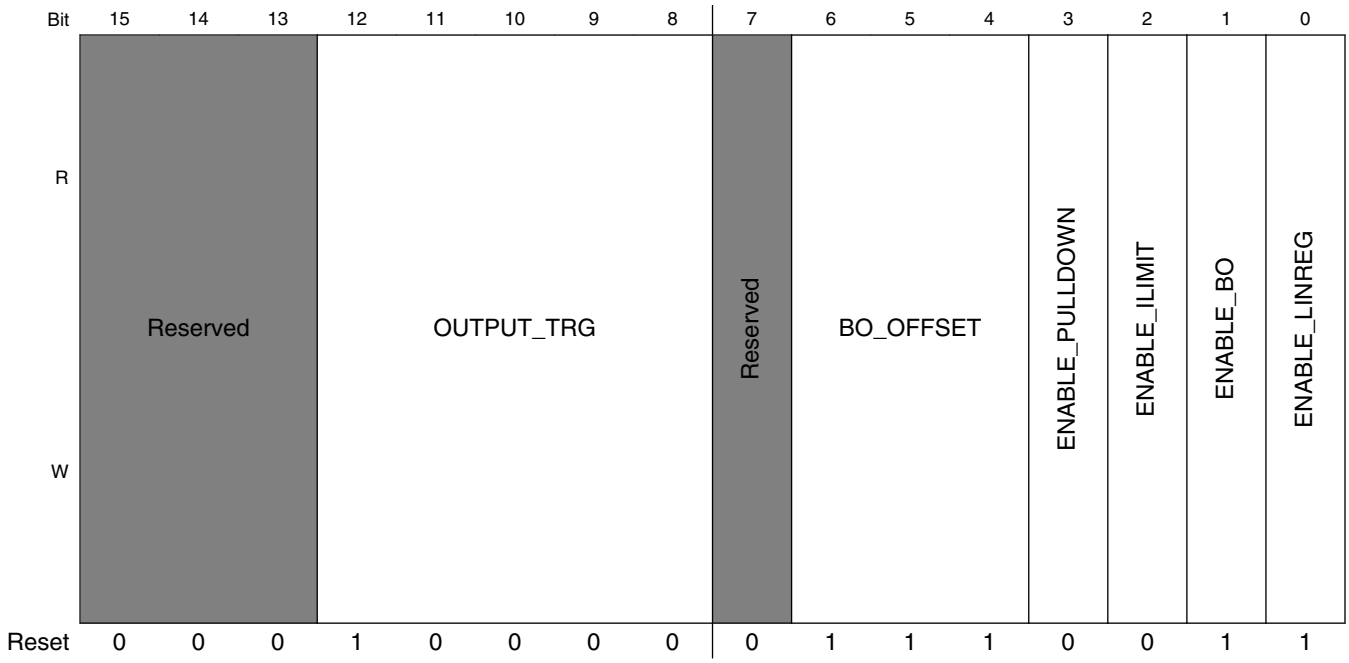
Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400D_8154	Miscellaneous Register 0 (PMU_MISC0_SET)	32	R/W	0400_0000h	16.6.5/1167
400D_8158	Miscellaneous Register 0 (PMU_MISC0_CLR)	32	R/W	0400_0000h	16.6.5/1167
400D_815C	Miscellaneous Register 0 (PMU_MISC0_TOG)	32	R/W	0400_0000h	16.6.5/1167
400D_8160	Miscellaneous Register 1 (PMU_MISC1)	32	R/W	0000_0000h	16.6.6/1171
400D_8164	Miscellaneous Register 1 (PMU_MISC1_SET)	32	R/W	0000_0000h	16.6.6/1171
400D_8168	Miscellaneous Register 1 (PMU_MISC1_CLR)	32	R/W	0000_0000h	16.6.6/1171
400D_816C	Miscellaneous Register 1 (PMU_MISC1_TOG)	32	R/W	0000_0000h	16.6.6/1171
400D_8170	Miscellaneous Control Register (PMU_MISC2)	32	R/W	0027_2727h	16.6.7/1173
400D_8174	Miscellaneous Control Register (PMU_MISC2_SET)	32	R/W	0027_2727h	16.6.7/1173
400D_8178	Miscellaneous Control Register (PMU_MISC2_CLR)	32	R/W	0027_2727h	16.6.7/1173
400D_817C	Miscellaneous Control Register (PMU_MISC2_TOG)	32	R/W	0027_2727h	16.6.7/1173

16.6.1 Regulator 1P1 Register (PMU_REG_1P1*n*)

This register defines the control and status bits for the 1.1V regulator. This regulator is designed to power the digital portions of the analog cells.

Address: 400D_8000h base + 110h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved												SELREF_WEAK_LINREG	ENABLE_WEAK_LINREG	OK_VDD1P1	BO_VDD1P1
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



PMU_REG_1P1*n* field descriptions

Field	Description
31–20 -	This field is reserved.
19 SELREF_ WEAK_LINREG	Selects the source for the reference voltage of the weak 1p1 regulator. 0 Weak-linreg output tracks low-power-bandgap voltage 1 Weak-linreg output tracks VDD_SOC_IN voltage
18 ENABLE_ WEAK_LINREG	Enables the weak 1p1 regulator. This regulator can be used when the main 1p1 regulator is disabled, under low-power conditions.
17 OK_VDD1P1	Status bit that signals when the regulator output is ok. 1 = regulator output > brownout target
16 BO_VDD1P1	Status bit that signals when a brownout is detected on the regulator output.
15–13 -	This field is reserved.
12–8 OUTPUT_TRG	Control bits to adjust the regulator output voltage. Each LSB is worth 25mV. Programming examples are detailed below. Other output target voltages may be interpolated from these examples. Choices must be in this range: 0x1b >= output_trg >= 0x04 **NOTE:** There may be reduced chip functionality or reliability at the extremes of the programming range. 0x04 0.8V 0x10 1.1V 0x1b 1.375V
7 -	This field is reserved.

Table continues on the next page...

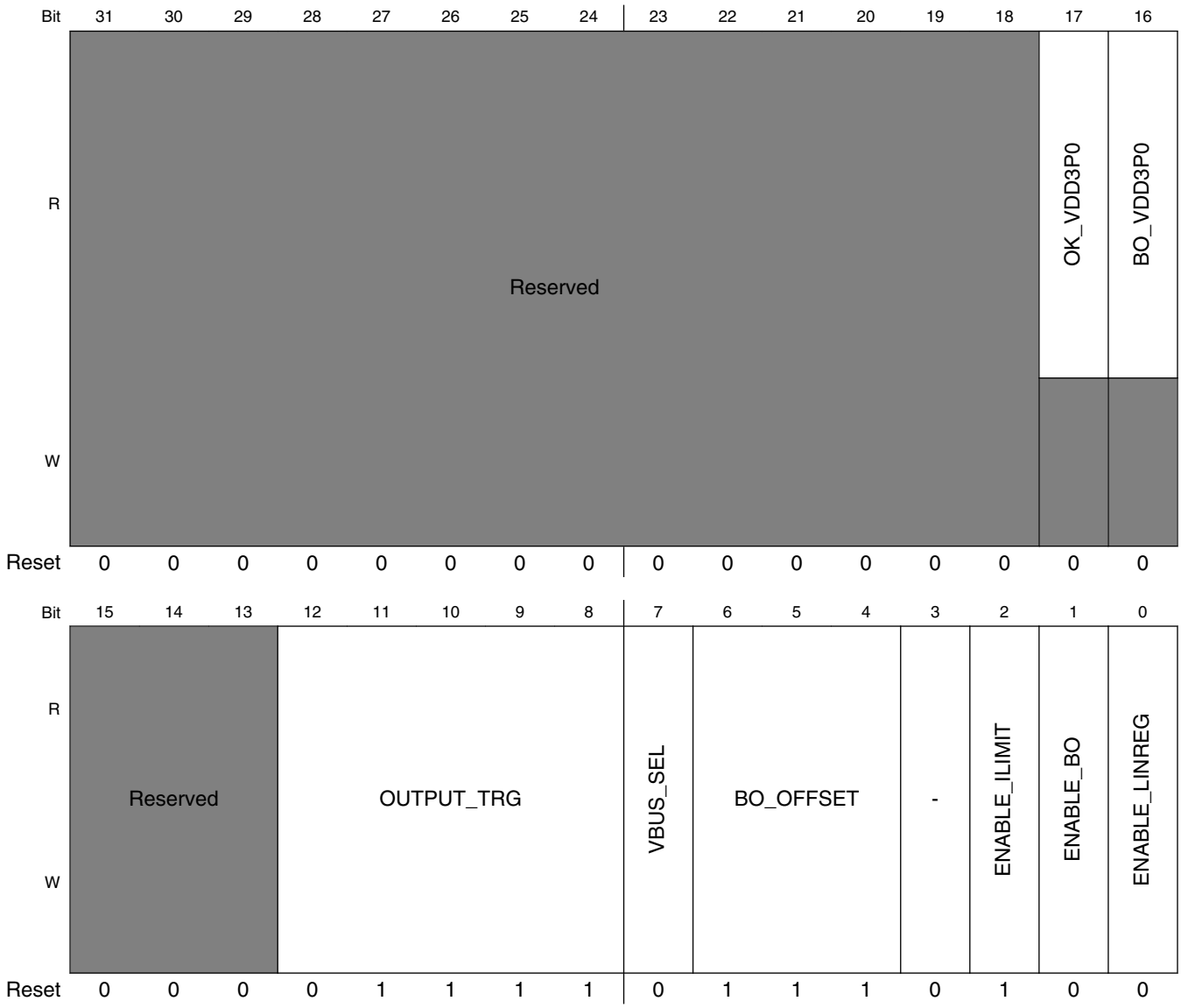
PMU_REG_1P1*n* field descriptions (continued)

Field	Description
6–4 BO_OFFSET	Control bits to adjust the regulator brownout offset voltage in 25mV steps. The reset brown-offset is 175mV below the programmed target code. Brownout target = OUTPUT_TRG - BO_OFFSET. Some steps may be irrelevant because of input supply limitations or load operation.
3 ENABLE_ PULLDOWN	Control bit to enable the pull-down circuitry in the regulator
2 ENABLE_ILIMIT	Control bit to enable the current-limit circuitry in the regulator.
1 ENABLE_BO	Control bit to enable the brownout circuitry in the regulator.
0 ENABLE_ LINREG	Control bit to enable the regulator output.

16.6.2 Regulator 3P0 Register (PMU_REG_3P0*n*)

This register defines the control and status bits for the 3.0V regulator powered by the host USB VBUS pin.

Address: 400D_8000h base + 120h offset + (4d × i), where i=0d to 3d



PMU_REG_3P0*n* field descriptions

Field	Description
31–18 -	This field is reserved.

Table continues on the next page...

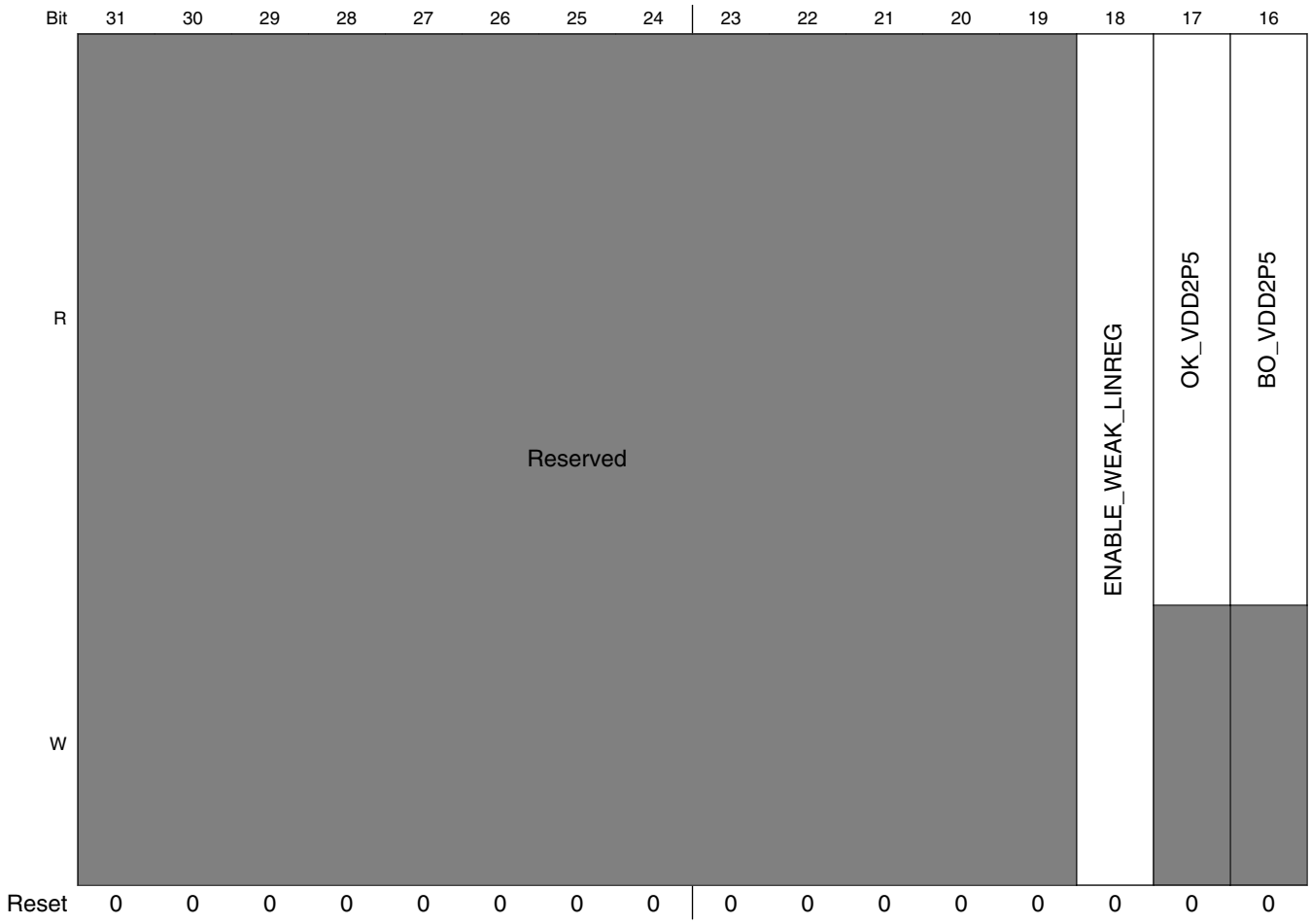
PMU_REG_3P0*n* field descriptions (continued)

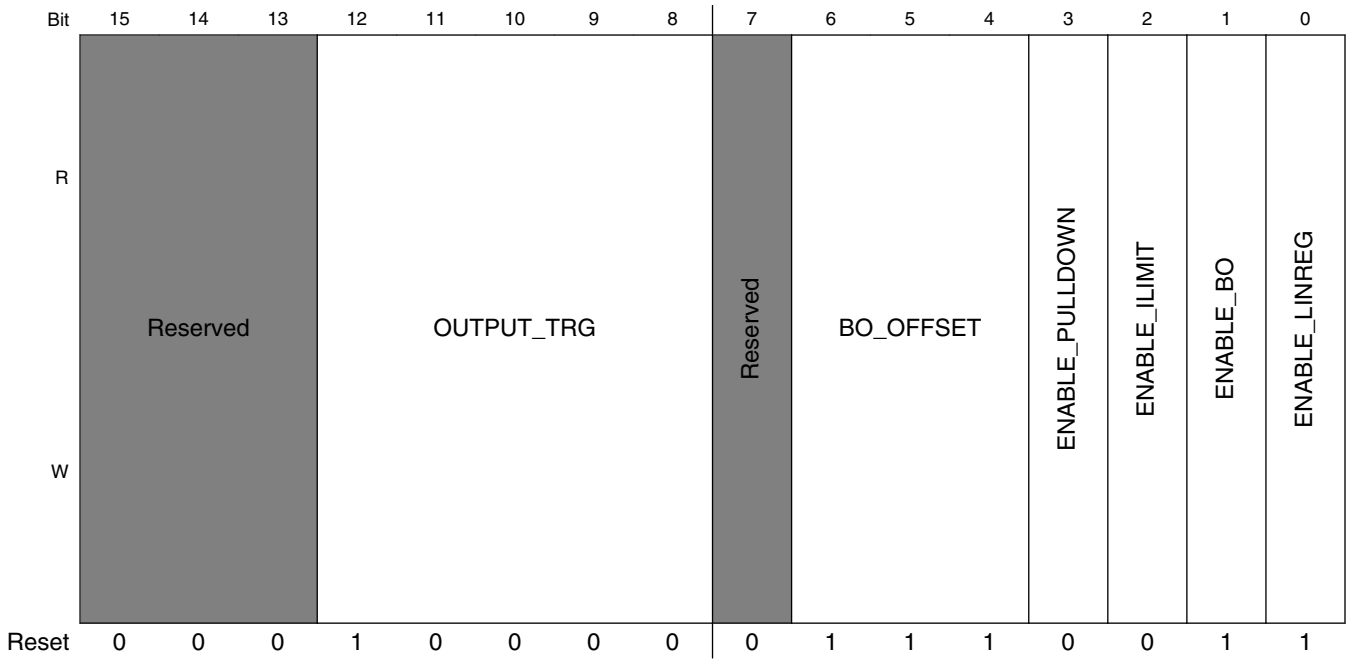
Field	Description
17 OK_VDD3P0	Status bit that signals when the regulator output is ok. 1 = regulator output > brownout target
16 BO_VDD3P0	Status bit that signals when a brownout is detected on the regulator output.
15–13 -	This field is reserved.
12–8 OUTPUT_TRG	Control bits to adjust the regulator output voltage. Each LSB is worth 25mV. Programming examples are detailed below. Other output target voltages may be interpolated from these examples. **NOTE:** There may be reduced chip functionality or reliability at the extremes of the programming range. 0x00 2.625V 0x0f 3.000V 0x1f 3.400V
7 VBUS_SEL	Select input voltage source for LDO_3P0 from either USB_OTG1_VBUS or USB_OTG2_VBUS. If only one of the two VBUS voltages is present, it is automatically selected. 1 **USB_OTG1_VBUS** — Utilize VBUS OTG1 power 0 **USB_OTG2_VBUS** — Utilize VBUS OTG2 power
6–4 BO_OFFSET	Control bits to adjust the regulator brownout offset voltage in 25mV steps. The reset brown-offset is 175mV below the programmed target code. Brownout target = OUTPUT_TRG - BO_OFFSET. Some steps may not be relevant because of input supply limitations or load operation.
3 -	Reserved
2 ENABLE_ILIMIT	Control bit to enable the current-limit circuitry in the regulator.
1 ENABLE_BO	Control bit to enable the brownout circuitry in the regulator.
0 ENABLE_ LINREG	Control bit to enable the regulator output to be set by the programmed target voltage setting and internal bandgap reference.

16.6.3 Regulator 2P5 Register (PMU_REG_2P5*n*)

This register defines the control and status bits for the 2.5V regulator.

Address: 400D_8000h base + 130h offset + (4d × i), where i=0d to 3d





PMU_REG_2P5*n* field descriptions

Field	Description
31–19 -	This field is reserved.
18 ENABLE_ WEAK_LINREG	Enables the weak 2p5 regulator. This low power regulator is used when the main 2p5 regulator is disabled to keep the 2.5V output roughly at 2.5V. Scales directly with the value of VDDHIGH_IN.
17 OK_VDD2P5	Status bit that signals when the regulator output is ok. 1 = regulator output > brownout target
16 BO_VDD2P5	Status bit that signals when a brownout is detected on the regulator output.
15–13 -	This field is reserved.
12–8 OUTPUT_TRG	Control bits to adjust the regulator output voltage. Each LSB is worth 25mV. Programming examples are detailed below. Other output target voltages may be interpolated from these examples. **NOTE:** There may be reduced chip functionality or reliability at the extremes of the programming range. 0x00 2.10V 0x10 2.50V 0x1f 2.875V
7 -	This field is reserved.
6–4 BO_OFFSET	Control bits to adjust the regulator brownout offset voltage in 25mV steps. The reset brown-offset is 175mV below the programmed target code. Brownout target = OUTPUT_TRG - BO_OFFSET. Some steps may be irrelevant because of input supply limitations or load operation.
3 ENABLE_ PULLDOWN	Control bit to enable the pull-down circuitry in the regulator

Table continues on the next page...

PMU_REG_2P5*n* field descriptions (continued)

Field	Description
2 ENABLE_ILIMIT	Control bit to enable the current-limit circuitry in the regulator.
1 ENABLE_BO	Control bit to enable the brownout circuitry in the regulator.
0 ENABLE_ LINREG	Control bit to enable the regulator output.

16.6.4 Digital Regulator Core Register (PMU_REG_CORE*n*)

This register defines the function of the digital regulators

Address: 400D_8000h base + 140h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved		FET_ODRIVE	RAMP_RATE		REG2_ADJ				REG2_TARG					REG1_ADJ	
Reset	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	REG1_ADJ		REG1_TARG						REG0_ADJ			REG0_TARG				
Reset	0	0	1	0	0	0	0	0	0	0	0	1	0	0	1	0

PMU_REG_CORE*n* field descriptions

Field	Description
31–30 -	This field is reserved.
29 FET_ODRIVE	If set, increases the gate drive on power gating FETs to reduce leakage in the off state. Care must be taken to apply this bit only when the input supply voltage to the power FET is less than 1.1V. **NOTE:** This bit should only be used in low-power modes where the external input supply voltage is nominally 0.9V.
28–27 RAMP_RATE	Regulator voltage ramp rate. 00 Fast 01 Medium Fast

Table continues on the next page...

PMU_REG_CORE*n* field descriptions (continued)

Field	Description
	10 Medium Slow 11 Slow
26–23 REG2_ADJ	This bit field defines the adjustment bits to calibrate the target value of Reg2. The adjustment is applied on top on any adjustment applied to the global reference in the misc0 register. 0000 No adjustment 0001 + 0.25% 0010 + 0.50% 0011 + 0.75% 0100 + 1.00% 0101 + 1.25% 0110 + 1.50% 0111 + 1.75% 1000 - 0.25% 1001 - 0.50% 1010 - 0.75% 1011 - 1.00% 1100 - 1.25% 1101 - 1.50% 1110 - 1.75% 1111 - 2.00%
22–18 REG2_TARG	This field defines the target voltage for the SOC power domain. Single-bit increments reflect 25mV core voltage steps. Some steps may not be relevant because of input supply limitations or load operation. **NOTE:** This register is capable of programming an over-voltage condition on the device. Consult the datasheet Operating Ranges table for the allowed voltages. 00000 Power gated off 00001 Target core voltage = 0.725V 00010 Target core voltage = 0.750V 00011 Target core voltage = 0.775V ... 10000 Target core voltage = 1.100V ... 11110 Target core voltage = 1.450V 11111 Power FET switched full on. No regulation.
17–14 REG1_ADJ	This bit field defines the adjustment bits to calibrate the target value of Reg1. The adjustment is applied on top on any adjustment applied to the global reference in the misc0 register. 0000 No adjustment 0001 + 0.25% 0010 + 0.50% 0011 + 0.75% 0100 + 1.00% 0101 + 1.25% 0110 + 1.50% 0111 + 1.75% 1000 - 0.25%

Table continues on the next page...

PMU_REG_CORE*n* field descriptions (continued)

Field	Description
	1001 - 0.50% 1010 - 0.75% 1011 - 1.00% 1100 - 1.25% 1101 - 1.50% 1110 - 1.75% 1111 - 2.00%
13–9 REG1_TARG	This bit field defines the target voltage for the vpu/gpu power domain. Single bit increments reflect 25mV core voltage steps. Not all steps will make sense to use either because of input supply limitations or load operation. 00000 Power gated off 00001 Target core voltage = 0.725V 00010 Target core voltage = 0.750V 00011 Target core voltage = 0.775V ... 10000 Target core voltage = 1.100V ... 11110 Target core voltage = 1.450V 11111 Power FET switched full on. No regulation.
8–5 REG0_ADJ	This bit field defines the adjustment bits to calibrate the target value of Reg0. The adjustment is applied on top on any adjustment applied to the global reference in the misc0 register. 0000 No adjustment 0001 + 0.25% 0010 + 0.50% 0011 + 0.75% 0100 + 1.00% 0101 + 1.25% 0110 + 1.50% 0111 + 1.75% 1000 - 0.25% 1001 - 0.50% 1010 - 0.75% 1011 - 1.00% 1100 - 1.25% 1101 - 1.50% 1110 - 1.75% 1111 - 2.00%
REG0_TARG	This field defines the target voltage for the Arm core power domain. Single-bit increments reflect 25mV core voltage steps. Some steps may not be relevant because of input supply limitations or load operation. **NOTE:** This register is capable of programming an over-voltage condition on the device. Consult the datasheet Operating Ranges table for the allowed voltages. 00000 Power gated off 00001 Target core voltage = 0.725V 00010 Target core voltage = 0.750V

Table continues on the next page...

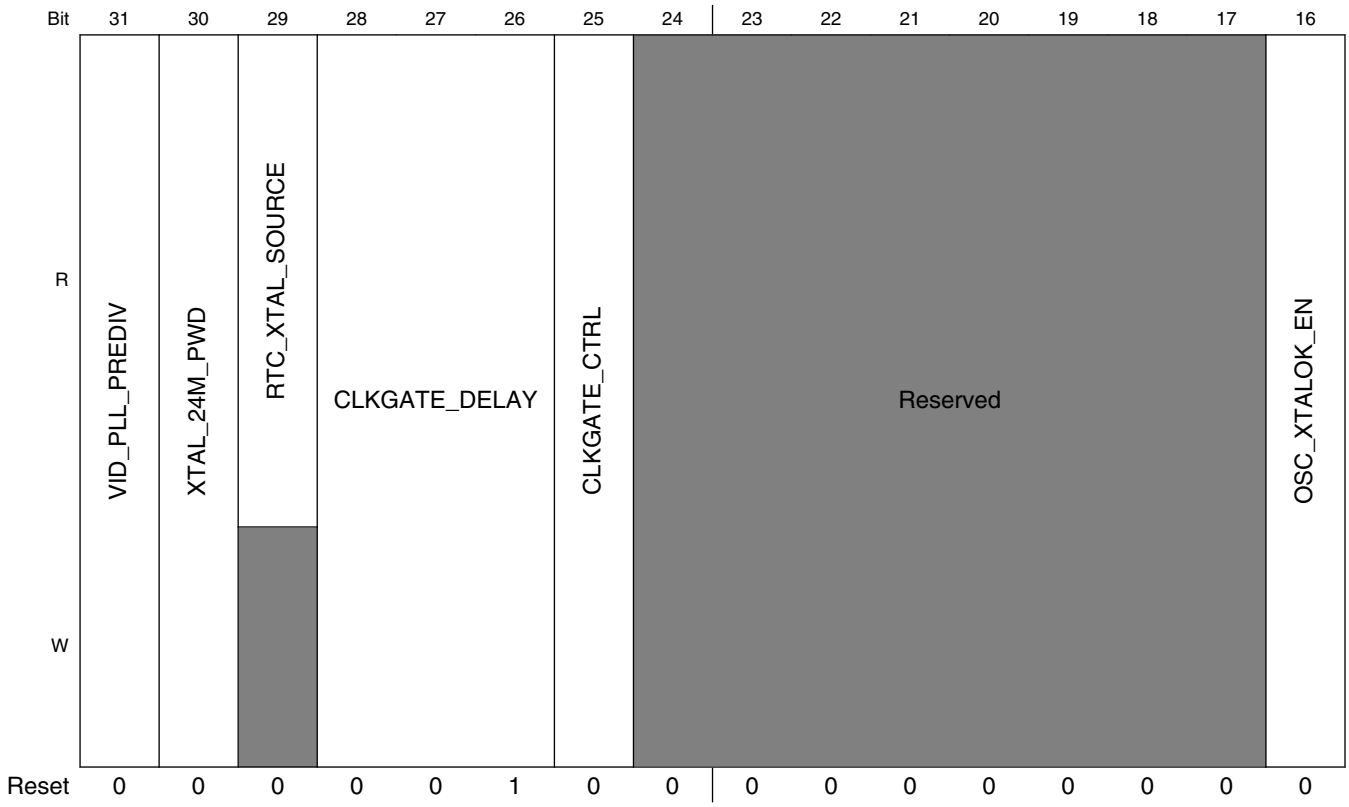
PMU_REG_CORE*n* field descriptions (continued)

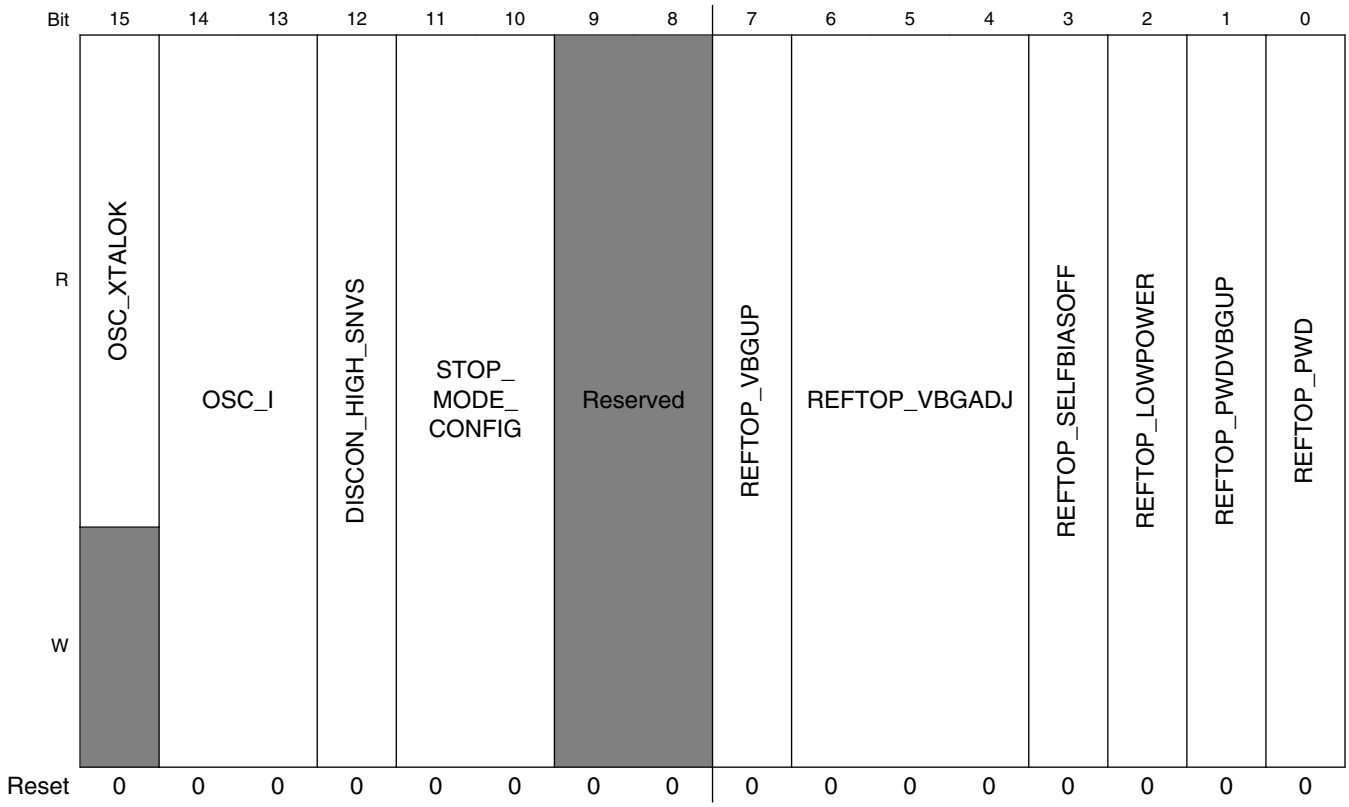
Field	Description
	00011 Target core voltage = 0.775V ... 10000 Target core voltage = 1.100V ... 11110 Target core voltage = 1.450V 11111 Power FET switched full on. No regulation.

16.6.5 Miscellaneous Register 0 (PMU_MISC0*n*)

This register defines the control and status bits for miscellaneous analog blocks.

Address: 400D_8000h base + 150h offset + (4d × i), where i=0d to 3d





PMU_MISC0*n* field descriptions

Field	Description
31 VID_PLL_ PREDIV	Predivider for the source clock of the PLL's. 0 Divide by 1 1 Divide by 2
30 XTAL_24M_PWD	This field powers down the 24M crystal oscillator if set true.
29 RTC_XTAL_ SOURCE	This field indicates which chip source is being used for the rtc clock. 0 Internal ring oscillator 1 RTC_XTAL
28–26 CLKGATE_ DELAY	This field specifies the delay between powering up the XTAL 24MHz clock and releasing the clock to the digital logic inside the analog block. **NOTE:** Do not change the field during a low power event. This is not a field that the user would normally need to modify. **NOTE:** Not related to PMU. 000 0.5ms 001 1.0ms 010 2.0ms 011 3.0ms 100 4.0ms

Table continues on the next page...

PMU_MISC0*n* field descriptions (continued)

Field	Description
	101 5.0ms 110 6.0ms 111 7.0ms
25 CLKGATE_CTRL	This bit allows disabling the clock gate (always ungated) for the xtal 24MHz clock that clocks the digital logic in the analog block. **NOTE:** Do not change the field during a low power event. This is not a field that the user would normally need to modify. **NOTE:** Not related to PMU. 0 **ALLOW_AUTO_GATE** — Allow the logic to automatically gate the clock when the XTAL is powered down. 1 **NO_AUTO_GATE** — Prevent the logic from ever gating off the clock.
24–17 -	This field is reserved. Always set to zero.
16 OSC_XTALOK_EN	This bit enables the detector that signals when the 24MHz crystal oscillator is stable. **NOTE:** Not related to PMU, Clocking content
15 OSC_XTALOK	Status bit that signals that the output of the 24-MHz crystal oscillator is stable. Generated from a timer and active detection of the actual frequency. **NOTE:** Not related to PMU, clocking content.
14–13 OSC_I	This field determines the bias current in the 24MHz oscillator. The aim is to start up with the highest bias current, which can be decreased after startup if it is determined to be acceptable. **NOTE:** Not related to PMU. 00 **NOMINAL** — Nominal 01 **MINUS_12_5_PERCENT** — Decrease current by 12.5% 10 **MINUS_25_PERCENT** — Decrease current by 25.0% 11 **MINUS_37_5_PERCENT** — Decrease current by 37.5%
12 DISCON_HIGH_SNVS	This bit controls a switch from VDD_HIGH_IN to VDD_SNVS_IN. 0 Turn on the switch 1 Turn off the switch
11–10 STOP_MODE_CONFIG	Configure the analog behavior in stop mode. 00 **SUSPEND (DSM)** — All analog except rtc powered down on stop mode assertion. 01 **STANDBY** — Analog regulators are ON. 10 **STOP (lower power)** — Analog regulators are ON. 11 **STOP (very lower power)** — Analog regulators are OFF.
9–8 -	This field is reserved. Reserved
7 REFTOP_VBGUP	Status bit that signals the analog bandgap voltage is up and stable. 1 - Stable.

Table continues on the next page...

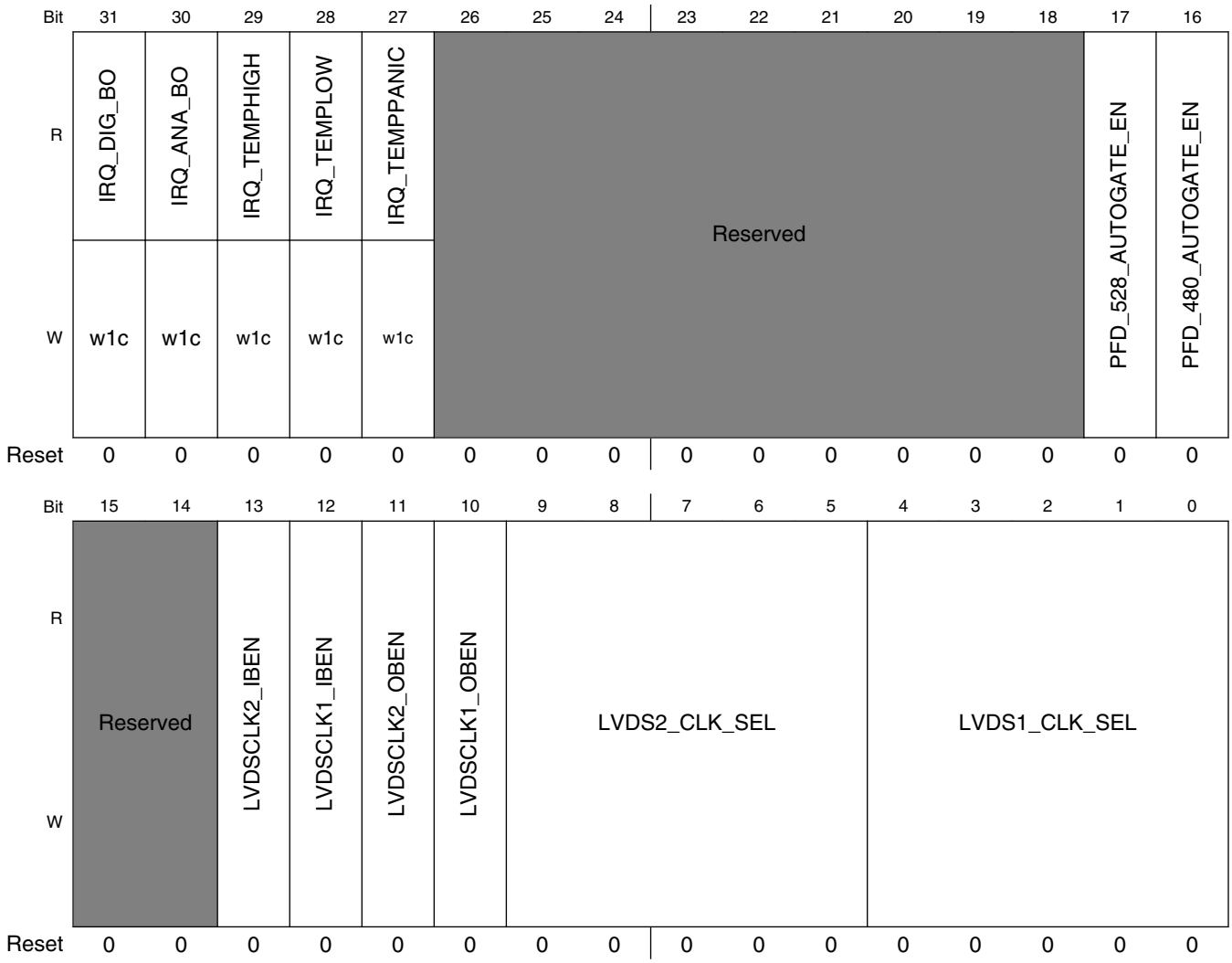
PMU_MISC0*n* field descriptions (continued)

Field	Description
6–4 REFTOP_ VBGADJ	000 Nominal VBG 001 VBG+0.78% 010 VBG+1.56% 011 VBG+2.34% 100 VBG-0.78% 101 VBG-1.56% 110 VBG-2.34% 111 VBG-3.12%
3 REFTOP_ SELFBIASOFF	Control bit to disable the self-bias circuit in the analog bandgap. The self-bias circuit is used by the bandgap during startup. This bit should be set after the bandgap has stabilized and is necessary for best noise performance of analog blocks using the outputs of the bandgap. **NOTE:** Value should be returned to zero before removing vddhigh_in or asserting bit 0 of this register (REFTOP_PWD) to assure proper restart of the circuit. 0 Uses coarse bias currents for startup 1 Uses bandgap-based bias currents for best performance.
2 REFTOP_ LOWPOWER	Control bit to enable the low-power mode in the analog bandgap.
1 REFTOP_ PWDVBGUP	Control bit to power down the VBG-up detection circuitry in the analog bandgap.
0 REFTOP_PWD	Control bit to power-down the analog bandgap reference circuitry. **NOTE:** A note of caution, the bandgap is necessary for correct operation of most of the LDO, pll, and other analog functions on the die.

16.6.6 Miscellaneous Register 1 (PMU_MISC1*n*)

This register defines the control and status bits for miscellaneous analog blocks.

Address: 400D_8000h base + 160h offset + (4d × i), where i=0d to 3d



PMU_MISC1*n* field descriptions

Field	Description
31 IRQ_DIG_BO	This status bit is set to one when when any of the digital regulator brownout interrupts assert. Check the regulator status bits to discover which regulator interrupt asserted.
30 IRQ_ANA_BO	This status bit is set to one when when any of the analog regulator brownout interrupts assert. Check the regulator status bits to discover which regulator interrupt asserted.
29 IRQ_TEMPHIGH	This status bit is set to one when the temperature sensor high interrupt asserts for high temperature.

Table continues on the next page...

PMU_MISC1*n* field descriptions (continued)

Field	Description
	NOTE: Not related to PMU, Temperature Monitor content.
28 IRQ_TEMPLOW	This status bit is set to one when the temperature sensor low interrupt asserts for low temperature. **NOTE:** Not related to PMU, Temperature Monitor content.
27 IRQ_ TEMPPANIC	This status bit is set to one when the temperature sensor panic interrupt asserts for a panic high temperature. **NOTE:** Not related to PMU, Temperature Monitor content.
26–18 -	This field is reserved. Reserved
17 PFD_528_ AUTOGATE_EN	This enables a feature that will clkgate (reset) all PFD_528 clocks anytime the PLL_528 is unlocked or powered off.
16 PFD_480_ AUTOGATE_EN	This enables a feature that will clkgate (reset) all PFD_480 clocks anytime the USB1_PLL_480 is unlocked or powered off.
15–14 -	This field is reserved. Reserved
13 LVDSCLK2_ IBEN	This enables the LVDS input buffer for anaclk2/2b. Do not enable input and output buffers simultaneously. **NOTE:** Not related to PMU.
12 LVDSCLK1_ IBEN	This enables the LVDS input buffer for anaclk1/1b. Do not enable input and output buffers simultaneously. **NOTE:** Not related to PMU, Clocking content.
11 LVDSCLK2_ OBEN	This enables the LVDS output buffer for anaclk2/2b. Do not enable input and output buffers simultaneously. **NOTE:** Not related to PMU.
10 LVDSCLK1_ OBEN	This enables the LVDS output buffer for anaclk1/1b. Do not enable input and output buffers simultaneously. **NOTE:** Not related to PMU, clocking content.
9–5 LVDS2_CLK_ SEL	This field selects the clk to be routed to anaclk2/2b. **NOTE:** Not related to PMU. 00000 **ARM_PLL** — ARM PLL 00001 **SYS_PLL** — System PLL 00010 **PFD4** — ref_pfd4_clk == pll2_pfd0_clk 00011 **PFD5** — ref_pfd5_clk == pll2_pfd1_clk 00100 **PFD6** — ref_pfd6_clk == pll2_pfd2_clk 00101 **PFD7** — ref_pfd7_clk == pll2_pfd3_clk 00110 **AUDIO_PLL** — Audio PLL 00111 **VIDEO_PLL** — Video PLL

Table continues on the next page...

PMU_MISC1*n* field descriptions (continued)

Field	Description
	01000 **MLB_PLL** — MLB PLL 01001 **ETHERNET_REF** — ethernet ref clock (ENET_PLL) 01010 **PCIE_REF** — PCIe ref clock (125M) 01011 **SATA_REF** — SATA ref clock (100M) 01100 **USB1_PLL** — USB1 PLL clock 01101 **USB2_PLL** — USB2 PLL clock 01110 **PFD0** — ref_pfd0_clk == pll3_pfd0_clk 01111 **PFD1** — ref_pfd1_clk == pll3_pfd1_clk 10000 **PFD2** — ref_pfd2_clk == pll3_pfd2_clk 10001 **PFD3** — ref_pfd3_clk == pll3_pfd3_clk 10010 **XTAL** — xtal (24M) 10011 **LVDS1** — LVDS1 (loopback) 10100 **LVDS2** — LVDS2 (not useful) 10101 to 11111 **-** — ref_pfd7_clk == pll2_pfd3_clk
LVDS1_CLK_SEL	This field selects the clk to be routed to anaclk1/1b. **NOTE:** Not related to PMU. 00000 **ARM_PLL** — ARM PLL 00001 **SYS_PLL** — System PLL 00010 **PFD4** — ref_pfd4_clk == pll2_pfd0_clk 00011 **PFD5** — ref_pfd5_clk == pll2_pfd1_clk 00100 **PFD6** — ref_pfd6_clk == pll2_pfd2_clk 00101 **PFD7** — ref_pfd7_clk == pll2_pfd3_clk 00110 **AUDIO_PLL** — Audio PLL 00111 **VIDEO_PLL** — Video PLL 01001 **ETHERNET_REF** — ethernet ref clock (ENET_PLL) 01100 **USB1_PLL** — USB1 PLL clock 01101 **USB2_PLL** — USB2 PLL clock 01110 **PFD0** — ref_pfd0_clk == pll3_pfd0_clk 01111 **PFD1** — ref_pfd1_clk == pll3_pfd1_clk 10000 **PFD2** — ref_pfd2_clk == pll3_pfd2_clk 10001 **PFD3** — ref_pfd3_clk == pll3_pfd3_clk 10010 **XTAL** — xtal (24M) 10101 to 11111 **-** — ref_pfd7_clk == pll2_pfd3_clk

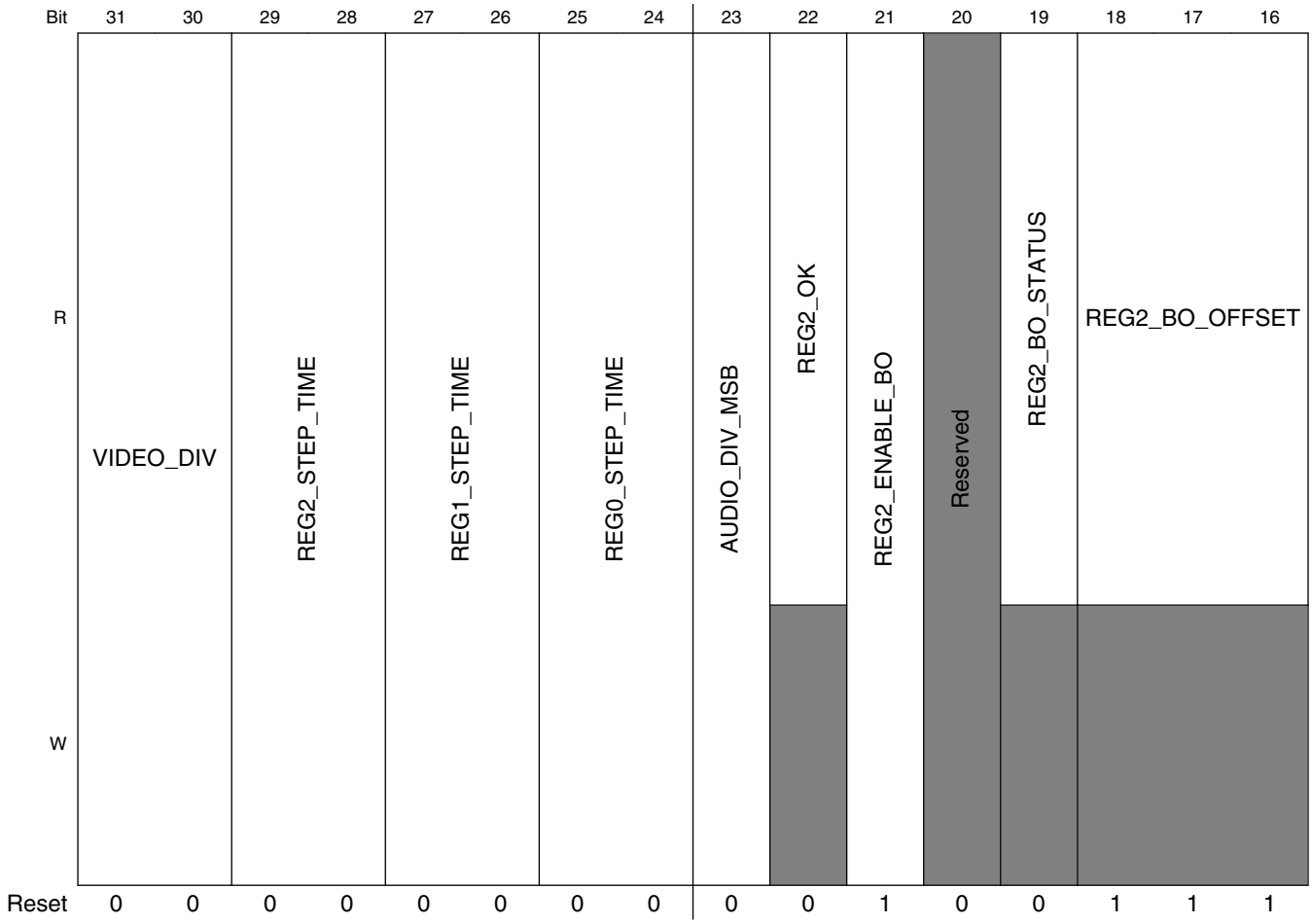
16.6.7 Miscellaneous Control Register (PMU_MISC2*n*)

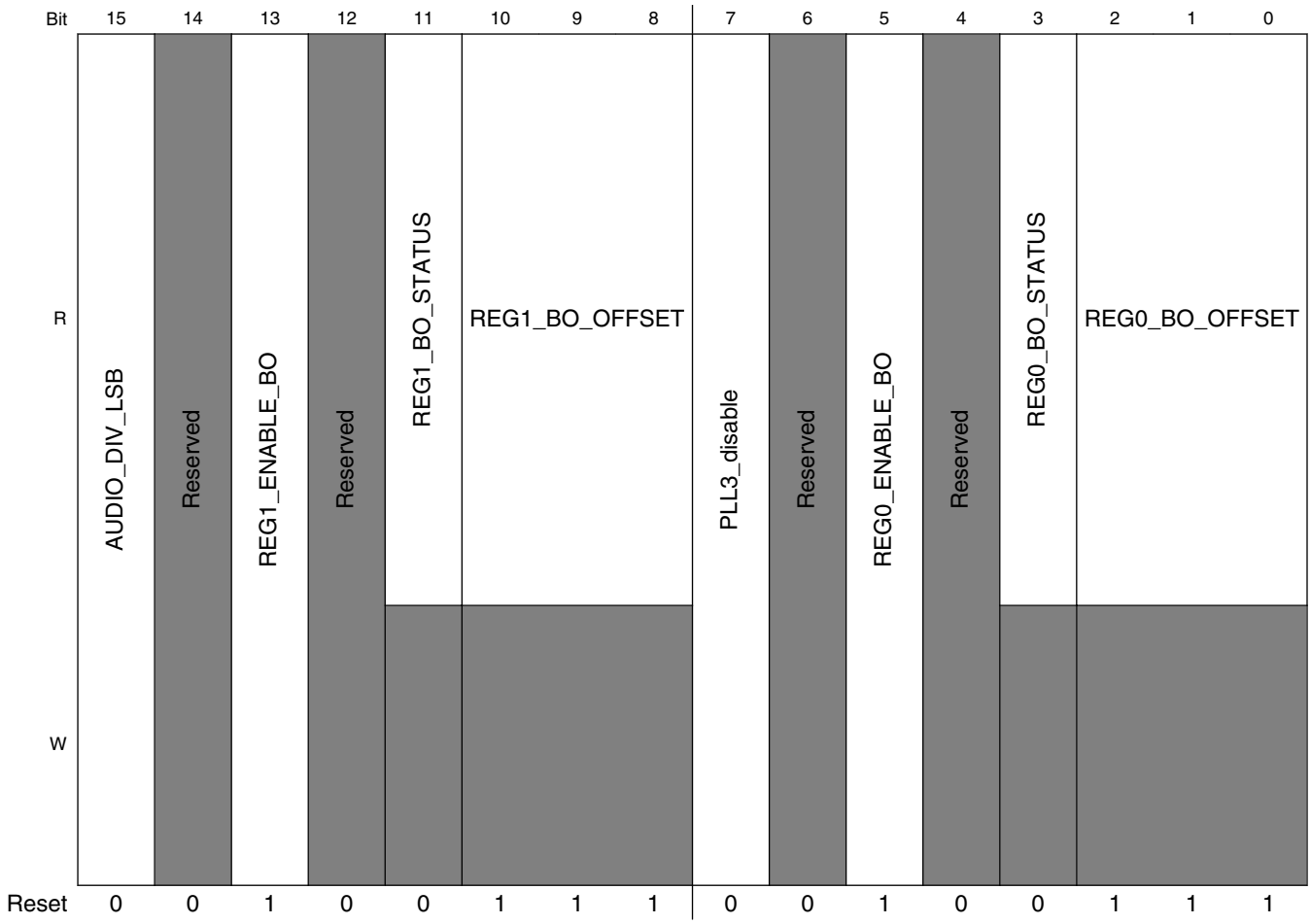
This register defines the control for miscellaneous PMU Analog blocks.

NOTE

This register is shared with CCM.

Address: 400D_8000h base + 170h offset + (4d × i), where i=0d to 3d





PMU_MISC2*n* field descriptions

Field	Description
31–30 VIDEO_DIV	Post-divider for video. The output clock of the video PLL should be gated prior to changing this divider to prevent glitches. This divider is feed by PLL_VIDEOn[POST_DIV_SELECT] to achieve division ratios of /1, /2, /4, /8, and /16. **NOTE:** Not related to PMU. See Clock Controller Module (CCM) for more information. 00 divide by 1 (Default) 01 divide by 2 10 divide by 1 11 divide by 4
29–28 REG2_STEP_TIME	Number of clock periods (24MHz clock). 00 **64_CLOCKS** — 64 01 **128_CLOCKS** — 128 10 **256_CLOCKS** — 256 11 **512_CLOCKS** — 512
27–26 REG1_STEP_TIME	Number of clock periods (24MHz clock). 00 **64_CLOCKS** — 64 01 **128_CLOCKS** — 128

Table continues on the next page...

PMU_MISC2*n* field descriptions (continued)

Field	Description
	10 **256_CLOCKS** — 256 11 **512_CLOCKS** — 512
25–24 REG0_STEP_TIME	Number of clock periods (24MHz clock). 00 **64_CLOCKS** — 64 01 **128_CLOCKS** — 128 10 **256_CLOCKS** — 256 11 **512_CLOCKS** — 512
23 AUDIO_DIV_MSB	MSB of Post-divider for Audio PLL. The output clock of the video PLL should be gated prior to changing this divider to prevent glitches. This divider is feed by PLL_AUDIOn[POST_DIV_SELECT] to achieve division ratios of /1, /2, /4, /8, and /16. **NOTE:** MSB bit value pertains to the first bit, please program the LSB bit (bit 15) as well to change divider value **NOTE:** Not related to PMU. See Clock Controller Module (CCM) for more information. 00 divide by 1 (Default) 01 divide by 2 10 divide by 1 11 divide by 4
22 REG2_OK	Signals that the voltage is above the brownout level for the SOC supply. 1 = regulator output > brownout_target
21 REG2_ENABLE_BO	Enables the brownout detection.
20 -	This field is reserved.
19 REG2_BO_STATUS	Reg2 brownout status bit.
18–16 REG2_BO_OFFSET	This field defines the brown out voltage offset for the xPU power domain. IRQ_DIG_BO is also asserted. Single-bit increments reflect 25mV brownout voltage steps. The reset brown-offset is 175mV below the programmed target code. Brownout target = OUTPUT_TRG - BO_OFFSET. Some steps may be irrelevant because of input supply limitations or load operation. 100 Brownout offset = 0.100V 111 Brownout offset = 0.175V
15 AUDIO_DIV_LSB	LSB of Post-divider for Audio PLL. The output clock of the video PLL should be gated prior to changing this divider to prevent glitches. This divider is feed by PLL_AUDIOn[POST_DIV_SELECT] to achieve division ratios of /1, /2, /4, /8, and /16. **NOTE:** LSB bit value pertains to the last bit, please program the MSB bit (bit 23) as well, to change divider value **NOTE:** Not related to PMU. See Clock Controller Module (CCM) for more information. 00 divide by 1 (Default) 01 divide by 2 10 divide by 1 11 divide by 4

Table continues on the next page...

PMU_MISC2*n* field descriptions (continued)

Field	Description
14 -	This field is reserved. Reserved
13 REG1_ENABLE_BO	Enables the brownout detection.
12 -	This field is reserved.
11 REG1_BO_STATUS	Reg1 brownout status bit. 1 Brownout, supply is below target minus brownout offset.
10–8 REG1_BO_OFFSET	This field defines the brown out voltage offset for the xPU power domain. IRQ_DIG_BO is also asserted. Single-bit increments reflect 25mV brownout voltage steps. The reset brown-offset is 175mV below the programmed target code. Brownout target = OUTPUT_TRG - BO_OFFSET. Some steps may be irrelevant because of input supply limitations or load operation. 100 Brownout offset = 0.100V 111 Brownout offset = 0.175V
7 PLL3_disable	Default value of "0". Should be set to "1" to turn off the USB-PLL(PLL3) in run mode. **NOTE:** Not related to PMU. See Clock Controller Module (CCM) for more information.
6 -	This field is reserved.
5 REG0_ENABLE_BO	Enables the brownout detection.
4 -	This field is reserved.
3 REG0_BO_STATUS	Reg0 brownout status bit. 1 Brownout, supply is below target minus brownout offset.
REG0_BO_OFFSET	This field defines the brown out voltage offset for the CORE power domain. IRQ_DIG_BO is also asserted. Single-bit increments reflect 25mV brownout voltage steps. Some steps may be irrelevant because of input supply limitations or load operation. 100 Brownout offset = 0.100V 111 Brownout offset = 0.175V

Chapter 17
General Power Controller (GPC)

17.1 Chip-specific GPC information

Table 17-1. Reference links to related information

Topic	Related module(s)	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
Interrupts, DMA Events and XBAR Assignments	-	Interrupts, DMA Events and XBAR Assignments

17.2 Overview

The General Power Control (GPC) block includes the sub-blocks listed here.

- CPU Power Gating Control (PGC)

Each sub-block has its own IP registers.

GPC determines wake-up IRQ for exiting WAIT/STOP mode (with or without CPU power gating).

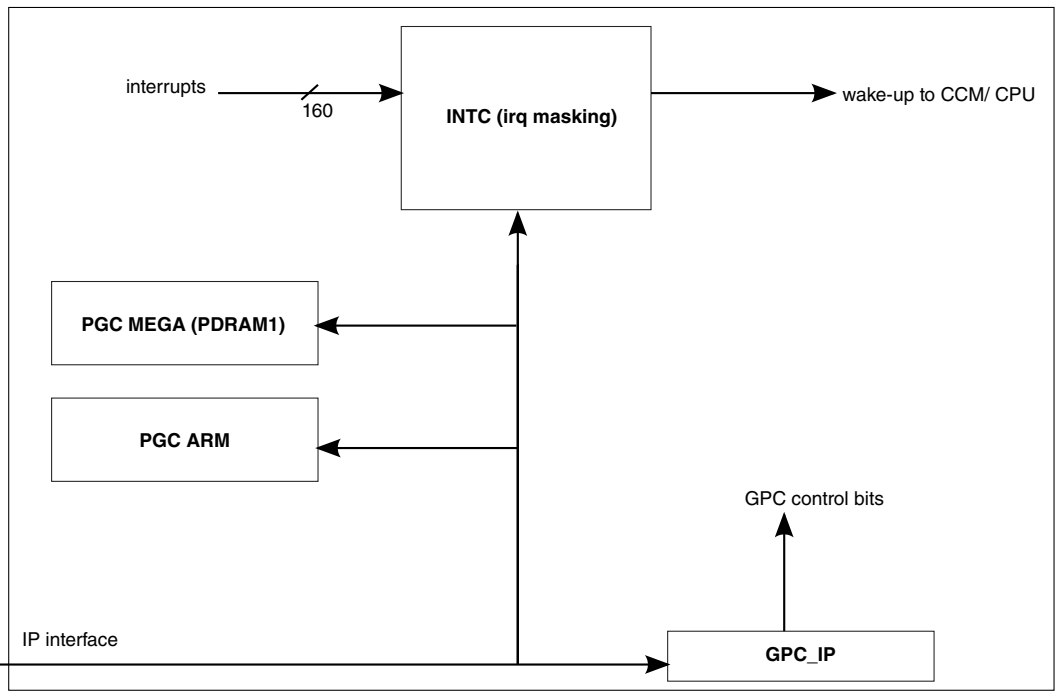


Figure 17-1. GPC Block Diagram

17.3 Clocks

The table found here describes the clock sources for GPC.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 17-2. GPC Clocks

Clock name	Clock Root	Description
ipg_clk	ipg_clk_root	Peripheral clock
ipg_clk_s	ipg_clk_root	Peripheral access clock
pgc_clk	ipg_clk_root	PGC peripheral clock
sys_clk	ipg_clk_root	Module clock

17.4 Power Gating Control (PGC)

Power Gating (PGC) is applied to the Arm CPU and FlexRAM optionally in STOP low power mode, after all essential CPU registers data are saved by Arm dormant procedure.

If any of the unmasked interrupts appears, CPU is powered up and clock restore request (exit from STOP mode) is sent to CCM.

PGC power down sequence:

- CCM sends power down request when the chip is about to enter stop mode. The user should define which modules will be powered down (CTRL registers of corresponding PGC module, bit 0).

PGC power up sequence:

- One of the power up irq is asserted.
- Power up request is asserted in GPC and in CCM.
- The Power Gated modules are powered up, according to PGC settings of appropriate module.

The Power Gated modules require reset after powering up. The next figure describes GPC-SRC handshake procedure for reset after power gating.

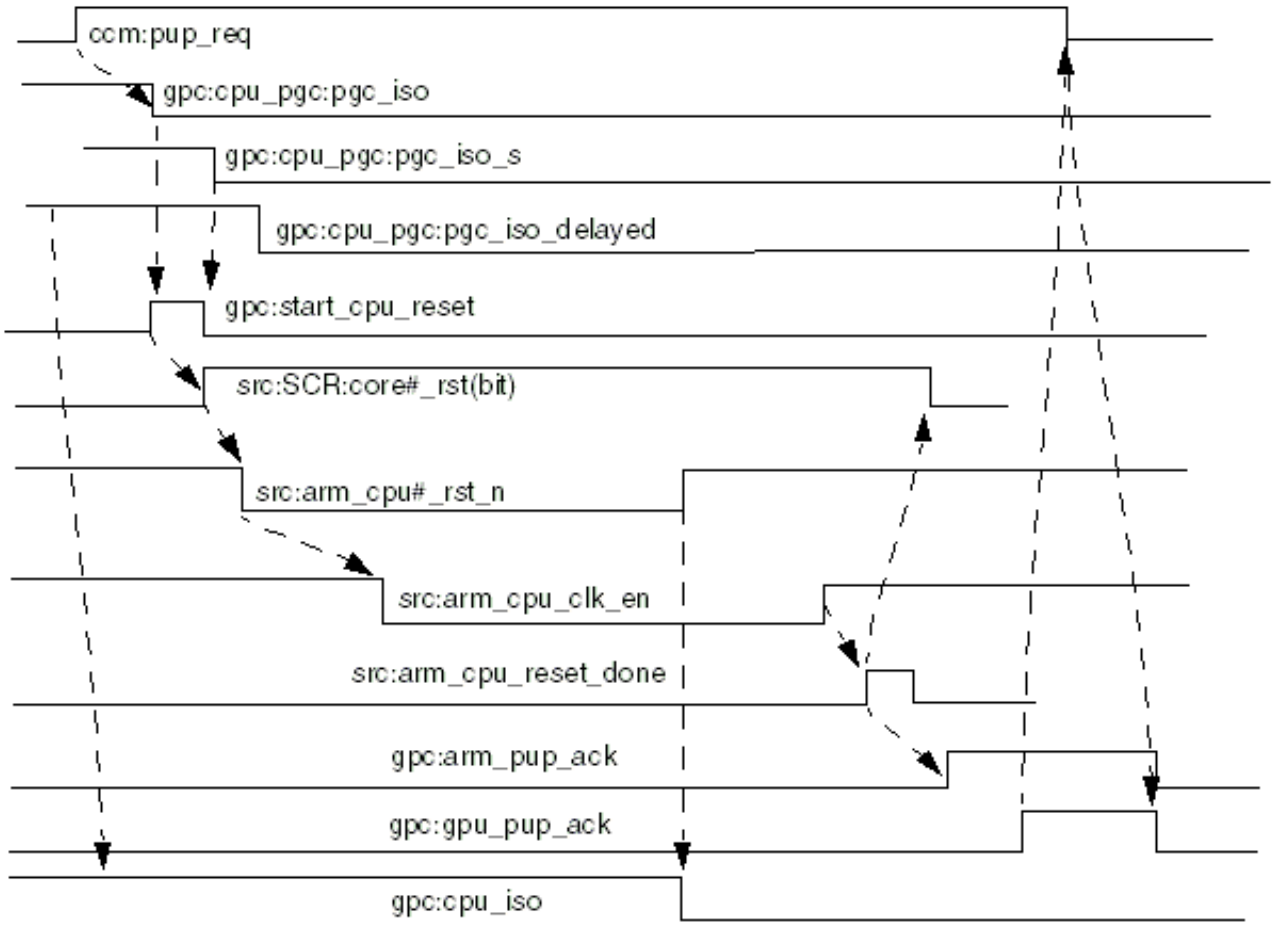


Figure 17-2. GPC-SRC handshake for reset after power gating

17.4.1 Overview

The Power Gating Controller (PGC) is a power management component that controls the power-down and power-up sequencing of individual subsystems.

The sequence timing is programmable using the PGC control registers. Figure 17-3 shows PGC as part of the SoC's overall power management scheme.

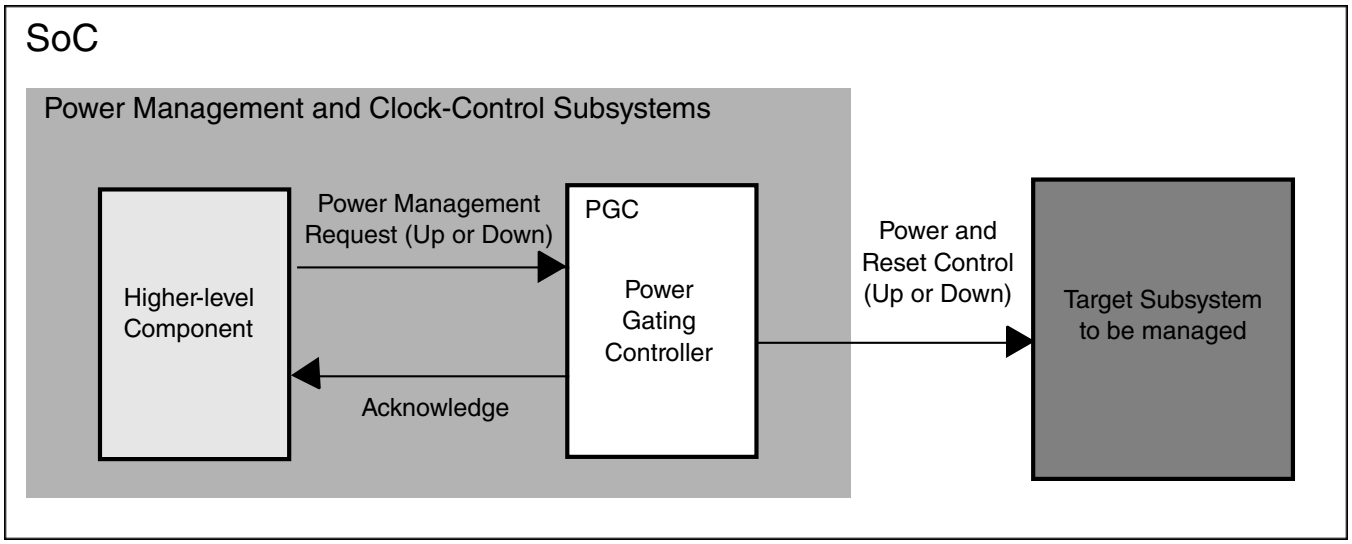


Figure 17-3. PGC Block Diagram

17.4.1.1 Features

Key features of the PGC include:

- Provides the ability to switch off power to a target subsystem.
- Generates power-up and power-down control sequences.
- Provides programmable registers to adjust the timing of the power control signals.

17.5 GPC Interrupt Controller (INTC)

The INTC (Interrupt Controller) detects an interrupt and generates the wakeup signal. It supports up to 160 interrupts.

17.5.1 Interrupt Controller features

The features of the GPC INTC are listed below.

Features:

- Supports up to 160 interrupts
- Provides an option to mask/unmask each interrupt
- Detects interrupts and generates the wake up signal
- 32-bits IP bus interface
- All registers are byte-accessible

17.6 GPC Memory Map/Register Definition

Detailed descriptions of each register can be found below.

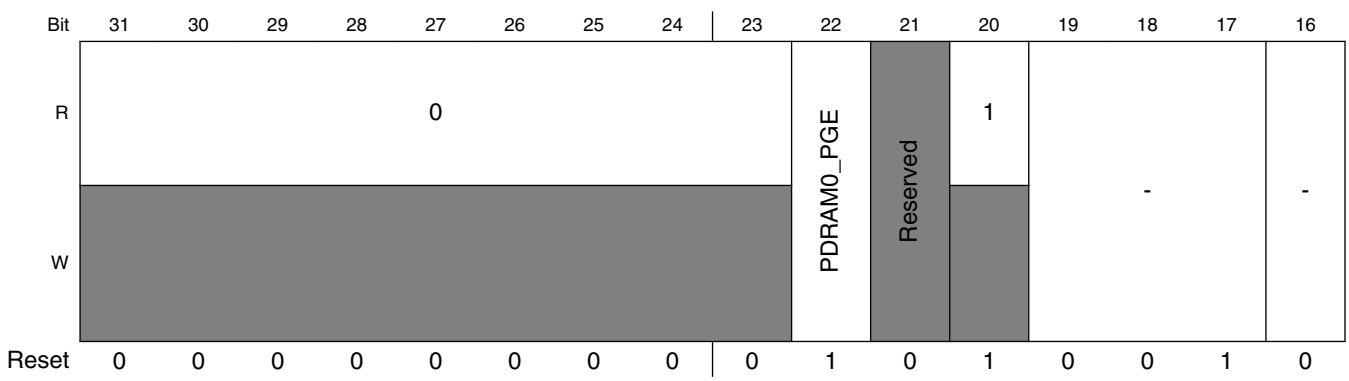
Writes to GPC registers only takes effect in supervisor mode.

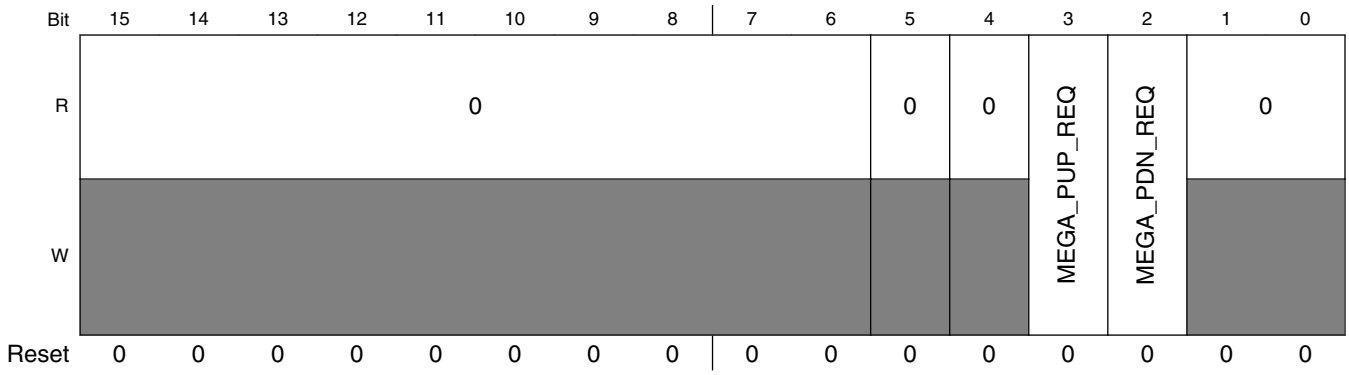
GPC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400F_4000	GPC Interface control register (GPC_CNTR)	32	R/W	0052_0000h	17.6.1/1184
400F_4008	IRQ masking register 1 (GPC_IMR1)	32	R/W	0000_0000h	17.6.2/1186
400F_400C	IRQ masking register 2 (GPC_IMR2)	32	R/W	0000_0000h	17.6.3/1186
400F_4010	IRQ masking register 3 (GPC_IMR3)	32	R/W	0000_0000h	17.6.4/1187
400F_4014	IRQ masking register 4 (GPC_IMR4)	32	R/W	0000_0000h	17.6.5/1187
400F_4018	IRQ status resister 1 (GPC_ISR1)	32	R	0000_0000h	17.6.6/1188
400F_401C	IRQ status resister 2 (GPC_ISR2)	32	R	0000_0000h	17.6.7/1188
400F_4020	IRQ status resister 3 (GPC_ISR3)	32	R	0000_0000h	17.6.8/1189
400F_4024	IRQ status resister 4 (GPC_ISR4)	32	R	0000_0000h	17.6.9/1189
400F_4034	IRQ masking register 5 (GPC_IMR5)	32	R/W	0000_0000h	17.6.10/ 1190
400F_4038	IRQ status resister 5 (GPC_ISR5)	32	R	0000_0000h	17.6.11/ 1190

17.6.1 GPC Interface control register (GPC_CNTR)

Address: 400F_4000h base + 0h offset = 400F_4000h





GPC_CNTR field descriptions

Field	Description
31–23 Reserved	This read-only field is reserved and always has the value 0.
22 PDRAM0_PGE	FlexRAM PDRAM0 Power Gate Enable 1 FlexRAM PDRAM0 domain will be powered down when the CPU core is powered down.. 0 FlexRAM PDRAM0 domain will keep power even if the CPU core is powered down.
21 -	This field is reserved. Reserved
20 Reserved	This read-only field is reserved and always has the value 1.
19–17 -	Reserved This bitfield is readable/writeable, but avoid writing to this reserved bitfield.
16 -	Reserved This bitfield is readable/writeable, but avoid writing to this reserved bitfield.
15–6 Reserved	This read-only field is reserved and always has the value 0.
5 Reserved	This read-only field is reserved and always has the value 0.
4 Reserved	This read-only field is reserved and always has the value 0.
3 MEGA_PUP_REQ	MEGA domain (FlexRAM PDRAM1) power up request. Self-clear bit. **Writable but read will always return 0.** **NOTE:** Software can directly power up FlexRAM PDRAM1 by writing this bit, given PGC_MEGA_CTRL[PCR] is set and FlexRAM PDRAM1 has been powered down by setting MEGA_PDN_REQ. 0 — No Request 1 — Request power up sequence
2 MEGA_PDN_REQ	MEGA domain (FlexRAM PDRAM1) power down request. Self-clear bit. **Writable but read will always return 0.** **NOTE:** Software can directly power down FlexRAM PDRAM1 by writing this bit, given PGC_MEGA_CTRL[PCR] is set.

Table continues on the next page...

GPC_CNTR field descriptions (continued)

Field	Description
	0 — No Request 1 — Request power down sequence
Reserved	This read-only field is reserved and always has the value 0.

17.6.2 IRQ masking register 1 (GPC_IMR1)

IMR1 Register - masking of irq[31:0].

Address: 400F_4000h base + 8h offset = 400F_4008h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IMR1																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPC_IMR1 field descriptions

Field	Description
IMR1	IRQ[31:0] masking bits: 1-irq masked, 0-irq is not masked

17.6.3 IRQ masking register 2 (GPC_IMR2)

IMR2 Register - masking of irq[63:32].

Address: 400F_4000h base + Ch offset = 400F_400Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IMR2																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPC_IMR2 field descriptions

Field	Description
IMR2	IRQ[63:32] masking bits: 1-irq masked, 0-irq is not masked

17.6.4 IRQ masking register 3 (GPC_IMR3)

IMR3 Register - masking of irq[95:64].

Address: 400F_4000h base + 10h offset = 400F_4010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IMR3																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPC_IMR3 field descriptions

Field	Description
IMR3	IRQ[95:64] masking bits: 1-irq masked, 0-irq is not masked

17.6.5 IRQ masking register 4 (GPC_IMR4)

IMR4 Register - masking of irq[127:96].

Address: 400F_4000h base + 14h offset = 400F_4014h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IMR4																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

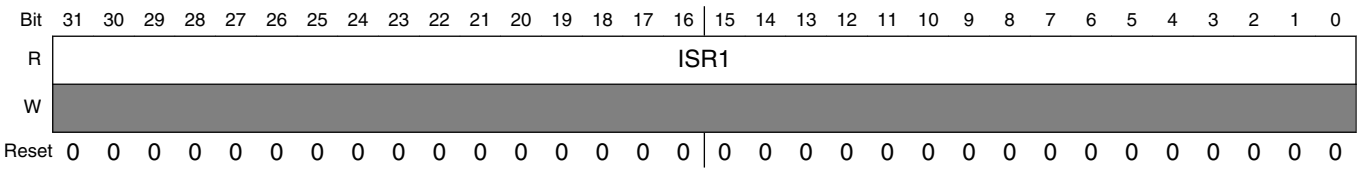
GPC_IMR4 field descriptions

Field	Description
IMR4	IRQ[127:96] masking bits: 1-irq masked, 0-irq is not masked

17.6.6 IRQ status resister 1 (GPC_ISR1)

ISR1 Register - status of irq [31:0].

Address: 400F_4000h base + 18h offset = 400F_4018h



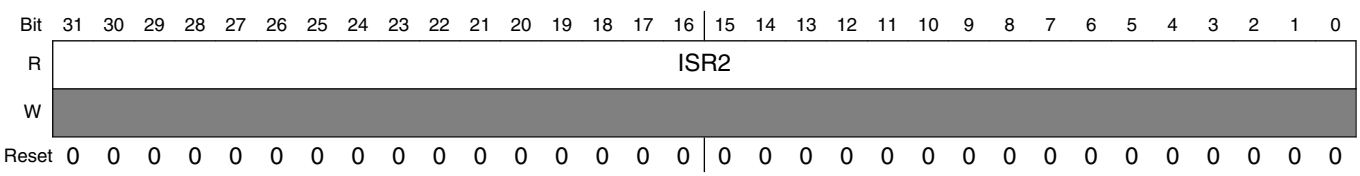
GPC_ISR1 field descriptions

Field	Description
ISR1	IRQ[31:0] status, read only

17.6.7 IRQ status resister 2 (GPC_ISR2)

ISR2 Register - status of irq [63:32].

Address: 400F_4000h base + 1Ch offset = 400F_401Ch



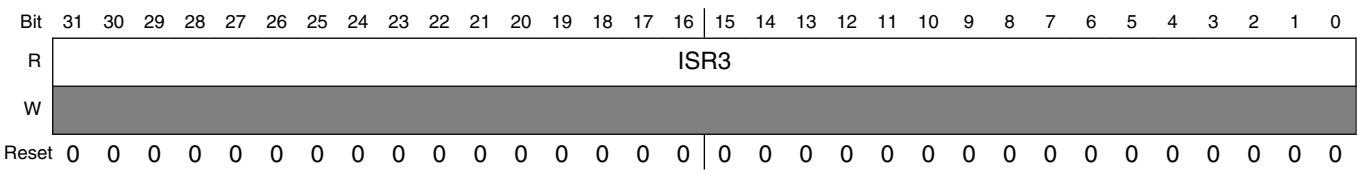
GPC_ISR2 field descriptions

Field	Description
ISR2	IRQ[63:32] status, read only

17.6.8 IRQ status resister 3 (GPC_ISR3)

ISR3 Register - status of irq [95:64].

Address: 400F_4000h base + 20h offset = 400F_4020h



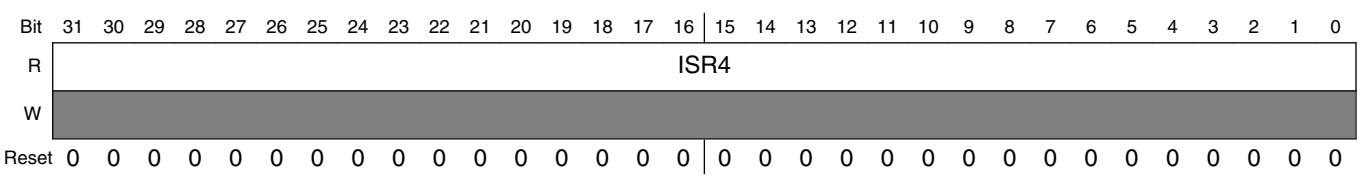
GPC_ISR3 field descriptions

Field	Description
ISR3	IRQ[95:64] status, read only

17.6.9 IRQ status resister 4 (GPC_ISR4)

ISR4 Register - status of irq [127:96].

Address: 400F_4000h base + 24h offset = 400F_4024h



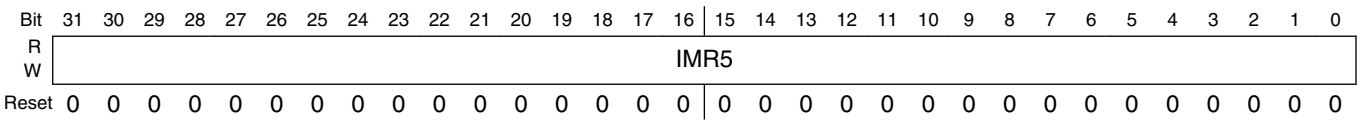
GPC_ISR4 field descriptions

Field	Description
ISR4	IRQ[127:96] status, read only

17.6.10 IRQ masking register 5 (GPC_IMR5)

IMR5 Register - masking of irq[159:128].

Address: 400F_4000h base + 34h offset = 400F_4034h



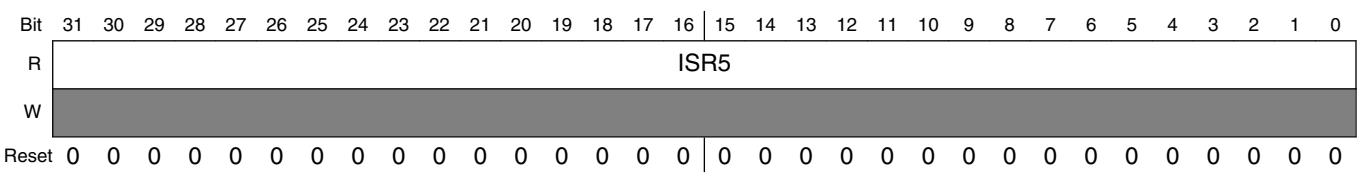
GPC_IMR5 field descriptions

Field	Description
IMR5	IRQ[159:128] masking bits: 1-irq masked, 0-irq is not masked

17.6.11 IRQ status resister 5 (GPC_ISR5)

ISR5 Register - status of irq [159:128].

Address: 400F_4000h base + 38h offset = 400F_4038h



GPC_ISR5 field descriptions

Field	Description
ISR5	IRQ[159:128] status, read only

17.7 PGC Memory Map/Register Definition

The PGC registers can be accessed only in supervisor mode.

Attempts to access registers when not in supervisor mode or attempts to access an unimplemented address location might trigger a bus transfer error. In this case, software should take appropriate action (such as ignore the error, log the error, or initiate a soft reset).

All PGC registers are byte-accessible.

NOTE

PGC MEGA is used to control the power gate of PDRAM1 domain . PGC CPU is used to control the power gate of PDM7 domain (CM7 CPU). FlexRAM PDRAM0 domain is further controlled by GPC_CNTR[PDRAM0_PGE] bit.

PGC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400F_4220	PGC Mega Control Register (PGC_MEGA_CTRL)	32	R/W	0000_0000h	17.7.1/1192
400F_4224	PGC Mega Power Up Sequence Control Register (PGC_MEGA_PUPSCR)	32	R/W	0000_0F01h	17.7.2/1192
400F_4228	PGC Mega Pull Down Sequence Control Register (PGC_MEGA_PDNSCR)	32	R/W	0000_0101h	17.7.3/1193
400F_422C	PGC Mega Power Gating Controller Status Register (PGC_MEGA_SR)	32	R/W	0000_0000h	17.7.4/1194
400F_42A0	PGC CPU Control Register (PGC_CPU_CTRL)	32	R/W	0000_0000h	17.7.5/1194
400F_42A4	PGC CPU Power Up Sequence Control Register (PGC_CPU_PUPSCR)	32	R/W	0000_0F01h	17.7.6/1195
400F_42A8	PGC CPU Pull Down Sequence Control Register (PGC_CPU_PDNSCR)	32	R/W	0000_0101h	17.7.7/1196
400F_42AC	PGC CPU Power Gating Controller Status Register (PGC_CPU_SR)	32	R/W	0000_0000h	17.7.8/1196

17.7.1 PGC Mega Control Register (PGC_MEGA_CTRL)

The PGCR enables the response to a power-down request.

Address: 400F_4000h base + 220h offset = 400F_4220h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0															PCR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PGC_MEGA_CTRL field descriptions

Field	Description
31–1 Reserved	This read-only field is reserved and always has the value 0.
0 PCR	Power Control **NOTE:** PCR must not change from power-down request (pdn_req) assertion until the target subsystem is completely powered up. 0 Do not switch off power even if pdn_req is asserted. 1 Switch off power when pdn_req is asserted.

17.7.2 PGC Mega Power Up Sequence Control Register (PGC_MEGA_PUPSCR)

The PUPSCR contains the power-up timing parameters.

Address: 400F_4000h base + 224h offset = 400F_4224h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																SW2ISO								0		SW					
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	1

PGC_MEGA_PUPSCR field descriptions

Field	Description
31–14 Reserved	This read-only field is reserved and always has the value 0.
13–8 SW2ISO	After asserting power toggle on/off signal (switch_b) to switch on power, the PGC waits a number of IPG clocks equal to the value of SW2ISO before negating isolation. **NOTE:** SW2ISO must not be programmed to zero.
7–6 Reserved	This read-only field is reserved and always has the value 0.
SW	After a power-up request (pup_req assertion), the PGC waits a number of IPG clocks equal to the value of SW before asserting power toggle on/off signal (switch_b) to switch on power. **NOTE:** SW must not be programmed to zero. **NOTE:** The PGC clock is generated from the IPG_CLK_ROOT. for frequency configuration of the IPG_CLK_ROOT. See Clock Controller Module (CCM).

17.7.3 PGC Mega Pull Down Sequence Control Register (PGC_MEGA_PDNSCR)

The PDNSCR contains the power-down timing parameters.

Address: 400F_4000h base + 228h offset = 400F_4228h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																		ISO2SW						0		ISO					
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

PGC_MEGA_PDNSCR field descriptions

Field	Description
31–14 Reserved	This read-only field is reserved and always has the value 0.
13–8 ISO2SW	After asserting isolation, the PGC waits a number of IPG clocks equal to the value of ISO2SW before negating power toggle on/off signal (switch_b) to switch off power. **NOTE:** ISO2SW must not be programmed to zero.
7–6 Reserved	This read-only field is reserved and always has the value 0.
ISO	After a power-down request (pdn_req assertion), the PGC waits a number of IPG clocks equal to the value of ISO before asserting isolation. **NOTE:** ISO must not be programmed to zero.

17.7.4 PGC Mega Power Gating Controller Status Register (PGC_MEGA_SR)

The SR contains the status parameters.

Address: 400F_4000h base + 22Ch offset = 400F_422Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0															PSR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PGC_MEGA_SR field descriptions

Field	Description
31–1 Reserved	This read-only field is reserved and always has the value 0.
0 PSR	Power status. When in functional (or software-controlled debug) mode, PGC hardware sets PSR as soon as any of the power control output changes its state to one (isolation or power off occurs). Write one to clear this bit. Software should clear this bit after power up; otherwise, PSR continues to reflect the power status of the initial power down. 0 The target subsystem was not powered down for the previous power-down request. 1 The target subsystem was powered down for the previous power-down request.

17.7.5 PGC CPU Control Register (PGC_CPU_CTRL)

The PGCR enables the response to a power-down request.

Address: 400F_4000h base + 2A0h offset = 400F_42A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0															PCR
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

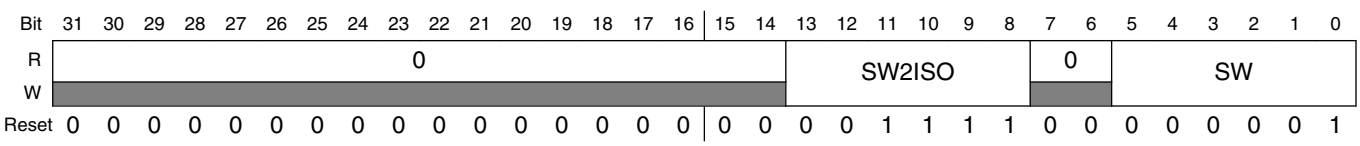
PGC_CPU_CTRL field descriptions

Field	Description
31–1 Reserved	This read-only field is reserved and always has the value 0.
0 PCR	Power Control **NOTE:** PCR must not change from power-down request (pdn_req) assertion until the target subsystem is completely powered up. 0 Do not switch off power even if pdn_req is asserted. 1 Switch off power when pdn_req is asserted.

17.7.6 PGC CPU Power Up Sequence Control Register (PGC_CPU_PUPSCR)

The PUPSCR contains the power-up timing parameters.

Address: 400F_4000h base + 2A4h offset = 400F_42A4h



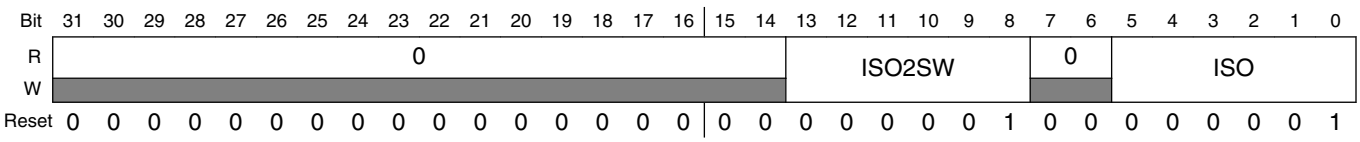
PGC_CPU_PUPSCR field descriptions

Field	Description
31–14 Reserved	This read-only field is reserved and always has the value 0.
13–8 SW2ISO	After asserting power toggle on/off signal (switch_b) to switch on power, the PGC waits a number of IPG clocks equal to the value of SW2ISO before negating isolation. **NOTE:** SW2ISO must not be programmed to zero.
7–6 Reserved	This read-only field is reserved and always has the value 0.
SW	After a power-up request (pup_req assertion), the PGC waits a number of IPG clocks equal to the value of SW before asserting power toggle on/off signal (switch_b) to switch on power. **NOTE:** SW must not be programmed to zero. **NOTE:** The PGC clock is generated from the IPG_CLK_ROOT. for frequency configuration of the IPG_CLK_ROOT. See Clock Controller Module (CCM).

17.7.7 PGC CPU Pull Down Sequence Control Register (PGC_CPU_PDNSCR)

The PDNSCR contains the power-down timing parameters.

Address: 400F_4000h base + 2A8h offset = 400F_42A8h



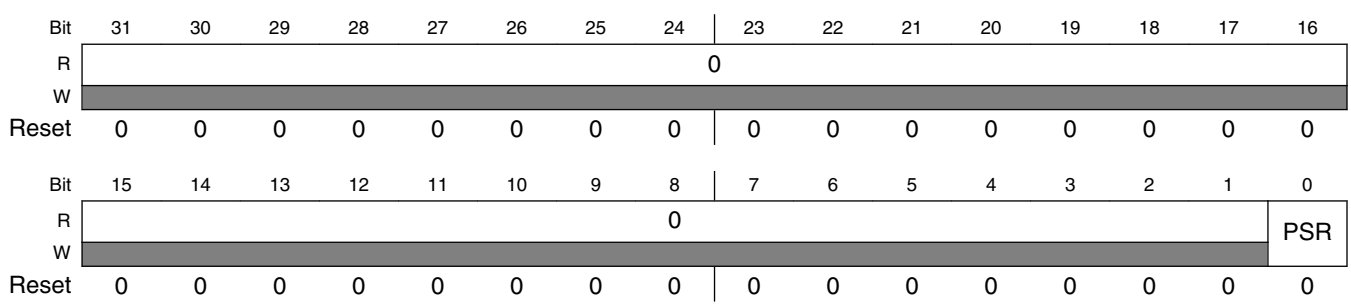
PGC_CPU_PDNSCR field descriptions

Field	Description
31–14 Reserved	This read-only field is reserved and always has the value 0.
13–8 ISO2SW	After asserting isolation, the PGC waits a number of IPG clocks equal to the value of ISO2SW before negating power toggle on/off signal (switch_b) to switch off power. **NOTE:** ISO2SW must not be programmed to zero.
7–6 Reserved	This read-only field is reserved and always has the value 0.
ISO	After a power-down request (pdn_req assertion), the PGC waits a number of IPG clocks equal to the value of ISO before asserting isolation. **NOTE:** ISO must not be programmed to zero.

17.7.8 PGC CPU Power Gating Controller Status Register (PGC_CPU_SR)

The SR contains the status parameters.

Address: 400F_4000h base + 2ACh offset = 400F_42ACh



PGC_CPU_SR field descriptions

Field	Description
31–1 Reserved	This read-only field is reserved and always has the value 0.
0 PSR	Power status. When in functional (or software-controlled debug) mode, PGC hardware sets PSR as soon as any of the power control output changes its state to one (isolation or power off occurs). Write one to clear this bit. Software should clear this bit after power up; otherwise, PSR continues to reflect the power status of the initial power down. 0 The target subsystem was not powered down for the previous power-down request. 1 The target subsystem was powered down for the previous power-down request.

Chapter 18
DCDC Converter (DCDC)

18.1 Chip-specific DCDC information

Table 18-1. Reference links to related information

Topic	Related module(s)	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
Interrupts, DMA Events and XBAR Assignments	-	Interrupts, DMA Events and XBAR Assignments

NOTE

It is recommended to use the internal DCDC as core supply for cost solution.

NOTE

When DCDC bypass mode is used, PSWITCH pin should always be tied to low. No inductor is needed, just keep DCDC_LP pin floating. DCDC_IN still needs to be powered, but it can be short with VDDsoc, not necessarily to 3.3 V.

NOTE

If DCDC bypass mode is used, it is not recommended to switch back to the internal DCDC enable mode, because the board design is different for DCDC bypass mode and internal DCDC enable mode.

NOTE

Discontinuous conduction mode can increase the efficiency of DCDC significantly in case of light current loading, so it is generally recommended. But transient performance is a little worse than continuous mode when the current loading has big increasement suddenly, where the output might has 130 mV ripple.

18.2 Overview

This module is a synchronous buck mode DCDC converter with a single output.

18.2.1 Block diagram

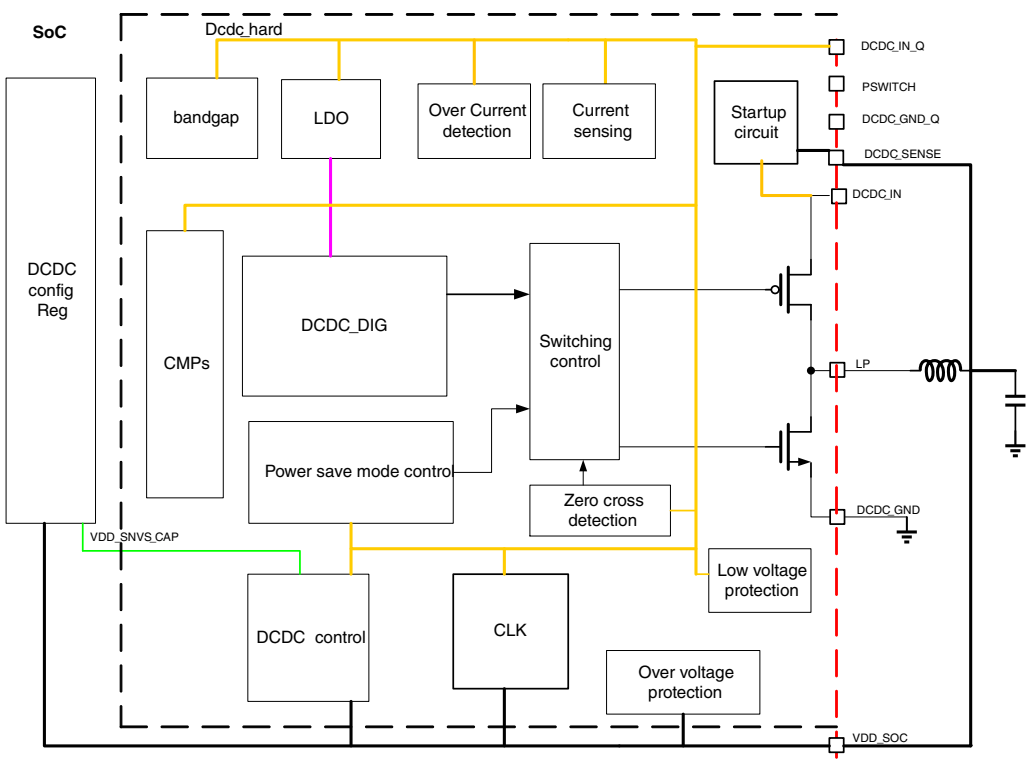


Figure 18-1. DCDC block diagram

18.2.2 Features

The DCDC module includes the following features:

- Buck mode, a single output
- PSWITCH pin to control DCDC to enable or bypass DCDC
- Continuous conduction mode (CCM) and discontinuous conduction mode (DCM) operation
- Power-save mode operation
- Over current protection
- Over voltage protection
- Low voltage detection
- Internal oscillator to support the case when the crystal oscillator is not present as in the block diagram

18.3 Functional description

The DCDC converter is a synchronous step down converter with a single output. In run mode, it can operate in the continuous conduction mode or discontinuous conduction mode, depending on the level of the load current. The load current level is about 50mA in a typical case, but it depends on the exact value of inductance, input voltage level and output target. If DCM mode is enabled, the DCDC can switch to DCM mode automatically if the current of the inductance goes to zero. The current does not need to be monitored to switch modes. Discontinuous conduction mode must be enabled by the register configuration.

The DCDC can also be configured to operate in power-save mode when the load current is less than 50 mA. During the power-save mode, the converter operates with reduced switching frequency in PFM mode and with a minimum quiescent current to maintain high efficiency.

DCM and CCM both work in run mode, in which the DCDC regulates the output continuously.

PFM is Pulse Frequency Modulation. In PFM, which is also named Pulse Mode, the DCDC is idle until the output drops to the low threshold. The DCDC then starts to charge the output to the high threshold. After it reaches the high threshold, the DCDC is idle again.

The DCDC also includes the following three protection functions:

- Over current protection. In run mode, DCDC will shut down when detecting abnormal large current in the P-type power switch. In power save mode, DCDC will stop charging inductor when detecting large current in the P-type power switch. The threshold is also different in run mode and in power save mode: the former is 1 A–2 A, and the latter is 200 mA–250 mA.

- Over voltage protection. DCDC will shut down when detecting the output voltage is too high.
- Low voltage detection. DCDC will shut down when detecting the input voltage is too low.

Another function of DCDC is that it can detect the peak current in the P-channel switch. When the peak current exceeds the threshold, DCDC will give an alert signal, and the threshold can be configured. By this way, DCDC can roughly detect the current loading.

18.4 Application information

18.4.1 Turn on DCDC

The below conditions must be met to enable the DCDC:

- Power snvs_1p0 and clk32k in snvs_1p0 are available
- DCDC_IN powered up
- After DCDC_IN powered up, delay about 1ms reset time to enable DCDC
- PSWITCH and PMIC_ON_REQ are both high to enable DCDC

The sequence is:

1. Power snvs1p0 and clk32k
2. PMIC_ON_REQ = 1
3. DCDC_IN powered up
4. Delay 1ms reset time
5. PSWITCH = 1
6. DCDC is now enabled

OR:

1. Power snvs_1p0 and clk32k
2. DCDC_IN powered up
3. Delay 1ms reset time
4. PSWITCH = 1
5. PMIC_ON_REQ = 1
6. DCDC is now enabled

Only if when PSWITCH and PMIC_ON_REQ are both high, DCDC can be enabled, PMIC_ON_REQ can be before or after DCDC_IN ， just to assure delay about 1ms reset time to enable DCDC after DCDC_IN power up,

NOTE

For safety purpose, over-voltage detector needs to be enabled, i.e. PWD_HIGH_VOLT_DET=1'b0.

18.4.2 Target Voltage Adjustment

DISABLE_STEP=0x0

set TRG

NOTE

If setting disable_step=0, the overshoot will be low when changing the target, but the settling time is longer. If setting disable_step=1, the settling time is shorter, but the overshoot might be higher. The following settings can speed up the settling time: pwd_cmp_offset=0x0; loopctrl_en_rcscale=0x4.

18.4.3 Continuous Conduction Mode

pwd_zcd=0x1

pwd_cmp_offset=0x0

loopctrl_en_rcscale=0x3

18.4.4 Discontinuous Conduction Mode

pwd_zcd=0x0

pwd_cmp_offset=0x0

loopctrl_en_rcscale=0x4

DCM_SET_CTRL=1'b1

NOTE

Because discontinuous conduction mode can increase the efficiency of DCDC in case of low current loading, it is always recommended.

18.4.5 Power Saving Mode

Before entering this mode, set the target value of run mode the same as power saving mode, because DCDC will switch back to run mode if it detects the current loading is larger than about 50 mA (the threshold can be configured).

PMIC_STBY_REQ=0x1

18.5 Memory Map and register definition

This section includes the DCDC module memory map and detailed descriptions of all registers.

18.5.1 DCDC register descriptions

18.5.1.1 DCDC memory map

DCDC base address: 4008_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	DCDC Register 0 (REG0)	32	RW	1403_0111h
4h	DCDC Register 1 (REG1)	32	RW	111B_A29Ch
8h	DCDC Register 2 (REG2)	32	RW	0000_0002h
Ch	DCDC Register 3 (REG3)	32	RW	0000_010Eh

18.5.1.2 DCDC Register 0 (REG0)

18.5.1.2.1 Offset

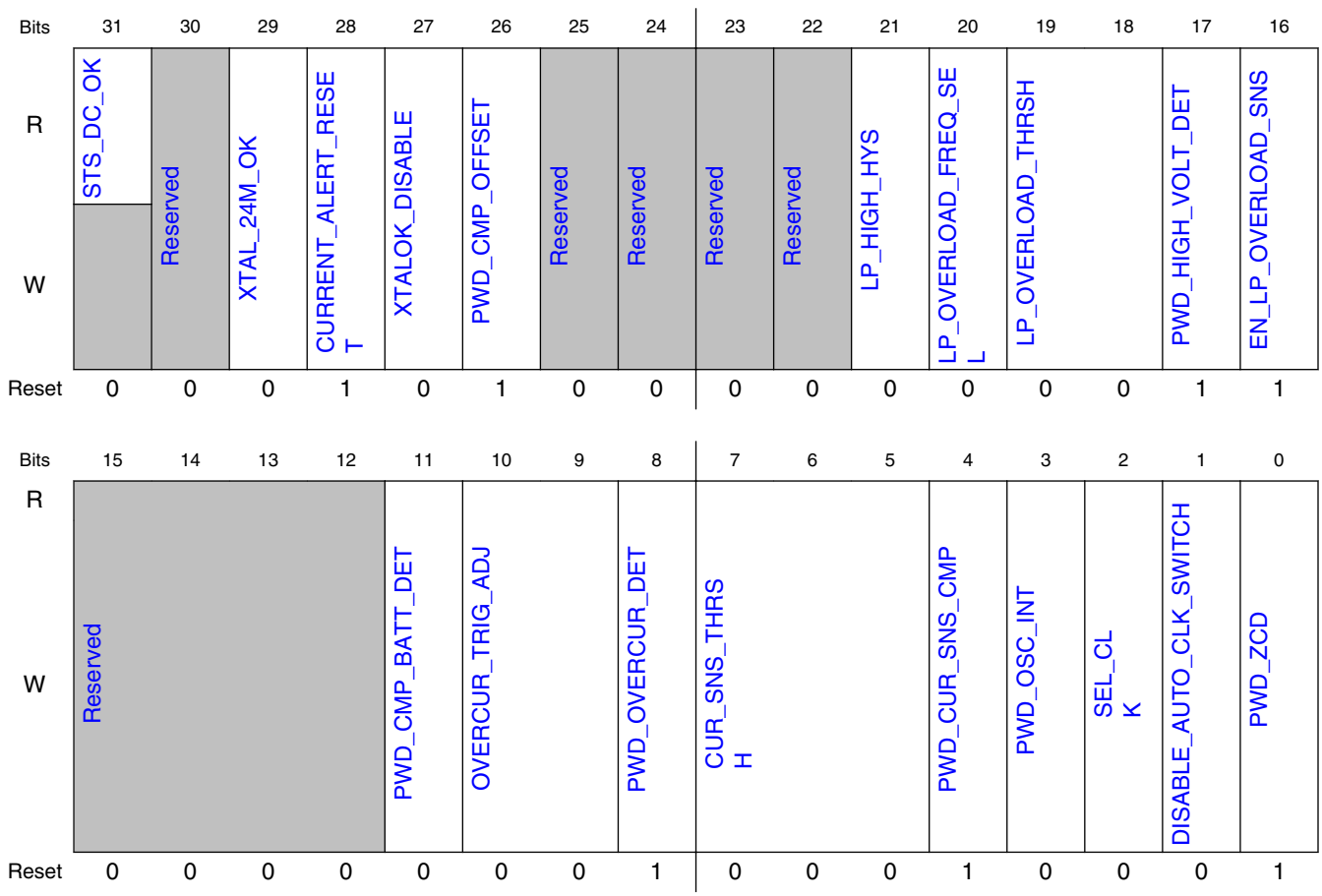
Register	Offset
REG0	0h

18.5.1.2.2 Function

DCDC register 0

This register controls various high-level functions of the DCDC

18.5.1.2.3 Diagram



18.5.1.2.4 Fields

Field	Function
31 STS_DC_OK	DCDC Output OK This is the status bit indicates if the DCDC output voltage is stable. The lock time depends on the loading and the DCDC mode. 0b - DCDC is settling 1b - DCDC already settled
30 —	Reserved
29	24M XTAL OK

Table continues on the next page...

Field	Function
XTAL_24M_OK	This bit determines if the DCDC uses the internal ring oscillator or the xtal 24M. 0b - DCDC uses internal ring OSC 1b - DCDC uses xtal 24M
28 CURRENT_ALERT_RESET	Reset Current Alert Signal This bit is the reset signal of current detector, if current loading is larger than the threshold of the current detector which is set by CUR_SNS_THRSH, the current detector will give out an alert signal, CURRENT_ALERT_RESET can reset the alert signal. 0b - Current Alert Signal not reset 1b - Current Alert Signal reset
27 XTALOK_DISABLE	Disable xtalok detection circuit This is a debug bit which should not be changed in real case. This bit disables and bypasses the clock 24 MHz reference clock detector. 0b - Enable xtalok detection circuit 1b - Disable xtalok detection circuit and always outputs OK signal "1"
26 PWD_CMP_OFFSET	Power down output range comparator Output range comparator monitors the output voltage of DCDC. When the DCDC output voltage is out of range, this comparator will speed up DCDC control loop in an attempt to bring the DCDC output voltage back in range. PWD_CMP_OFFSET must be set to 0 (output range comparator on) if LOOPCTRL_EN_RCSCALE is to be used to speed up the transient response. 0b - Output range comparator powered up 1b - Output range comparator powered down
25 —	Reserved
24 —	Reserved
23 —	Reserved
22 —	Reserved
21 LP_HIGH_HYS	Low Power High Hysteric Value Adjust hysteretic value in low power from 12.5mV to 25mV 0b - Adjust hysteretic value in low power to 12.5mV 1b - Adjust hysteretic value in low power to 25mV
20 LP_OVERLOAD_FREQ_SEL	Low Power Overload Frequency Select The period of counting the charging times in power save mode 0b - eight 32k cycle 1b - sixteen 32k cycle
19-18 LP_OVERLOAD_THRSH	Low Power Overload Threshold the threshold of the counting number of charging times during the period that lp_overload_freq_sel sets in power save mode. 00b - 32 01b - 64 10b - 16

Table continues on the next page...

Field	Function
	11b - 8
17 PWD_HIGH_VOLT_DET	Power Down High Voltage Detection Power down overvoltage detection comparator 0b - Overvoltage detection comparator is enabled 1b - Overvoltage detection comparator is disabled
16 EN_LP_OVERLOAD_SNS	Low Power Overload Sense Enable Enable the overload detection in power save mode, if current is larger than the overloading threshold (typical value is 50 mA), DCDC will switch to the run mode automatically 0b - Overload Detection in power save mode disabled 1b - Overload Detection in power save mode enabled
15-12 —	Reserved
11 PWD_CMP_BATT_DET	Power Down Battery Detection Comparator Set to "1" to power down the low voltage detection comparator 0b - Low voltage detection comparator is enabled 1b - Low voltage detection comparator is disabled
10-9 OVERCUR_TRIG_ADJ	Overcurrent Trigger Adjust The threshold of over current detection in run mode and power save mode: 00b - In Run Mode, 1 A. In Power Save Mode, 0.25 A 01b - In Run Mode, 2 A. In Power Save Mode, 0.25 A 10b - In Run Mode, 1 A. In Power Save Mode, 0.2 A 11b - In Run Mode, 2 A. In Power Save Mode, 0.2 A
8 PWD_OVERCUR_DET	Power down overcurrent detection comparator 0b - Overcurrent detection comparator is enabled 1b - Overcurrent detection comparator is disabled
7-5 CUR_SNS_THRSH	Current Sense (detector) Threshold Set the threshold of current detector, if the peak current of the inductor exceeds the threshold, the current detector will assert. 000b - 150 mA 001b - 250 mA 010b - 350 mA 011b - 450 mA 100b - 550 mA 101b - 650 mA
4 PWD_CUR_SNS_CMP	Power down signal of the current detector. Power down the control signal of the current detector. 0b - Current Detector powered up 1b - Current Detector powered down
3 PWD_OSC_INT	Power down internal osc Only set this bit when 24 MHz crystal osc is available 0b - Internal oscillator powered up 1b - Internal oscillator powered down
2 SEL_CLK	Select Clock Select 24 MHz Crystal clock for DCDC, **when DISABLE_AUTO_CLK_SWITCH is set.**

Table continues on the next page...

Field	Function
	If DISABLE_AUTO_CLK_SWITCH is set to 0 and 24M xtal is OK, the clock source will switch from internal ring OSC to 24M xtal automatically. If DISABLE_AUTO_CLK_SWITCH is set to 1, SEL_CLK will determine which clock source the DCDC uses: if SEL_CLK=0, DCDC will use internal ring OSC, if SEL_CLK=1, DCDC will use 24M xtal. 0b - DCDC uses internal ring oscillator 1b - DCDC uses 24M xtal
1 DISABLE_AUTO_CLK_SWITCH	Disable Auto Clock Switch Disable automatic clock switch from internal osc to xtal clock. 0b - If DISABLE_AUTO_CLK_SWITCH is set to 0 and 24M xtal is OK, the clock source will switch from internal ring OSC to 24M xtal automatically 1b - If DISABLE_AUTO_CLK_SWITCH is set to 1, SEL_CLK will determine which clock source the DCDC uses
0 PWD_ZCD	Power Down Zero Cross Detection Power down the zero cross detection function for discontinuous conductor mode. 0b - Zero cross detetion function powered up 1b - Zero cross detetion function powered down

18.5.1.3 DCDC Register 1 (REG1)

18.5.1.3.1 Offset

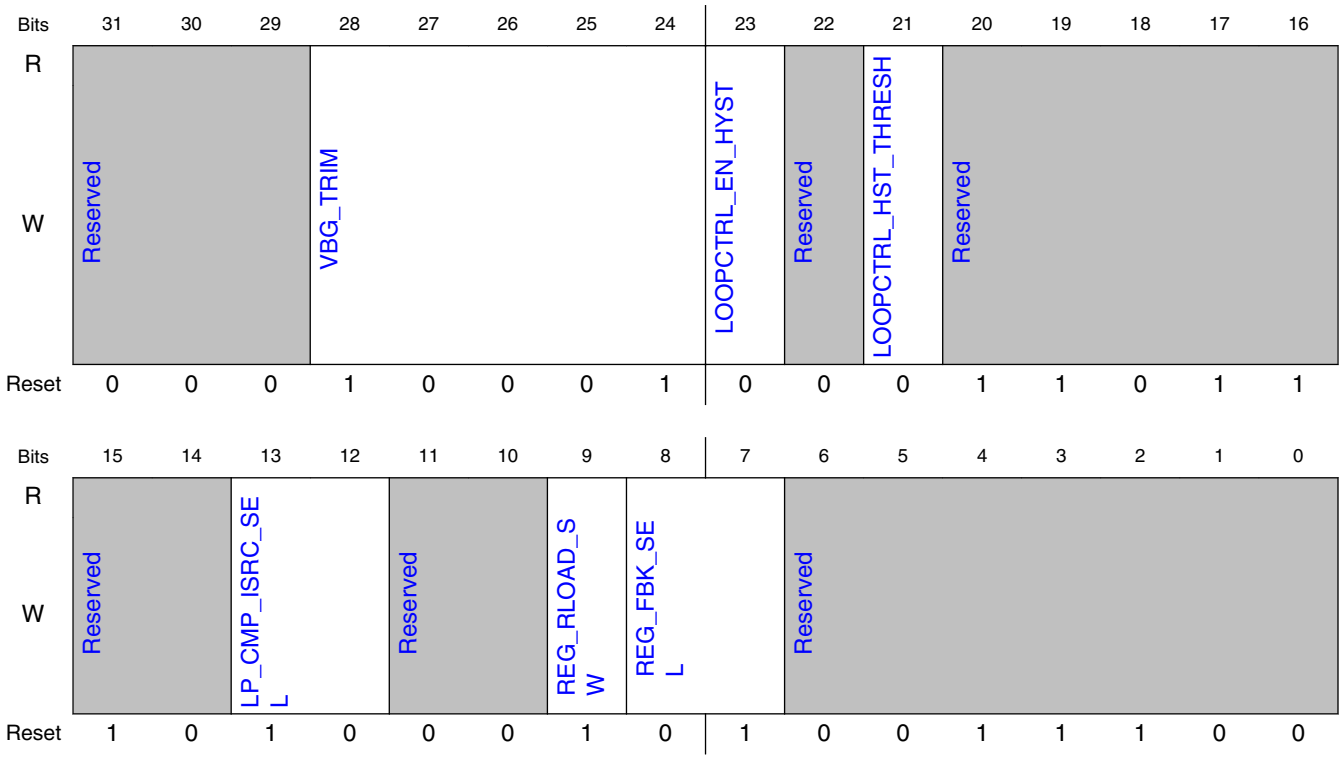
Register	Offset
REG1	4h

18.5.1.3.2 Function

DCDC register 1

This register controls various high-level functions of the DCDC

18.5.1.3.3 Diagram



18.5.1.3.4 Fields

Field	Function
31-29 —	Reserved
28-24 VBG_TRIM	Trim Bandgap Voltage Each bit encoding is 3mV step starting from 0.552V • 0x00 = 0.552V • 0x01 = 0.555V • ... • 0x11 = 0.603 V (default value) • ... • 0x1F 0.645 V
23 LOOPCTRL_EN_HYST	Enable Hysteresis Enable hysteresis in switching converter common mode analog comparators. This feature will improve transient supply ripple and efficiency 0b - Disable hysteresis in switching converter common mode analog comparators 1b - Enable hysteresis in switching converter common mode analog comparators
22 —	Reserved
21	Increase Threshold Detection Increase the hysteresis threshold for the common mode analog comparator.

Table continues on the next page...

Field	Function
LOOPCTRL_HST_THRESH	0b - Lower hysteresis threshold (about 2.5mV in typical, but this value can vary with PVT corners 1b - Higher hysteresis threshold (about 5mV in typical)
20-14 —	Reserved
13-12 LP_CMP_ISRC_SEL	Low Power Comparator Current Bias Set the current bias of low power comparator 00b - 50 nA 01b - 100 nA 10b - 200 nA 11b - 400 nA
11-10 —	Reserved
9 REG_RLOAD_SW	This controls the load resistor of the internal regulator of DCDC. The load resistor is connected by default to 1. To disconnect the load resistor, set this bit to 0. The internal regulator is only powered on in RUN mode. In SUSPEND mode, the whole internal regulator is powered down. If REG_RLOAD_SW is set to 1, the DCDC connects a resistor loading on the output of internal regulator when the regulator starts up. This is good for the stability of the regulator startup. Connecting this load resistor increases the current of DCDC_IN by 0.5mA. After the STS_DC_OK asserts, this control bit can be set to 0 to disconnect the resistor. This saves the current caused by the resistor loading. 0b - Load resistor disconnected 1b - Load resistor connected
8-7 REG_FBK_SEL	Select the feedback point of the internal regulator 00b - The regulator outputs 1.0V with 1.2V reference voltage 01b - The regulator outputs 1.1V with 1.2V reference voltage 10b - The regulator outputs 1.0V with 1.3V reference voltage 11b - The regulator outputs 1.1V with 1.3V reference voltage
6-0 —	Reserved

18.5.1.4 DCDC Register 2 (REG2)

18.5.1.4.1 Offset

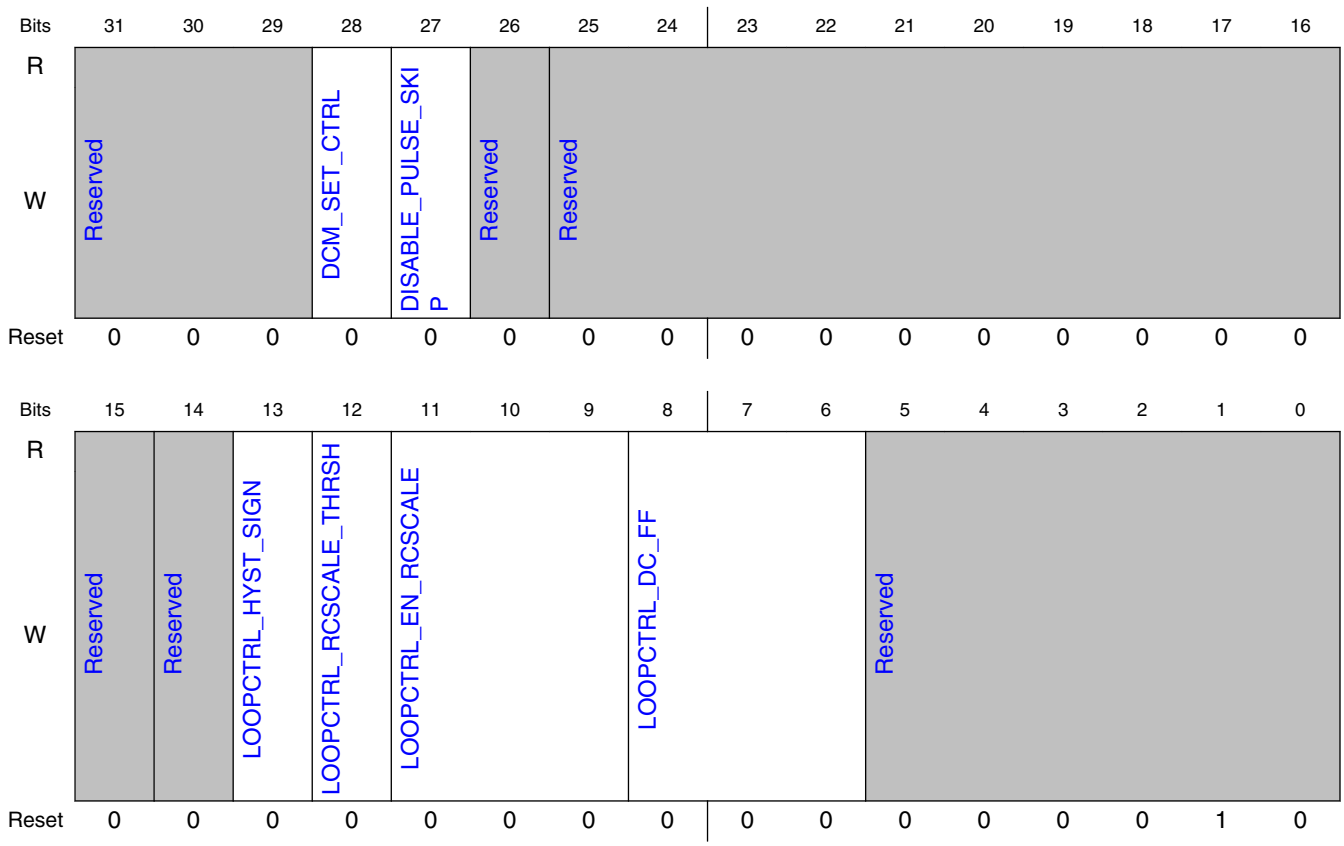
Register	Offset
REG2	8h

18.5.1.4.2 Function

DCDC register 2

This register controls various high-level functions of the DCDC

18.5.1.4.3 Diagram



18.5.1.4.4 Fields

Field	Function
31-29 —	Reserved
28 DCM_SET_CTRL	DCM Set Control Set high to improve the transition from heavy load to light load When DCDC is working in DCM and DCM_SET_CTRL= 1, the DCDC will switch to CCM to discharge the output when the overshoot on the output is higher than the out-of-range threshold, by which the overshoot on the transition can be improved. If DCM_SET_CTRL=0, the DCDC doesn't support this function. DCM_SET_CTRL=1 is recommended when supporting DCM mode.
27 DISABLE_PULSE_SKIP	Disable Pulse Skip Set to "0" : stop charging if the duty cycle is lower than what set by NEGLIMIT_IN. 0b - DCDC will be idle to save current dissipation when the duty cycle get to the low limit which is set by NEGLIMIT_IN.

Table continues on the next page...

Field	Function
	1b - DCDC will keep working with the low limited duty cycle NEGLIMIT_IN.
26 —	Reserved
25-16 —	Reserved
15 —	Reserved
14 —	Reserved
13 LOOPCTRL_HYST_SIGN	Invert the sign of the hysteresis in DC-DC analog comparators. 0b - Do not invert sign of the hysteresis 1b - Invert sign of the hysteresis
12 LOOPCTRL_RCSCALE_THRSH	Increase the threshold detection for RC scale circuit. 0b - Do not increase the threshold detection for RC scale circuit. 1b - Increase the threshold detection for RC scale circuit.
11-9 LOOPCTRL_EN_RCSCALE	Enable RC Scale This enables the analog circuit of the DCDC to respond faster under transient load conditions. A larger number for this bit field means a faster transient response. In order for a faster transient response to be implemented, the offset comparators must be turned on (PWD_CMP_OFFSET = 0). This means about 80uA additional current consumption is needed. It is recommended set LOOPCTRL_EN_RCSCALE > 0 if big steps of current loading is required in a real application. LOOPCTRL_EN_RCSCALE = 0x3 is recommended for CCM and LOOPCTRL_EN_RCSCALE = 0x4 for DCM. However, the DCDC can be more stable with LOOPCTRL_EN_RCSCALE = 0, and the offset comparators are also powered down by default. This results in lower current consumption. If big steps of current loading is not required, it is recommended that the default value of 0 is used. The value of this bit field doesn't match an RC constant directly.
8-6 LOOPCTRL_DC_FF	Two's complement feed forward step in duty cycle in the switching DC-DC converter. Each time this field makes a transition from 0x0, the loop filter of the DC-DC converter is stepped once by a value proportional to the change. This can be used to force a certain control loop behavior, such as improving response under known heavy load transients.
5-0 —	Reserved

18.5.1.5 DCDC Register 3 (REG3)

18.5.1.5.1 Offset

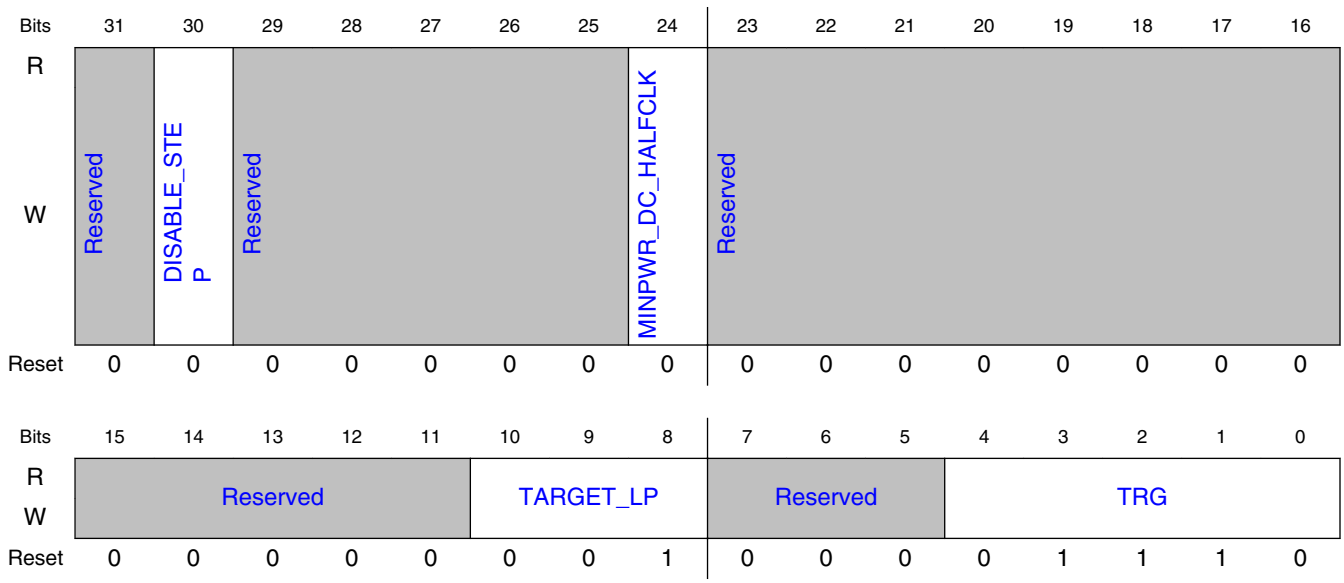
Register	Offset
REG3	Ch

18.5.1.5.2 Function

DCDC register 3

This register controls various high-level functions of the DCDC

18.5.1.5.3 Diagram



18.5.1.5.4 Fields

Field	Function
31 —	Reserved
30 DISABLE_STEP	Disable Step Disable stepping for the output VDD_SOC of DCDC 0b - Enable stepping for the output of VDD_SOC of DCDC 1b - Disable stepping for the output of VDD_SOC of DCDC
29-25 —	Reserved
24 MINPWR_DC_HALFCLK	Set DCDC clock to half frequency for continuous mode 0b - DCDC clock remains at full frequency for continuous mode 1b - DCDC clock set to half frequency for continuous mode
23-11 —	Reserved
10-8	Low Power Target Value

Table continues on the next page...

Field	Function
TARGET_LP	Target value of standby (low power) mode 000b - 0.9 V 001b - 0.925 V 010b - 0.95 V 011b - 0.975 V 100b - 1.0 V
7-5 —	Reserved
4-0 TRG	Target value of VDD_SOC Target value of VDD_SOC, 25 mV each step • 0x0: 0.8V • 0xE: 1.15V • 0x1F:1.575V

Chapter 19
Temperature Monitor (TEMPMON)

19.1 Chip-specific TEMPMON information

Table 19-1. Reference links to related information

Topic	Related module(s)	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
Interrupts, DMA Events and XBAR Assignments	-	Interrupts, DMA Events and XBAR Assignments

19.2 Overview

The temperature sensor module implements a temperature sensor/conversion function based on a temperature-dependent voltage to time conversion.

The module features alarm functions that can raise independent interrupt signals if the temperature is above two high-temperature thresholds and below a low temperature threshold. These temperature thresholds are programmable and designated as low, high and panic temperature. The panic threshold is a special programmable threshold in that if the temperature increases above this value and the temperature-panic-reset interrupt is enabled in the System Reset Controller, the hardware will assume that software no longer has control over the thermal situation and will initiate a reset of the chip.

In order to avoid false panic temperature initiated resets, the panic alarm will not fire until the temperature panic condition has been met for four consecutive conversion cycles. A self-repeating mode can also be programmed which executes a temperature sensing operation based on a programmed delay.

Since the high and low temperature thresholds are programmable, they form a sliding temperature bracket that can be tailored to the application's needs. For example, at start-up software can set the low temperature threshold to the minimum temperature code and the high temperature threshold to a maximum operating temperature for the system.

The system can then use this module to monitor the on-die temperature and take appropriate actions such as throttling back the core frequency when a the high temperature interrupt is set. After the high temperature interrupt is set then the system can program the low temperature threshold to a desired cool down temperature. The system would then switch to monitoring the low temperature alarm and wait for its interrupt to be set. With this scheme, after the low temperature interrupt is set then software could be assured that the temperature has cooled down to a safe level and the process could be repeated.

The high-level implementation of the temperature sensor is shown in the figure below.

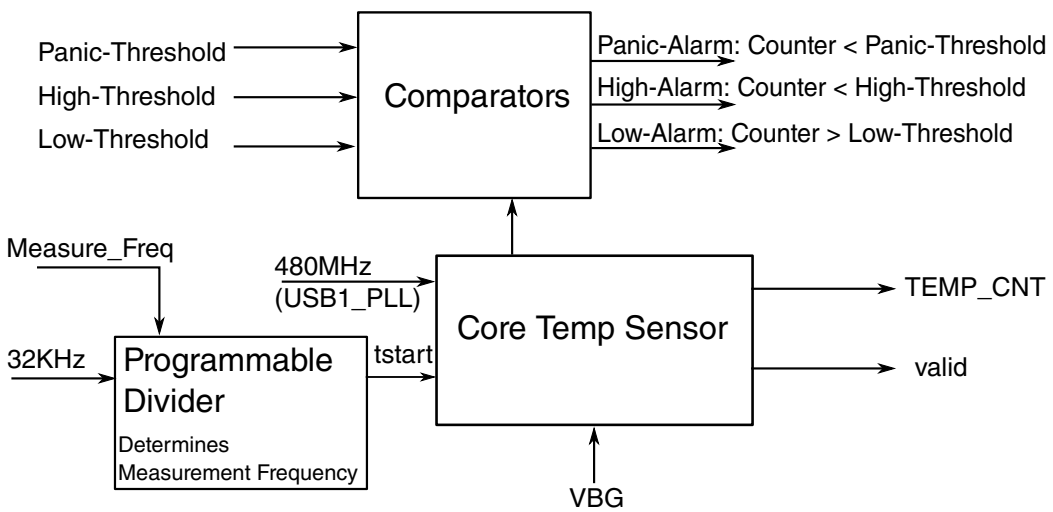


Figure 19-1. High Level Temp Sensor System Diagram

As shown in the figure above, the temperature sensor uses and assumes that the bandgap reference, 480MHz PLL and 32KHz RTC modules are properly programmed and fully settled for correct operation.

19.3 Software Usage Guidelines

During normal system operation software can use the temperature sensor counter output (TEMP_CNT) in conjunction with the fused temperature calibration data to determine the on-die operational temperature or to set an over-temperature interrupt alarm to within a couple of °C.

Based on calibration, two sets of temperature and counter values will be available via fuses on the device. These data points will correspond to the points (N_1, T_1) and (N_2, T_2) in the curve below.

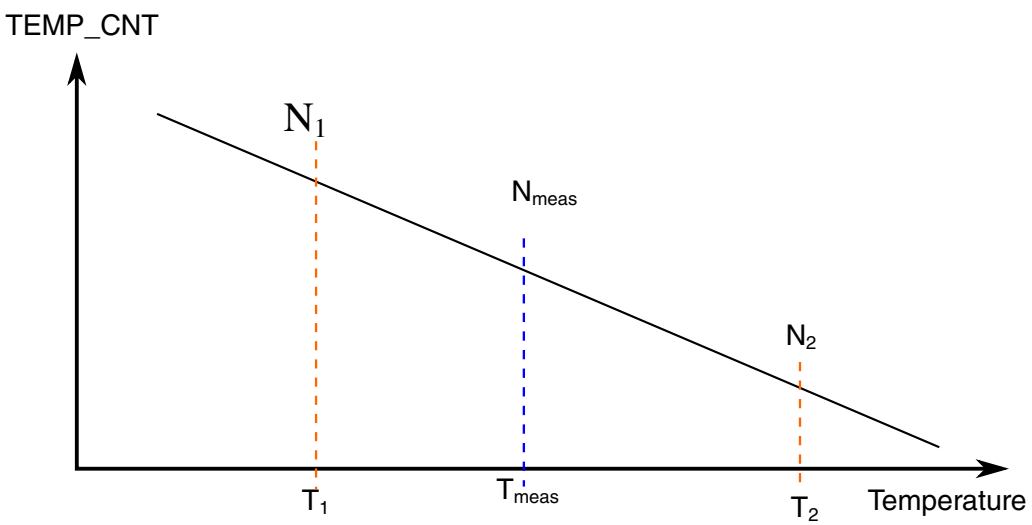


Figure 19-2. Temperature Measurement Cycle

After a temperature measurement cycle, software should use the calibration points in conjunction with the temperature code value in the TEMPMON_TEMPSENSE0[TEMP_CNT] bitfield to calculate the temperature for the device using the following equation:

$T_{meas} = T_2 - (N_{meas} - N_2) * ((T_2 – T_1) / (N_1 – N_2))$

Likewise, to determine the alarm counter value to be written in the TEMPMON_TEMPSENSE0 register for a temperature based interrupt, the above equation can be solved for the N_{meas} value that should be used based on the desired temperature trigger.

The temperature calibration point fuse values are available in the OCOTP_ANA1 register. The temperature calibration values are fused individually for each part in the product testing process. The fields of this register are described in the following table.

Table 19-2. OCOTP_ANA1 Temperature Sensor Calibration Data

Bit Range	Bit Mask	Name	Description
[31:20]	FFF0_0000h	ROOM_COUNT	Value of TEMPMON_TEMPSENSE0[TEMP_VALUE] after a measurement cycle at room temperature (25.0 °C).
[19:8]	000F_FF00h	HOT_COUNT	Value of TEMPMON_TEMPSENSE0[TEMP_VALUE] after a measurement cycle at the hot temperature, i.e. HOT_TEMP.
[7:0]	0000_00FFh	HOT_TEMP	The hot temperature test point. Each LSB equals 1 °C.

The points on the calibration curve are as follows.

- (N_1, T_1) = (ROOM_COUNT, 25.0)
- (N_2, T_2) = (HOT_COUNT, HOT_TEMP)
- (N_{meas}, T_{meas}) = (TEMP_CNT, T_{meas})

Substituting the fields from OCOTP_ANA1 into the earlier equation results in the following:

T_{meas} = HOT_TEMP - (N_{meas} - HOT_COUNT) * ((HOT_TEMP - 25.0) / (ROOM_COUNT – HOT_COUNT))

19.4 TEMPMON Memory Map/Register Definition

TEMPMON memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400D_8180	Tempsensor Control Register 0 (TEMPMON_TEMPSENSE0)	32	R/W	0000_0001h	19.4.1/1220
400D_8184	Tempsensor Control Register 0 (TEMPMON_TEMPSENSE0_SET)	32	R/W	0000_0001h	19.4.1/1220
400D_8188	Tempsensor Control Register 0 (TEMPMON_TEMPSENSE0_CLR)	32	R/W	0000_0001h	19.4.1/1220
400D_818C	Tempsensor Control Register 0 (TEMPMON_TEMPSENSE0_TOG)	32	R/W	0000_0001h	19.4.1/1220
400D_8190	Tempsensor Control Register 1 (TEMPMON_TEMPSENSE1)	32	R/W	0000_0001h	19.4.2/1222
400D_8194	Tempsensor Control Register 1 (TEMPMON_TEMPSENSE1_SET)	32	R/W	0000_0001h	19.4.2/1222
400D_8198	Tempsensor Control Register 1 (TEMPMON_TEMPSENSE1_CLR)	32	R/W	0000_0001h	19.4.2/1222
400D_819C	Tempsensor Control Register 1 (TEMPMON_TEMPSENSE1_TOG)	32	R/W	0000_0001h	19.4.2/1222
400D_8290	Tempsensor Control Register 2 (TEMPMON_TEMPSENSE2)	32	R/W	0000_0000h	19.4.3/1222

Table continues on the next page...

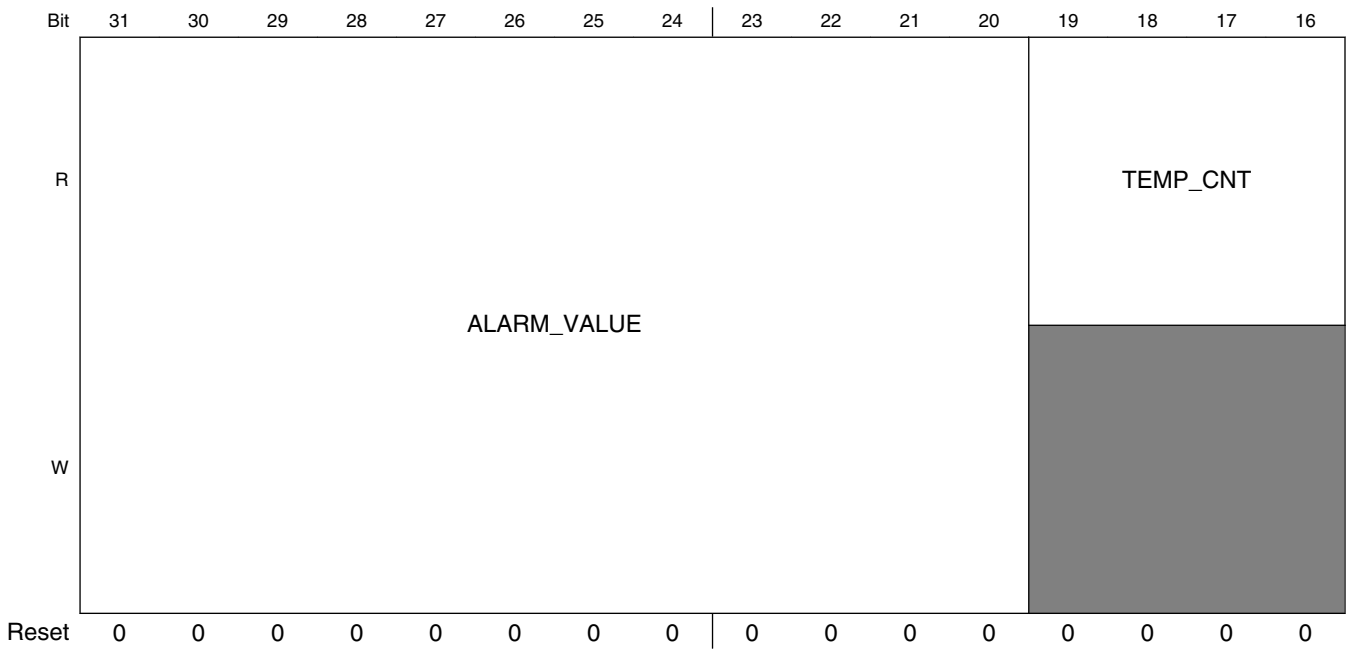
TEMPMON memory map (continued)

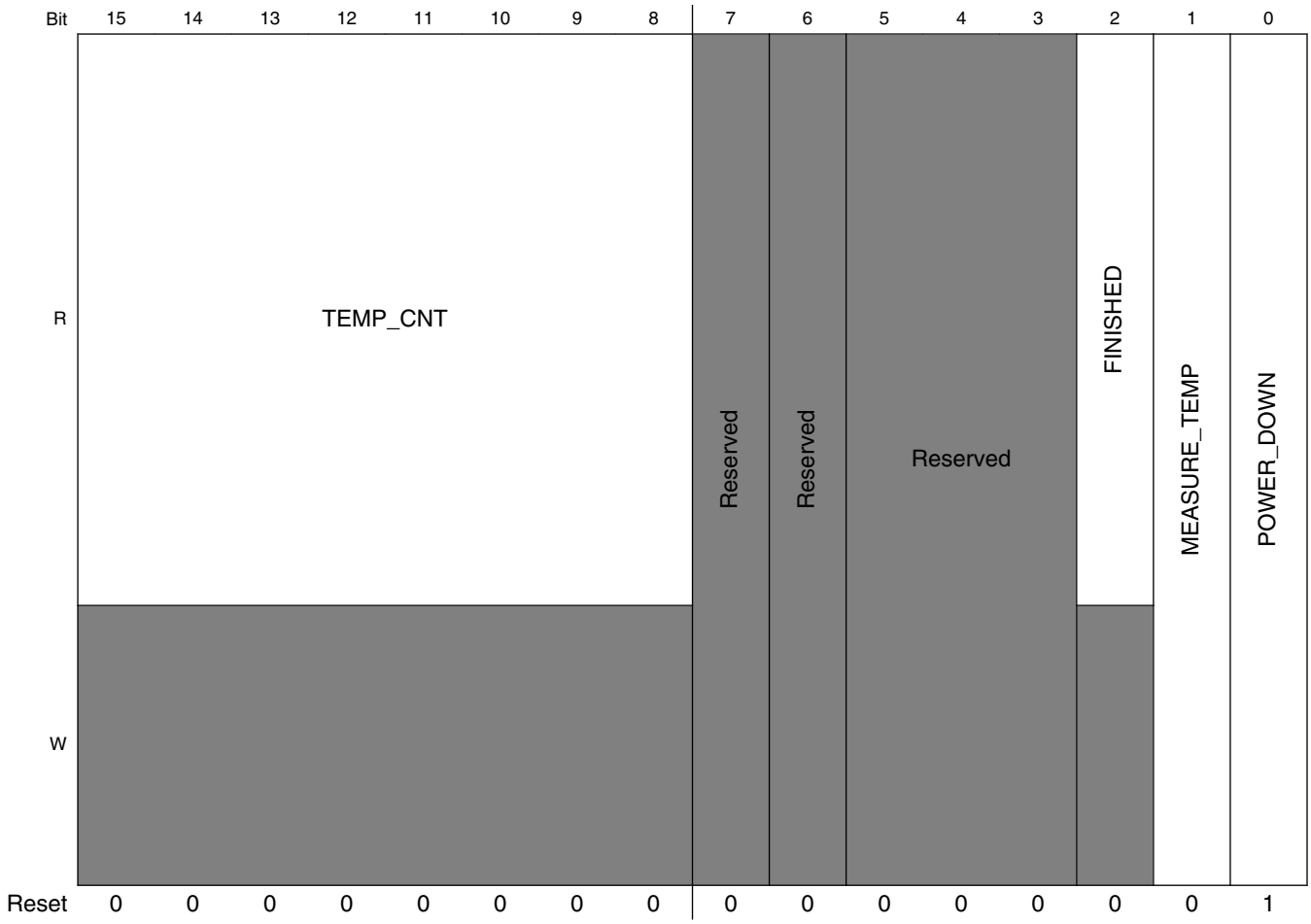
Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400D_8294	Tempsensor Control Register 2 (TEMPMON_TEMPSENSE2_SET)	32	R/W	0000_0000h	19.4.3/1222
400D_8298	Tempsensor Control Register 2 (TEMPMON_TEMPSENSE2_CLR)	32	R/W	0000_0000h	19.4.3/1222
400D_829C	Tempsensor Control Register 2 (TEMPMON_TEMPSENSE2_TOG)	32	R/W	0000_0000h	19.4.3/1222

19.4.1 Tempsensor Control Register 0 (TEMPMON_TEMPSENSE0*n*)

This register defines the basic controls for the temperature sensor minus the frequency of automatic sampling which is defined in the tempsensor.

Address: 400D_8000h base + 180h offset + (4d × i), where i=0d to 3d





TEMPMON_TEMPSENSE0*n* field descriptions

Field	Description
31–20 ALARM_VALUE	This bit field contains the temperature count (raw sensor output) that will generate a high alarm when TEMP_CNT is smaller than this field.
19–8 TEMP_CNT	This bit field contains the last measured temperature count.
7 -	This field is reserved. Reserved.
6 -	This field is reserved. Reserved.
5–3 -	This field is reserved. Reserved
2 FINISHED	Indicates that the latest temp is valid. This bit should be cleared by the sensor after the start of each measurement. 0 **INVALID** — Last measurement is not ready yet. 1 **VALID** — Last measurement is valid.
1 MEASURE_ TEMP	Starts the measurement process. If the measurement frequency is zero in the TEMPSENSE1 register, this results in a single conversion.

Table continues on the next page...

TEMPMON_TEMPSENSE0*n* field descriptions (continued)

Field	Description
	0 **STOP** — Do not start the measurement process. 1 **START** — Start the measurement process.
0 POWER_DOWN	This bit powers down the temperature sensor. 0 **POWER_UP** — Enable power to the temperature sensor. 1 **POWER_DOWN** — Power down the temperature sensor.

19.4.2 Tempsensor Control Register 1 (TEMPMON_TEMPSENSE1*n*)

This register defines the automatic repeat time of the temperature sensor.

Address: 400D_8000h base + 190h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																MEASURE_FREQ															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

TEMPMON_TEMPSENSE1*n* field descriptions

Field	Description
31–16 -	This field is reserved. Reserved.
MEASURE_ FREQ	This bits determines how many RTC clocks to wait before automatically repeating a temperature measurement. The pause time before remeasuring is the field value multiplied by the RTC period. 0x0000 Defines a single measurement with no repeat. 0x0001 Updates the temperature value at a RTC clock rate. 0x0002 Updates the temperature value at a RTC/2 clock rate. . . . — 0xFFFF Determines a two second sample period with a 32.768KHz RTC clock. Exact timings depend on the accuracy of the RTC clock.

19.4.3 Tempsensor Control Register 2 (TEMPMON_TEMPSENSE2*n*)

This register defines the automatic repeat time of the temperature sensor.

Address: 400D_8000h base + 290h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved				PANIC_ALARM_VALUE												Reserved				LOW_ALARM_VALUE											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TEMPMON_TEMPSENSE2*n* field descriptions

Field	Description
31–28 -	This field is reserved. Reserved
27–16 PANIC_ALARM_VALUE	This bit field contains the temperature count that will generate a panic interrupt when TEMP_CNT is smaller than this field.
15–12 -	This field is reserved. Reserved.
LOW_ALARM_VALUE	This bit field contains the temperature count that will generate a low alarm interrupt when the field is exceeded by TEMP_CNT.

Chapter 20
Secure Non-Volatile Storage (SNVS)

20.1 Chip-specific SNVS information

Table 20-1. Reference links to related information

Topic	Related module(s)	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
Interrupts, DMA Events and XBAR Assignments	-	Interrupts, DMA Events and XBAR Assignments

NOTE

Tamper Detection feature and ZMK hardware programming not applicable on this device.

20.2 Overview

The Secure Non-Volatile Storage (SNVS) is a companion module to the DCP module.

The SNVS non-security functionality is described in this document, but the SNVS security functionality is described only in the Security Reference Manual.

20.2.1 Block diagram

The following figure illustrates the structure of SNVS.

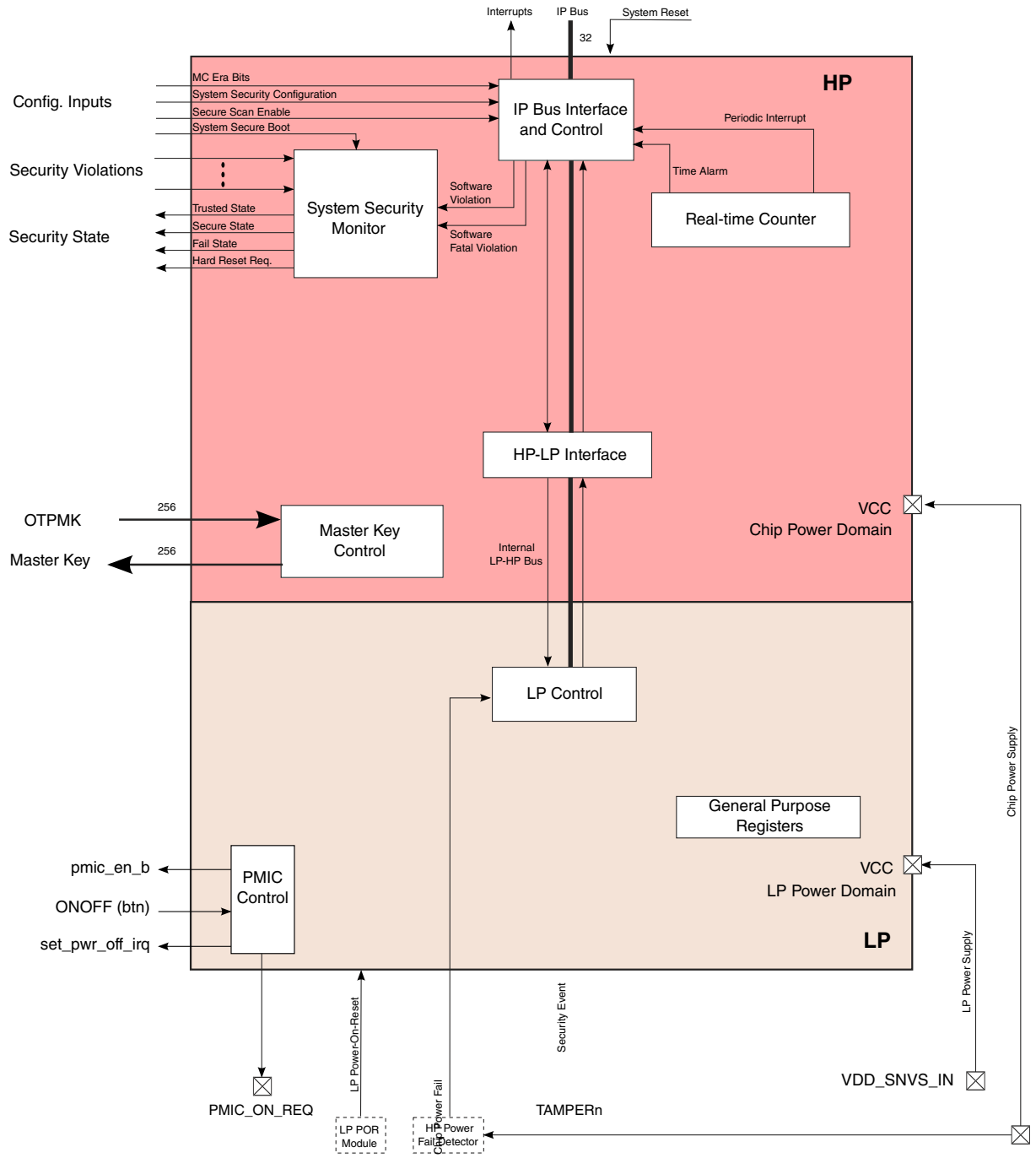


Figure 20-1. SNVS Block Diagram

20.2.2 Features

The following table summarizes the features of SNVS:

Table 20-2. SNVS feature list

Feature	Description	Links for Further Information
Real time counter (RTC)	• The RTC is driven by a dedicated clock, which is off when the system power is down. • Programmable time alarm interrupt • Periodic interrupt can be generated with software-selected frequency.	SNVS_HP Real Time Counter
General-purpose register	• The general-purpose register is available to software to store 256 bits of data. • The general-purpose register is zeroized when a security violation is detected. • If the SNVS_LP power input is connected to an uninterrupted power supply (see SNVS power domains), the general-purpose register value is retained even if the main chip is powered down.	Using the General-Purpose Register
Register access restrictions	• Some registers/values can be written only once per boot cycle.	privileged and non-privileged registers
Wakeup from power off	• Input signal from off chip requests SNVS_LP to power on the main SoC (Assuming that the SNVS_LP power input is connected to an uninterrupted power supply (see SNVS power domains). • Hardware debounces the input signal using software-specified signal bounce characteristics	LP Wake-Up Interrupt Enable

20.3 SNVS functional description

SNVS implements several non-security features that involve software interaction:

- reading or writing the Realtime Counter (RTC) (This is a non-privileged operation.) - software can also instruct SNVS to load the current SRTC value into the RTC
- reading or writing the General Purpose Register (GPR) (Note that there may be a significant delay when reading or writing registers in the LP section if the LP clock is different from the HP clock.)

The following sections describe in more detail the operation of SNVS.

20.3.1 SNVS power domains

In some versions of SNVS (including this version), the LP (Low Power) section is implemented in an independent power domain from the HP (High Power) section, and most other logic on the chip. Throughout the SNVS documentation whenever mention is made of "always-on" logic, this assumes a version of SNVS that implements an independent power domain for the LP section, and that the power for this section is supplied by an uninterrupted power supply. The purpose for the independent power domain is so that data can be retained and certain logic can remain functional even when the main chip logic is powered down. But this is possible only if the LP domain remains

powered via an uninterrupted power supply when the main chip power domain is powered off. Usually this uninterrupted power supply would be a battery, with possibly some power management logic to power the LP section from main power (and perhaps recharge the battery) when main power is on, and switch to battery power when the main power is off. In versions of SNVS with an independent LP power domain the LP section can be electrically isolated from the rest of the chip logic to ensure that its logic does not get corrupted when the main chip is powered down. If the battery runs down or is removed, an LP POR will occur when the LP section next powers up. Note that some OEMs may choose to connect LP power to HP/main chip power and dispense with a battery. In that case the SNVS will operate the same as an SNVS without an independent LP power domain. No state will be retained in the LP section when the chip is powered down, and an LP POR will occur whenever there is an HP POR.

20.3.2 Digital Low-Voltage Detector (LVD)

SNVS_LP incorporates a mechanism to detect interruptions of the SNVS_LP power supply that might cause the LP control, status, and secure counter values to change. The mechanism works as follows:

1. The LVD register (LPLVDR) is loaded with the known specific value 4173_6166h as part of the SNVS initialization process.
2. Subsequently, this register's value is continuously compared to the hardwired value 4173_6166h.
3. If the comparison indicates that any bit has changed, a low-voltage violation is asserted.

Digital low-voltage detection is always enabled and cannot be disabled. At LP POR this register is reset to all 0's, so the hardwired comparison fails and a low-voltage violation is reported. Therefore, before programming any feature in the SNVS the low-voltage violation should be cleared. The initialization software should write the proper value (4173_6166h) into LPLVDR (see SNVS_LP Digital Low-Voltage Detector Register (LPLVDR)) and should then clear the low-voltage event record in the LP status register (see SNVS_LP Status Register (LPSR)).

The following figure shows the LVD mechanism.

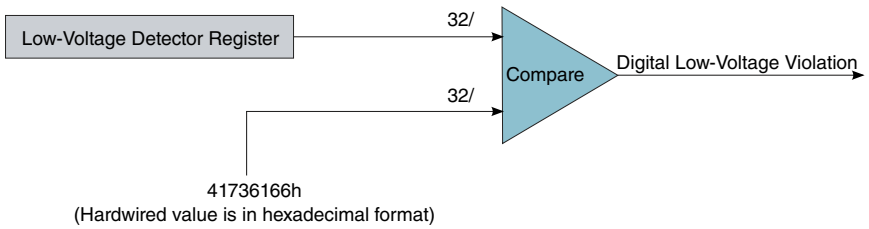


Figure 20-2. Digital low-voltage detector

20.3.3 Runtime Procedures

SNVS implements a number of features that are intended to be accessed by software at runtime (as opposed to accessed at boot time). These features include:

- Real Time Clock (see SNVS_HP Real Time Counter)
- General Purpose Register (see Using the General-Purpose Register)

Procedures for using these features are described in the following sections.

20.3.3.1 Using SNVS Timer Facilities

SNVS incorporates timer facilities that can optionally generate an interrupt at a specified time. As described in the following sections, SNVS_HP incorporates a Real Time Counter that is available for general use, and SNVS_LP incorporates a Secure Real Time Counter intended for security applications.

20.3.3.1.1 SNVS_HP Real Time Counter

SNVS_HP implements a real time counter that can be read or written by any application; it has no privileged software access restrictions. When the chip is powered down the RTC is not active and it is reset at chip POR. The RTC can be used to generate a functional interrupt request either at a specific time, or at a specific frequency, or both. To generate an interrupt request at a specific time HPTA_EN is set to 0, the desired time is written to HPTA_MS and HPTA_LS and then HPTA_EN is set to 1. HPTA_EN, HPTA_MS and HPTA_LS can be written by any software that has access to SNVS registers; there are no privileged access restrictions. The counter can be synchronized to the SNVS_LP SRTC by writing to the HP_TS bit of SNVS_HP Control Register. This is particularly useful if the SNVS_LP is powered from an uninterrupted power source because the RTC can then be set from a chip-internal time source.

20.3.3.1.2 RTC/SRTC control bits setting

All SNVS registers are programmed from the register bus, consequently any software-initiated changes are synchronized with the IP clock. Several registers can also change synchronously with the RTC/SRTC clock after they are programmed. To avoid IP clock and RTC/SRTC clock synchronization issues, the following values can be changed only when the corresponding function is disabled.

Table 20-3. RTC/SRTC synchronized values list

Function	Value/register	Control bit setting
HP section		
HP Real Time Counter	HPRTCMR and HPRTCLR Registers	RTC_EN = 0 : HPRTCMR/HPRTCLR **can** be programmed RTC_EN = 1 : HPRTCMR/HPRTCLR **cannot** be programmed
HP Time Alarm	HPTAMR and HPTALR Registers	HPTA_EN = 0 : HPTAMR/HPTALR **can** be programmed HPTA_EN = 1 : HPTAMR/HPTALR **cannot** be programmed
LP section		
LP Secure Real Time Counter	LPRTCMR and LPRTCLR Registers	SRTC_ENV = 0 : LPRTCMR/LPRTCLR **can** be programmed SRTC_ENV = 1 : LPRTCMR/LPRTCLR **cannot** be programmed
LP Time Alarm	LPTAR Register	LPTA_EN = 0 : LPTAR **can** be programmed LPTA_EN = 1 : LPTAR **cannot** be programmed

Use the following steps to program synchronized values:

1. Check the enable bit value. If set, clear it.

2. Verify that the enable bit is cleared. There are two reasons to verify the enable bit's setting:

- Enable bit clearing does not happen immediately; it takes three IP clock cycles and two RTC/SRTC clock cycles to change the enable bit's value.
- If the enable bit is locked for programming, it cannot be cleared.

3. Program the desired value.

4. Set the enable bit; it takes three IP clock cycles and two RTC/SRTC clock cycles for the bit to set.

NOTE

Incrementing the value programmed into RTC/SRTC registers by two compensates for the two RTC/SRTC clock cycle delay that is required to enable the counter.

20.3.3.1.3 Reading RTC and SRTC values

Software should follow the following procedure to ensure that it has read correct data from the RTC (HPRTCMR and HPRTCLR) and SRTC (LPSRTCMR and LPSRTCLR) registers:

- Read the most-significant half and the least-significant half of the RTC/SRTC and then read both halves again. If the values read are the same both times, the value is correct.
- If the two consecutive pairs of reads yield different results, perform two more reads.

The worst case scenario may require three sessions of two consecutive pairs of reads. There are several reasons that the values may be incorrectly read initially:

- Synchronization issues between the RTC/SRTC clock and the system clock
- Since the counter continues to increment, there may be a carry from the least-significant 32-bits to the most-significant bits in between reading the two halves of the counter

20.3.3.2 Using Other SNVS Registers

The sections below describe how to use the General Purpose Register.Monotonic Counter.The sections below describe how to use the General Purpose Register and the Monotonic Counter.

20.3.3.2.1 Using the General-Purpose Register

SNVS implements a 256-bit general-purpose register allows software to store a small amount of data. To maintain backward compatibility with versions of SNVS that implement only a 32-bit general purpose register, the most-significant word of the general purpose register is aliased to the original legacy address, and to maintain backward compatibility with versions of snvs_module_name that implement a 128-bit general purpose register, the most-significant half of the general purpose register is aliased to the previous legacy address address. The data in the GPR will be retained during system power-down mode as long as the SNVS_LP remains powered by an uninterrupted power source.

20.3.4 Clocks

The SNVS has the following clock sources:

- System peripheral clock input. This clock is used by the SNVS's internal logic, for example, the Security State Machine. This clock can be gated outside of the module when the SNVS indicates that it is not in use.
- HP RTC clock. This clock is used by SNVS_HP real-time counter. This clock does not need to be synchronous with other clocks.

20.3.5 Reset

The table below shows the different resets associated with this module.

Table 20-4. Reset summary

Reset	Source	Characteristics	Internally resets
HP hard	ipg_hard_async_reset_b	active-low, asynchronous	All SNVS_HP and SNVS_LP registers and flops.
LP Power On Reset (POR)	lp_por_b	active-low, asynchronous	All SNVS_LP registers and flops.
LP software reset	software	active-high, synchronous, one cycle	All SNVS_LP registers and flops. The LP software reset can be asserted if not disabled.

20.4 External Signals

Table 20-5. External Signals

Signal	Description	Direction
TAMPERn	External Tamper signals	I
PMIC_ON_REQ	PMIC ON Request signal	I

20.5 Initialization of SNVS

SNVS is implemented in two sections (HP and LP) that both must be initialized by software. If the SNVS_LP is powered by an uninterrupted power source that is separate from main SoC power, then SNVS can operate in either of two modes, depending upon whether the main SoC power is on or off.

- During main SoC power-down SNVS_HP is powered-down, but SNVS_LP is powered from the backup power supply and is electrically isolated from the rest of the chip. In this mode SNVS_LP keeps its registers' values but the LP registers cannot be read or written.
- During main SoC power-up the isolation of SNVS_LP is disabled and both SNVS_HP and SNVS_LP are powered from the main SoC power. Both LP and HP registers can be read and written (locks and privilege modes permitting). Signals between the SNVS_HP and SNVS_LP sections are enabled and all SNVS functions are operational.

Since the HP and LP sections reside in different power domains, the POR for the two sections can occur at different times. If the SNVS_LP section remains powered by an uninterrupted power source when the main SoC power is off, SNVS_LP is initialized rarely, typically once when the device is first powered on and again whenever the battery is replaced. During main SoC power-up the isolation of SNVS_LP is disabled and both SNVS_HP and SNVS_LP are powered from the main SoC power. Signals between the SNVS_HP and SNVS_LP sections are enabled and all SNVS functions are operational. The SNVS_HP section is powered from the main SoC power, so it must be initialized after the device is powered on. If the SNVS_LP section is powered from the main SoC power rather than from an uninterrupted power source, the SNVS_LP section must also be initialized at SoC POR.

- Initializing the LP section
  - The following steps should be completed to properly initialize the SNVS LP section (required only on LP POR, i.e. when the battery is replaced):
    - Software should write the proper initialization value (41736166h) into the LP Digital Low-Voltage Detector Register and clear the low-voltage event record in the LP status register. See Digital Low-Voltage Detector (LVD) for more details.
- Initializing the HP section
  - The following steps should be completed to properly initialize the SNVS HP section (required on HP POR, i.e. SoC POR):
    - Perform normal boot to put the SNVS into a functional state (Non-secure, Trusted, Secure) (see HP Command Register, SSM_ST bitfield).
    - Program SNVS general functions/configurations (see HP Control Register).

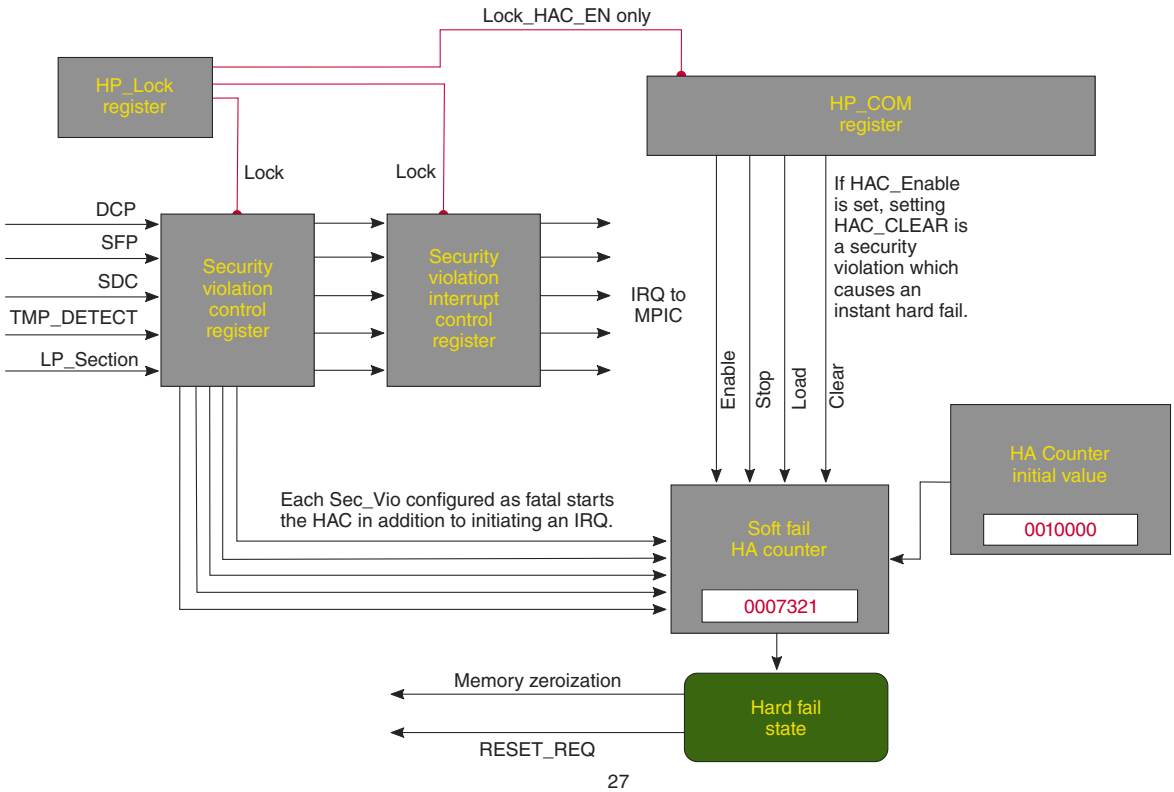


Figure 20-3. Relationship Between the Registers

20.6 Memory Map and register definition

This section contains detailed register descriptions for the SNVS registers. Each description includes a standard register diagram and register table. The register table provides detailed descriptions of the register bit and field functions, in bit order.

SNVS registers consist of two types:

- Privileged read/write accessible
- Non-privileged read/write accessible

Privileged read/write accessible registers can only be accessed for read/write by privileged software. Unauthorized write accesses are ignored, and unauthorized read accesses return zero. Non-privileged software can access privileged access registers when the non-privileged software access enable bit is set in the SNVS_HP Command Register.

- Non-Secure
- Trusted
- Secure

Non-privileged read/write accessible registers are read/write accessible by any software.

The LP register values are set only on LP POR and are unaffected by System (HP) POR. The HP registers are set only on System POR and are unaffected by LP POR.

The following table shows the SNVS main memory map.

NOTE

For more information on security-related bitfields, see the Security Reference Manual.

20.6.1 SNVS register descriptions

20.6.1.1 SNVS memory map

SNVS base address: 400D_4000h

Offset (hex)	Register	Width (In bits)	Access	Reset value (hex)
4	SNVS_HP Command Register (HPCOMR)	32	RW	0000_0000
8	SNVS_HP Control Register (HPCR)	32	RW	0000_0000
14	SNVS_HP Status Register (HPSR)	32	RW	8000_0000
24	SNVS_HP Real Time Counter MSB Register (HPRTCMR)	32	RW	0000_0000
28	SNVS_HP Real Time Counter LSB Register (HPRTCLR)	32	RW	0000_0000
2C	SNVS_HP Time Alarm MSB Register (HPTAMR)	32	RW	0000_0000
30	SNVS_HP Time Alarm LSB Register (HPTALR)	32	RW	0000_0000
34	SNVS_LP Lock Register (LPLR)	32	RW	0000_0000
38	SNVS_LP Control Register (LPCR)	32	RW	0000_0020
4C	SNVS_LP Status Register (LPSR)	32	RW	0000_0008
5C	SNVS_LP Secure Monotonic Counter MSB Register (LPSMCMR)	32	RW	0000_0000
60	SNVS_LP Secure Monotonic Counter LSB Register (LPSMCLR)	32	RW	0000_0000
64	SNVS_LP Digital Low-Voltage Detector Register (LPLVDR)	32	RW	0000_0000
68	SNVS_LP General Purpose Register 0 (legacy alias) (LPGPR0_legacy_alias)	32	RW	0000_0000
90 - 9C	SNVS_LP General Purpose Registers 0 .. 3 (LPGPR0_alias - LPGPR3_alias)	32	RW	0000_0000
100 - 11C	SNVS_LP General Purpose Registers 0 .. 7 (LPGPR0 - LPGPR7)	32	RW	0000_0000
BF8	SNVS_HP Version ID Register 1 (HPVIDR1)	32	RO	003E_0104
BFC	SNVS_HP Version ID Register 2 (HPVIDR2)	32	RO	0600_0000

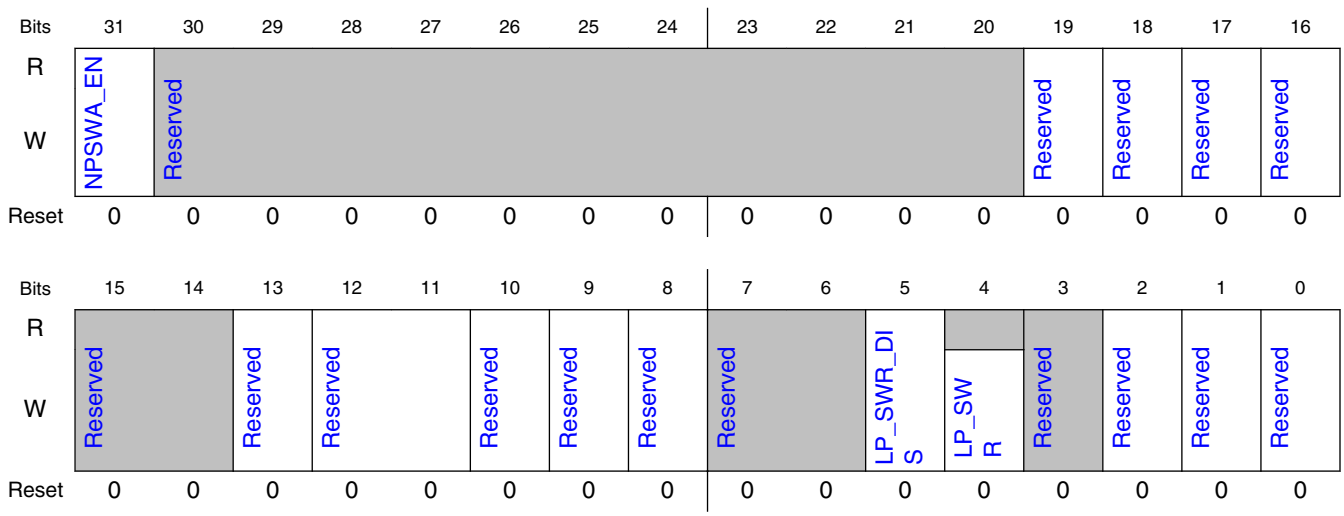
20.6.1.2 SNVS_HP Command Register (HPCOMR)

The SNVS_HP Command Register contains the command, configuration, and control bits for the SNVS block. This is a privileged write register.

20.6.1.2.1 Offset

Register	Offset
HPCOMR	4h

20.6.1.2.2 Diagram



20.6.1.2.3 Fields

Field	Description
31 NPSWA_EN	Non-Privileged Software Access Enable When set, allows non-privileged software to access all SNVS registers, including those that are privileged software read/write access only. 0 Only privileged software can access privileged registers 1 Any software can access privileged registers
30-20 —	Reserved
19 —	Reserved

Table continues on the next page...

Field	Description
18 —	Reserved
17 —	Reserved
16 —	Reserved
15-14 —	Reserved
13 —	Reserved
12-11 —	Reserved
10 —	Reserved
9 —	Reserved
8 —	Reserved
7-6 —	Reserved
5 LP_SWR_DIS	LP Software Reset Disable When set, disables the LP software reset. Once set, this bit can only be reset by the system reset. 0 - LP software reset is enabled 1 - LP software reset is disabled
4 LP_SWR	LP Software Reset When set to 1, the registers in the SNVS_LP section are reset. 0 - No Action 1 - Reset LP section
3 —	Reserved
2 —	Reserved
1 —	Reserved
0 —	Reserved

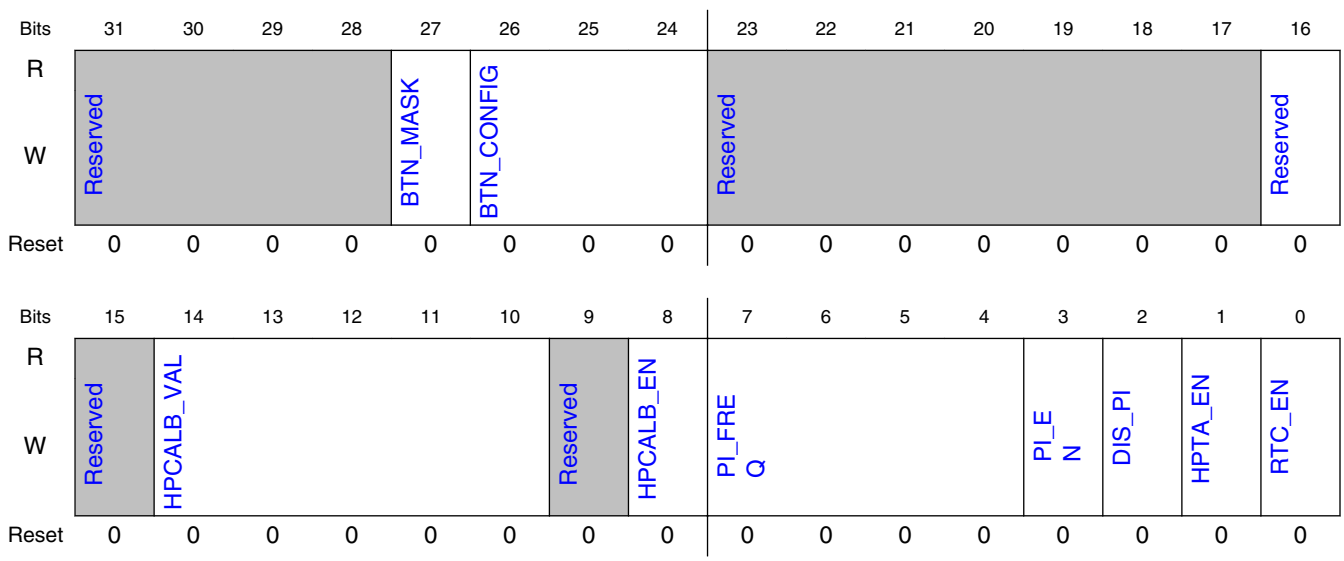
20.6.1.3 SNVS_HP Control Register (HPCR)

The SNVS_HP Control Register contains various control bits of the HP section of SNVS. This is *not* a privileged write register.

20.6.1.3.1 Offset

Register	Offset
HPCR	8h

20.6.1.3.2 Diagram



20.6.1.3.3 Fields

Field	Description
31-28 —	Reserved
27 BTN_MASK	Button interrupt mask. This bit is used to mask the button (BTN) interrupt request. 0: Interrupt disabled 1: Interrupt enabled
26-24 BTN_CONFIG	Button Configuration. This field is used to configure which feature of the button (BTN) input signal constitutes "active".

Table continues on the next page...

Field	Description
	000: Button signal is active high 001: Button signal is active low 010: Button signal is active on the falling edge 011: Button signal is active on the rising edge 100: Button signal is active on any edge All other patterns are Reserved
23-17 —	Reserved
16 —	Reserved
15 —	Reserved
14-10 HPCALB_VAL	HP Calibration Value Defines signed calibration value for the HP Real Time Counter. This field can be programmed only when RTC Calibration is disabled (HPCALB_EN is not set). This is a 5-bit 2's complement value, hence the allowable calibration values are in the range from -16 to +15 counts per 32768 ticks of the counter. 00000 - +0 counts per each 32768 ticks of the counter 00001 - +1 counts per each 32768 ticks of the counter 00010 - +2 counts per each 32768 ticks of the counter 01111 - +15 counts per each 32768 ticks of the counter 10000 - -16 counts per each 32768 ticks of the counter 10001 - -15 counts per each 32768 ticks of the counter 11110 - -2 counts per each 32768 ticks of the counter 11111 - -1 counts per each 32768 ticks of the counter
9 —	Reserved
8 HPCALB_EN	HP Real Time Counter Calibration Enabled Indicates that the time calibration mechanism is enabled. 0 - HP Timer calibration disabled 1 - HP Timer calibration enabled
7-4 PI_FREQ	Periodic Interrupt Frequency Defines frequency of the periodic interrupt. The interrupt is generated when a zero-to-one or one-to-zero transition occurs on the selected bit of the HP Real Time Counter and Real Time Counter and Periodic Interrupt are both enabled (RTC_EN and PI_EN are set). It is recommended to program this field when Periodic Interrupt is disabled (PI_EN is not set). The possible frequencies are: 0000 - - bit 0 of the HPRTCLR is selected as a source of the periodic interrupt 0001 - - bit 1 of the HPRTCLR is selected as a source of the periodic interrupt 0010 - - bit 2 of the HPRTCLR is selected as a source of the periodic interrupt 0011 - - bit 3 of the HPRTCLR is selected as a source of the periodic interrupt 0100 - - bit 4 of the HPRTCLR is selected as a source of the periodic interrupt 0101 - - bit 5 of the HPRTCLR is selected as a source of the periodic interrupt

Table continues on the next page...

Field	Description
	0110 - - bit 6 of the HPRTCLR is selected as a source of the periodic interrupt 0111 - - bit 7 of the HPRTCLR is selected as a source of the periodic interrupt 1000 - - bit 8 of the HPRTCLR is selected as a source of the periodic interrupt 1001 - - bit 9 of the HPRTCLR is selected as a source of the periodic interrupt 1010 - - bit 10 of the HPRTCLR is selected as a source of the periodic interrupt 1011 - - bit 11 of the HPRTCLR is selected as a source of the periodic interrupt 1100 - - bit 12 of the HPRTCLR is selected as a source of the periodic interrupt 1101 - - bit 13 of the HPRTCLR is selected as a source of the periodic interrupt 1110 - - bit 14 of the HPRTCLR is selected as a source of the periodic interrupt 1111 - - bit 15 of the HPRTCLR is selected as a source of the periodic interrupt
3 PI_EN	HP Periodic Interrupt Enable The periodic interrupt can be generated only if the HP Real Time Counter is enabled. 0 - HP Periodic Interrupt is disabled 1 - HP Periodic Interrupt is enabled
2 DIS_PI	Disable periodic interrupt in the functional interrupt 0 - Periodic interrupt will trigger a functional interrupt 1 - Disable periodic interrupt in the function interrupt
1 HPTA_EN	HP Time Alarm Enable When set, the time alarm interrupt is generated if the value in the HP Time Alarm Registers is equal to the value of the HP Real Time Counter. This bit syncs with the 32KHz clock. It won't update with the bus clock. 0 - HP Time Alarm Interrupt is disabled 1 - HP Time Alarm Interrupt is enabled
0 RTC_EN	HP Real Time Counter Enable. This bit syncs with the 32KHz clock. It won't update with the bus clock. 0 - RTC is disabled 1 - RTC is enabled

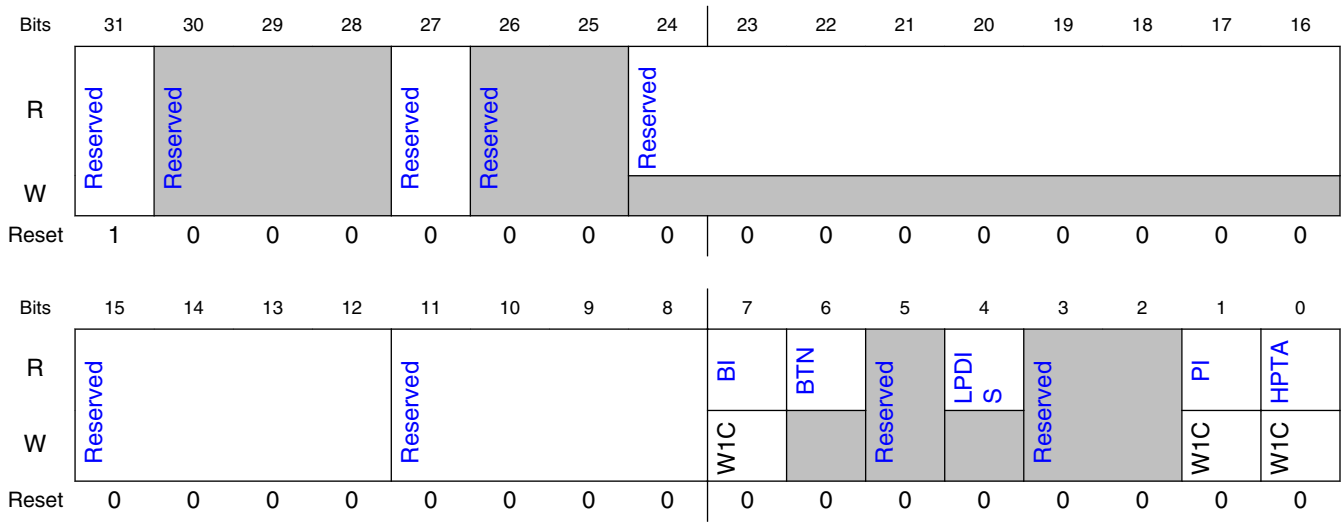
20.6.1.4 SNVS_HP Status Register (HPSR)

The HP Status Register reflects the internal state of the SNVS. This is *not* a privileged write register.

20.6.1.4.1 Offset

Register	Offset
HPSR	14h

20.6.1.4.2 Diagram



20.6.1.4.3 Fields

Field	Description
31 —	Reserved
30-28 —	Reserved
27 —	Reserved
26-25 —	Reserved
24-16 —	Reserved
15-12 —	Reserved
11-8 —	Reserved
7 BI	Button Interrupt Signal ipi_snvs_btn_int_b was asserted.
6 BTN	Button Value of the BTN input. This is the external button used for PMIC control. 0: BTN not pressed 1: BTN pressed
5	Reserved

Table continues on the next page...

Field	Description
—	
4 LPDIS	Low Power Disable If 1, the low power section has been disabled by means of an input signal to SNVS.
3-2 —	Reserved
1 PI	Periodic Interrupt Indicates that periodic interrupt has occurred since this bit was last cleared. 0 - No periodic interrupt occurred. 1 - A periodic interrupt occurred.
0 HPTA	HP Time Alarm Indicates that the HP Time Alarm has occurred since this bit was last cleared. 0 - No time alarm interrupt occurred. 1 - A time alarm interrupt occurred.

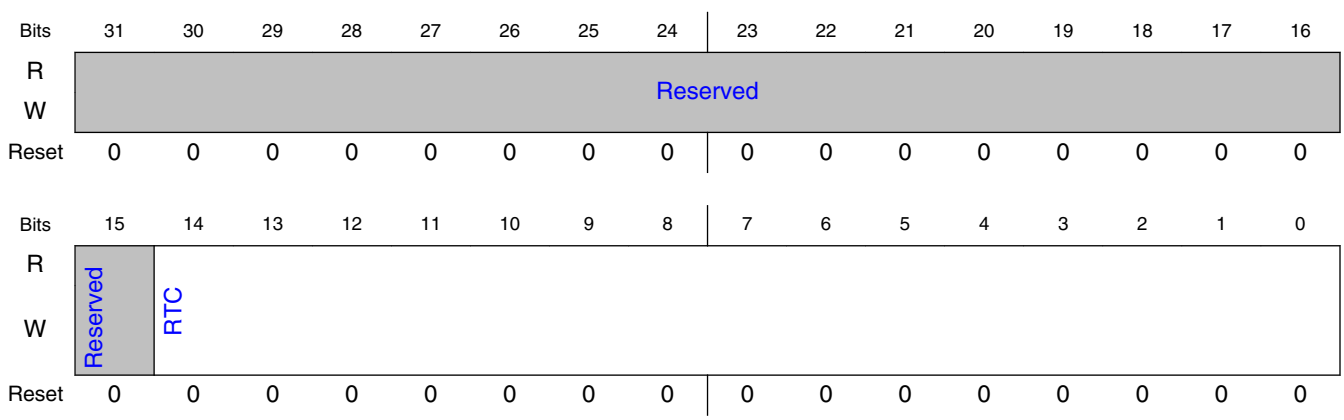
20.6.1.5 SNVS_HP Real Time Counter MSB Register (HPRTCMR)

The SNVS_HP Real Time Counter MSB register contains the 15 most-significant bits of the HP Real Time Counter. This is *not* a privileged write register.

20.6.1.5.1 Offset

Register	Offset
HPRTCMR	24h

20.6.1.5.2 Diagram



20.6.1.5.3 Fields

Field	Description
31-15 —	Reserved
14-0 RTC	HP Real Time Counter The most-significant 15 bits of the RTC. This register can be programmed only when RTC is not active (RTC_EN bit is not set).

20.6.1.6 SNVS_HP Real Time Counter LSB Register (HPRTCLR)

The SNVS_HP Real Time Counter LSB register contains the 32 least-significant bits of the HP real time counter. This is *not* a privileged write register.

20.6.1.6.1 Offset

Register	Offset
HPRTCLR	28h

20.6.1.6.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	RTC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RTC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

20.6.1.6.3 Fields

Field	Description
31-0 RTC	HP Real Time Counter least-significant 32 bits. This register can be programmed only when RTC is not active (RTC_EN bit is not set).

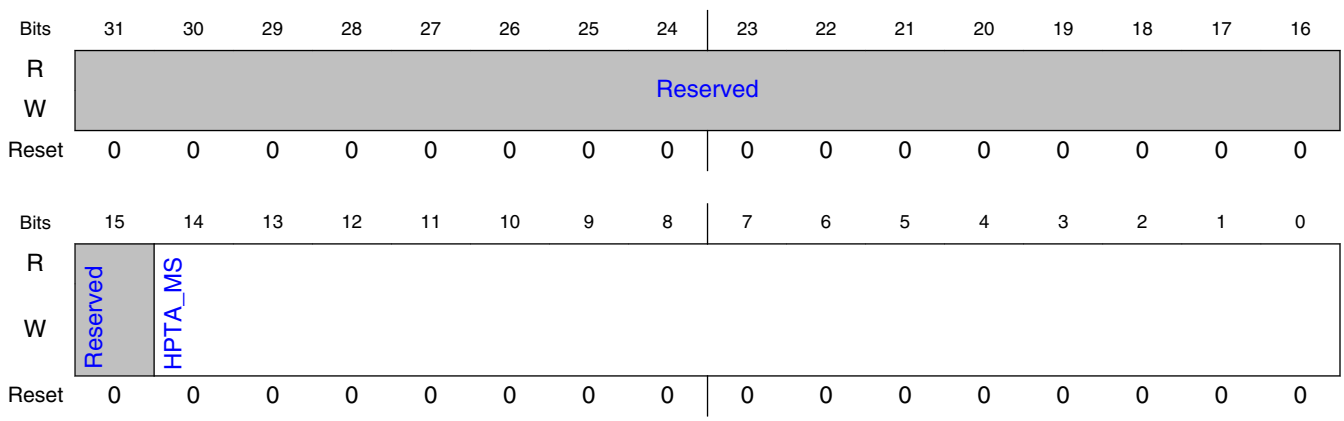
20.6.1.7 SNVS_HP Time Alarm MSB Register (HPTAMR)

The SNVS_HP Time Alarm MSB register contains the most-significant bits of the SNVS_HP Time Alarm value. This is *not* a privileged write register.

20.6.1.7.1 Offset

Register	Offset
HPTAMR	2Ch

20.6.1.7.2 Diagram



20.6.1.7.3 Fields

Field	Description
31-15 —	Reserved

Table continues on the next page...

Field	Description
14-0 HPTA_MS	HP Time Alarm, most-significant 15 bits. This register can be programmed only when HP time alarm is disabled (HPTA_EN bit is not set).

20.6.1.8 SNVS_HP Time Alarm LSB Register (HPTALR)

The SNVS_HP Time Alarm LSB register contains the 32 least-significant bits of the SNVS_HP Time Alarm value. This is *not* a privileged write register.

20.6.1.8.1 Offset

Register	Offset
HPTALR	30h

20.6.1.8.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	HPTA_LS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	HPTA_LS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

20.6.1.8.3 Fields

Field	Description
31-0 HPTA_LS	HP Time Alarm, 32 least-significant bits. This register can be programmed only when HP time alarm is disabled (HPTA_EN bit is not set).

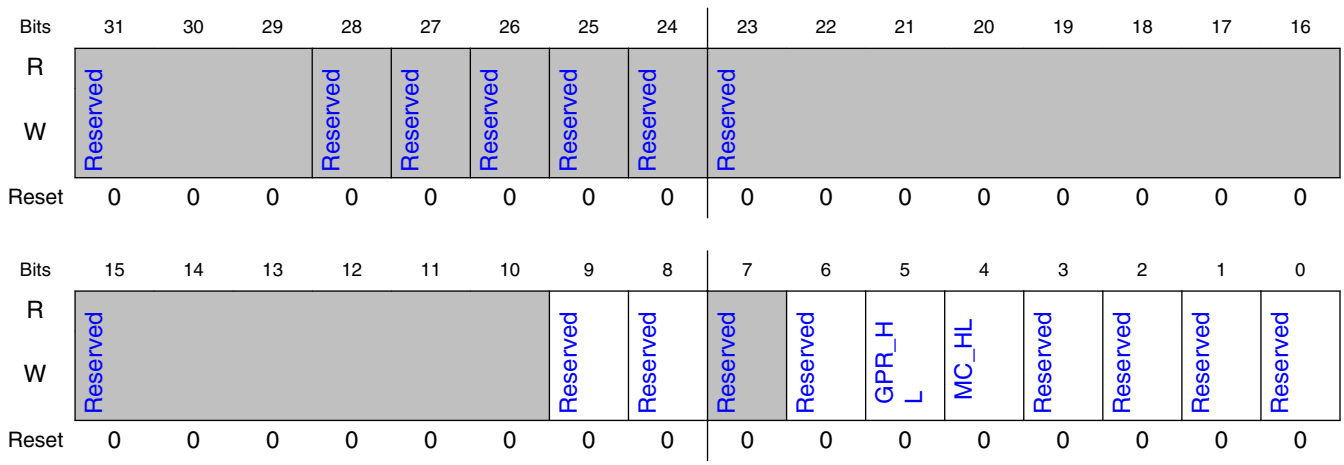
20.6.1.9 SNVS_LP Lock Register (LPLR)

The SNVS_LP Lock Register contains lock bits for the SNVS_LP registers. This is a privileged write register.

20.6.1.9.1 Offset

Register	Offset
LPLR	34h

20.6.1.9.2 Diagram



20.6.1.9.3 Fields

Field	Description
31-29 —	Reserved
28 —	Reserved
27 —	Reserved
26 —	Reserved
25 —	Reserved

Table continues on the next page...

Field	Description
24 —	Reserved
23-10 —	Reserved
9 —	Reserved
8 —	Reserved
7 —	Reserved
6 —	Reserved
5 GPR_HL	General Purpose Register Hard Lock When set, prevents any writes to the GPR. Once set, this bit can only be reset by the LP POR. 0 - Write access is allowed. 1 - Write access is not allowed.
4 MC_HL	Monotonic Counter Hard Lock When set, prevents any writes (increments) to the MC Registers and MC_ENV bit. Once set, this bit can only be reset by the LP POR. 0 - Write access (increment) is allowed. 1 - Write access (increment) is not allowed.
3 —	Reserved
2 —	Reserved
1 —	Reserved
0 —	Reserved

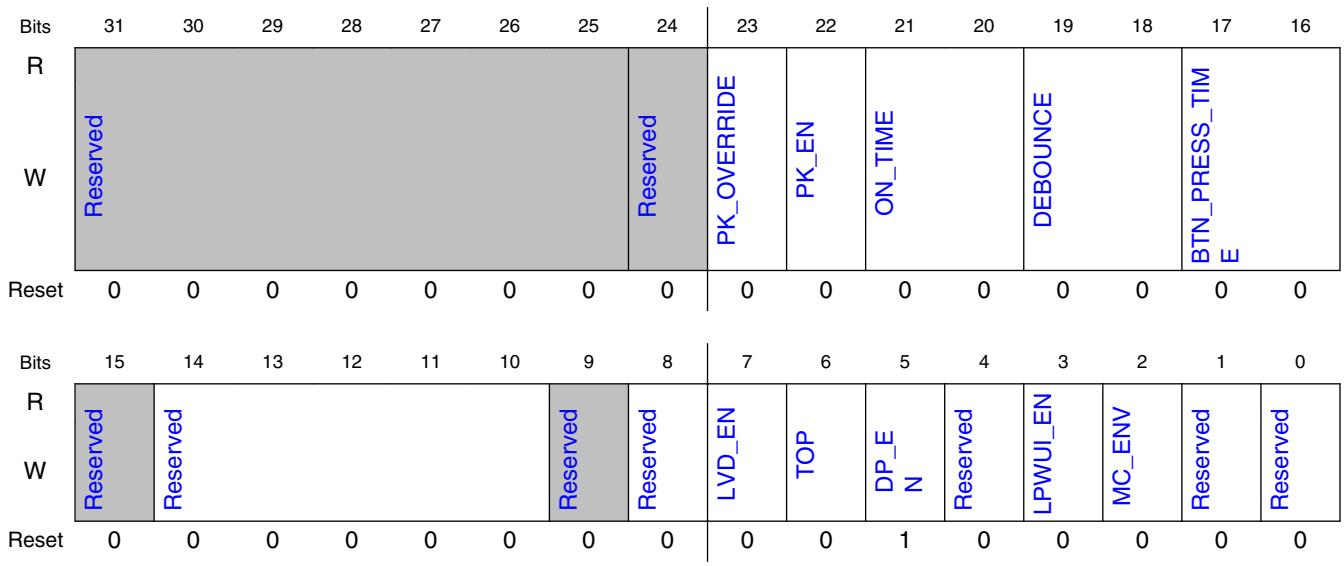
20.6.1.10 SNVS_LP Control Register (LPCR)

The SNVS_LP Control Register contains various control bits of the LP section of SNVS. This is a privileged write register.

20.6.1.10.1 Offset

Register	Offset
LPCR	38h

20.6.1.10.2 Diagram



20.6.1.10.3 Fields

Field	Description
31-25 —	Reserved
24 —	Reserved
23 PK_OVERRIDE	PMIC On Request Override The value written to PK_OVERRIDE will be asserted on output signal snvs_lp_pk_override. That signal is used to override the IOMUX control for the PMIC I/O pad.
22 PK_EN	PMIC On Request Enable The value written to PK_EN will be asserted on output signal snvs_lp_pk_en. That signal is used to turn off the pullup/pulldown circuitry in the PMIC I/O pad.
21-20 ON_TIME	The ON_TIME field is used to configure the period of time after BTN is asserted before pmic_en_b is asserted to turn on the SoC power. 00: 500msec off->on transition time 01: 50msec off->on transition time 10: 100msec off->on transition time

Table continues on the next page...

Field	Description
	11: 0msec off->on transition time
19-18 DEBOUNCE	This field configures the amount of debounce time for the BTN input signal. 00: 50msec debounce 01: 100msec debounce 10: 500msec debounce 11: 0msec debounce
17-16 BTN_PRESS_TIME	This field configures the button press time out values for the PMIC Logic. 00 : 5 secs 01 : 10 secs 10 : 15 secs 11 : long press disabled (pmic_en_b will not be asserted regardlessof how long BTN is asserted)
15 —	Reserved
14-10 —	Reserved
9 —	Reserved
8 —	Reserved
7 LVD_EN	Digital Low-Voltage Event Enable By default the detection of a low-voltage event does not cause the pmic_en_b signal to be asserted. Setting the Digital Low-Voltage Event Enable bit to 1 enables the low-voltage event for the PMIC. 0 - disabled 1 - enabled
6 TOP	Turn off System Power Asserting this bit causes a signal to be sent to the Power Management IC to turn off the system power. This bit will clear once power is off. This bit is only valid when the Dumb PMIC is enabled. 0 - Leave system power on. 1 - Turn off system power.
5 DP_EN	Dumb PMIC Enabled When set, software can control the system power. When cleared, the system requires a Smart PMIC to automatically turn power off. 0 - Smart PMIC enabled. 1 - Dumb PMIC enabled.
4 —	Reserved
3 LPWUI_EN	LP Wake-Up Interrupt Enable This interrupt line should be connected to the external pin and is intended to inform the external chip about an SNVS_LP event (MC rollover, SRTC rollover, or time alarm). This wake-up signal can be asserted only when the chip (HP section) is powered down, and the LP section is isolated. 0 LP wake-up interrupt is disabled.

Table continues on the next page...

Field	Description
	1 LP wake-up interrupt is enabled.
2 MC_ENV	Monotonic Counter Enabled and Valid When set, the MC can be incremented (by write transaction to the LPSMCMR or LPSMCLR). Once MC_SL or MC_HL bit is set this bit can be changed only by LP POR. 0 - MC is disabled or invalid. 1 - MC is enabled and valid.
1 —	Reserved
0 —	Reserved

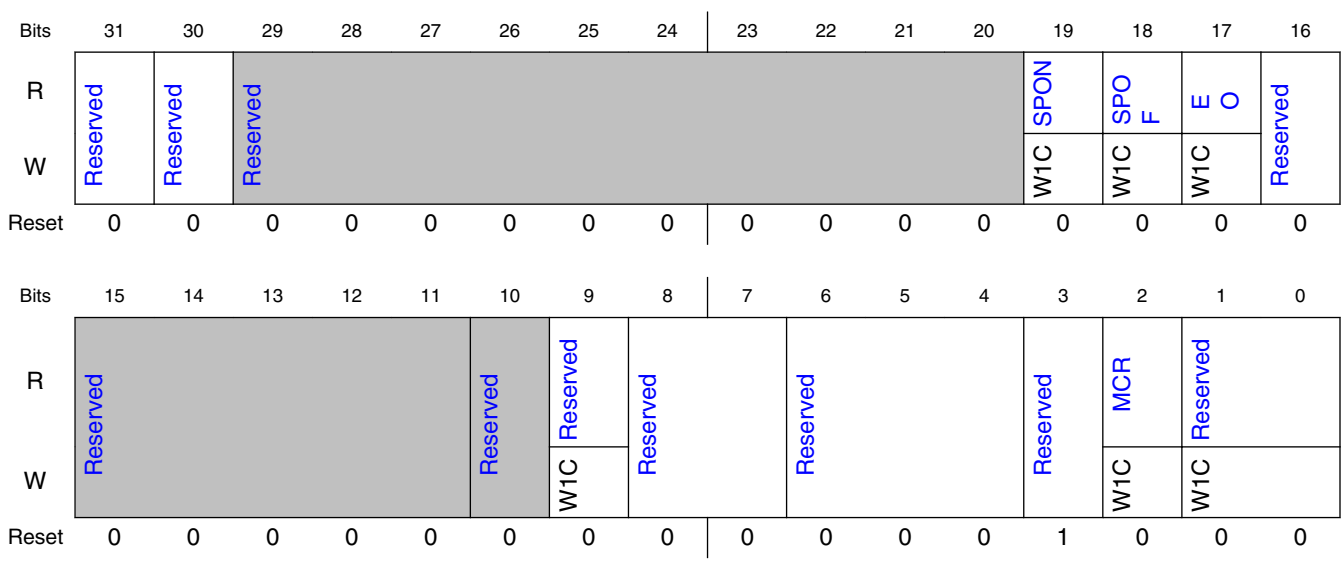
20.6.1.11 SNVS_LP Status Register (LPSR)

The SNVS_LP Status Register reflects the internal state and behavior of the SNVS_LP. This is a privileged write register.

20.6.1.11.1 Offset

Register	Offset
LPSR	4Ch

20.6.1.11.2 Diagram



20.6.1.11.3 Fields

Field	Description
31 —	Reserved
30 —	Reserved
29-20 —	Reserved
19 SPON	Set Power On The SPON bit is set when the set_pwr_on_irq interrupt is triggered, which happens when the power button is pressed longer than the configured debounce time. Writing to the SPON bit will clear the set_pwr_ofn_irq interrupt. 0 - Set Power On Interrupt was not detected. 1 - Set Power On Interrupt was detected.
18 SPOF	Set Power Off The SPO bit is set when the power button is pressed longer than the configured debounce time. Writing to the SPO bit will clear the set_pwr_off_irq interrupt. 0 - Set Power Off was not detected. 1 - Set Power Off was detected.
17 EO	Emergency Off This bit is set when a power off is requested. 0 - Emergency off was not detected. 1 - Emergency off was detected.
16 —	Reserved
15-11 —	Reserved
10 —	Reserved
9 —	Reserved
8-7 —	Reserved
6-4 —	Reserved
3 —	Reserved
2 MCR	Monotonic Counter Rollover 0 - MC has not reached its maximum value.

Table continues on the next page...

Field	Description
	1 - MC has reached its maximum value.
1-0 —	Reserved

20.6.1.12 SNVS_LP Secure Monotonic Counter MSB Register (LPSMCMR)

The SNVS_LP Secure Monotonic Counter MSB Register contains the monotonic counter era bits and the most-significant 16 bits of the monotonic counter. The monotonic counter is incremented by one if there is a write command to the LPSMCMR or LPSMCLR register. This is a non-privileged read-only register.

20.6.1.12.1 Offset

Register	Offset
LPSMCMR	5Ch

20.6.1.12.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	MC_ERA_BITS															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	MON_COUNTER															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

20.6.1.12.3 Fields

Field	Description
31-16 MC_ERA_BITS	Monotonic Counter Era Bits

Table continues on the next page...

Field	Description
	These bits are inputs to the module and typically connect to fuses. When the Monotonic Counter is in use (i.e. enabled and valid and powered by an uninterrupted power source), and the boot software detects that the Monotonic Counter most-significant 16 Bits and Monotonic Counter LSB Register have been reset (MC_ENV=0), the boot software can take action to ensure that the value in the monotonic counter remains monotonic (i.e. never decreasing). The action is to blow an additional MC_ERA_BITS fuse. Since the MC_ERA_BITS field forms the most-significant field of the monotonic counter, blowing an additional fuse guarantees that the new monotonic counter value is higher than any previous value.
15-0 MON_COUNTER	Monotonic Counter most-significant 16 Bits **Note that writing to this register does *not* change the value of this field to the value that was written.** **The 48-bit monotonic counter value (consisting of LPSMCMR[MON_COUNTER] prepended to LPSMCLR[MON_COUNTER]) is incremented by one when:** • A write transaction to the LPSMCMR or LPSMCLR register is detected. • The MC_ENV bit is set. • MC_SL and MC_HL bits are not set. This value can be reset only by LP POR.

20.6.1.13 SNVS_LP Secure Monotonic Counter LSB Register (LPSMCLR)

The SNVS_LP Secure Monotonic Counter LSB Register contains the 32 least-significant bits of the monotonic counter. The MC is incremented by one if there is a write command to the LPSMCMR or LPSMCLR register. This is a non-privileged read-only register.

20.6.1.13.1 Offset

Register	Offset
LPSMCLR	60h

20.6.1.13.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	MON_COUNTER															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MON_COUNTER															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

20.6.1.13.3 Fields

Field	Description
31-0 MON_COUNTER	Monotonic Counter bits **Note that writing to this register does *not* change the value of this field to the value that was written.** **The 48-bit monotonic counter value (consisting of LPSMCMR[MON_COUNTER] prepended to LPSMCLR[MON_COUNTER]) is incremented by one when:** • A write transaction to the LPSMCMR or LPSMCLR register is detected. • The MC_ENV bit is set. • MC_SL and MC_HL bits are not set. This value can be reset only by LP POR.

20.6.1.14 SNVS_LP Digital Low-Voltage Detector Register (LPLVDR)

The SNVS_LP Digital Low-Voltage Detector Register is a 32-bit read/write register that is used for storing the low-voltage detector value, as described in Digital Low-Voltage Detector (LVD). This is a privileged write register.

20.6.1.14.1 Offset

Register	Offset
LPLVDR	64h

20.6.1.14.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	LVD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	LVD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

20.6.1.14.3 Fields

Field	Description
31-0 LVD	Low-Voltage Detector Value

20.6.1.15 SNVS_LP General Purpose Register 0 (legacy alias) (LPGPR0_legacy_alias)

See register SNVS_LP General Purpose Registers 0 .. 7 (LPGPR0 - LPGPR7).

20.6.1.15.1 Offset

Register	Offset
LPGPR0_legacy_alias	68h

20.6.1.15.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	GPR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GPR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

20.6.1.15.3 Fields

Field	Description
31-0 GPR	General Purpose Register When GPR_SL or GPR_HL bit is set, the register cannot be programmed.

20.6.1.16 SNVS_LP General Purpose Registers 0 .. 3 (LPGPR0_alias - LPGPR3_alias)

See register SNVS_LP General Purpose Registers 0 .. 7 (LPGPR0 - LPGPR7).

20.6.1.16.1 Offset

Register	Offset
LPGPR0_alias	90h
LPGPR1_alias	94h
LPGPR2_alias	98h
LPGPR3_alias	9Ch

20.6.1.16.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	GPR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GPR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

20.6.1.16.3 Fields

Field	Description
31-0 GPR	General Purpose Register When GPR_SL or GPR_HL bit is set, the register cannot be programmed.

20.6.1.17 SNVS_LP General Purpose Registers 0 .. 7 (LPGPR0 - LPGPR7)

The SNVS_LP General Purpose Register is a 256-bit read/write register located in SNVS_LP, which can be used by any application for retaining data during an SoC power-down mode. This is a privileged read/write register. The full GPR register is accessed as 8 32-bit registers located in successive word addresses starting at offset 100h. For backward compatibility with earlier versions of SNVS, LPGPR0..LPGPR3 are aliased at the earlier offset of 90h and LPGPR0 is also aliased at its original offset of 68h. New software should access the GPR register at the preferred offset of 100h. The GPR will be automatically zeroized when an enabled security event occurs, unless GPR zeroization is disabled via the GPR_Z_DIS bit in the LP Control Register.

20.6.1.17.1 Offset

For a = 0 to 7:

Register	Offset
LPGPRa	100h + (a × 4h)

20.6.1.17.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	GPR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GPR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

20.6.1.17.3 Fields

Field	Description
31-0 GPR	General Purpose Register When GPR_SL or GPR_HL bit is set, the register cannot be programmed.

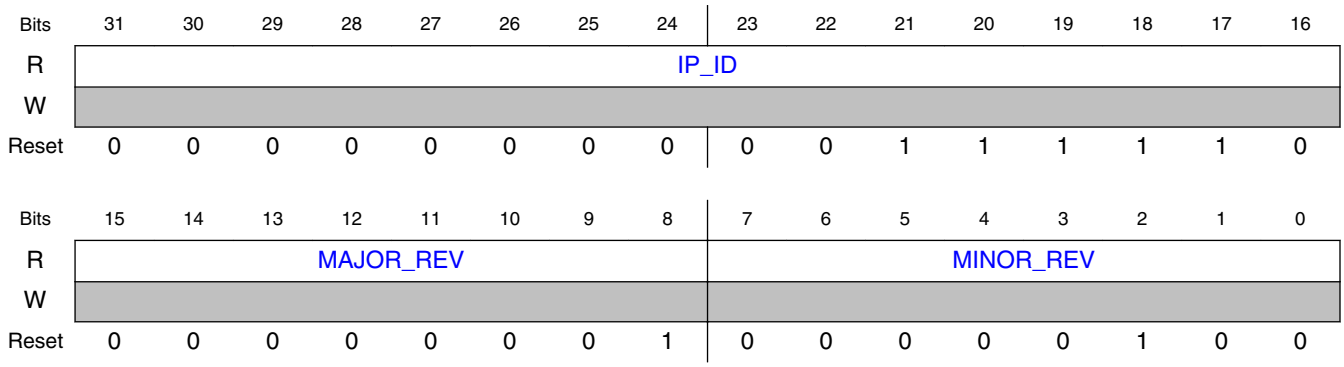
20.6.1.18 SNVS_HP Version ID Register 1 (HPVIDR1)

The SNVS_HP Version ID Register 1 is a non-privileged read-only register that contains the current version of the SNVS. The version consists of a module ID, a major version number, and a minor version number.

20.6.1.18.1 Offset

Register	Offset
HPVIDR1	BF8h

20.6.1.18.2 Diagram



20.6.1.18.3 Fields

Field	Description
31-16 IP_ID	SNVS block ID
15-8 MAJOR_REV	SNVS block major version number
7-0 MINOR_REV	SNVS block minor version number

20.6.1.19 SNVS_HP Version ID Register 2 (HPVIDR2)

The SNVS_HP Version ID Register 2 is a non-privileged read-only register that indicates the current version of the SNVS. Version ID register 2 consists of the following fields: integration options, ECO revision, and configuration options.

20.6.1.19.1 Offset

Register	Offset
HPVIDR2	BFCh

20.6.1.19.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	IP_ERA								Reserved							
W																
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ECO_REV								Reserved							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

20.6.1.19.3 Fields

Field	Description
31-24 IP_ERA	IP Era 00h - Era 1 or 2 03h - Era 3 04h - Era 4 05h - Era 5 06h - Era 6
23-16 —	Reserved
15-8 ECO_REV	SNVS ECO Revision The engineering change order revision number for this release of SNVS.
7-0 —	Reserved

Chapter 21
System Reset Controller (SRC)

21.1 Chip-specific SRC information

Table 21-1. Reference links to related information

Topic	Related module(s)	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
Interrupts, DMA Events and XBAR Assignments	-	Interrupts, DMA Events and XBAR Assignments

21.2 Overview

The System Reset Controller (SRC) is responsible for the generation of all the system reset signals and boot argument latching.

21.2.1 Features

The SRC includes the following features.

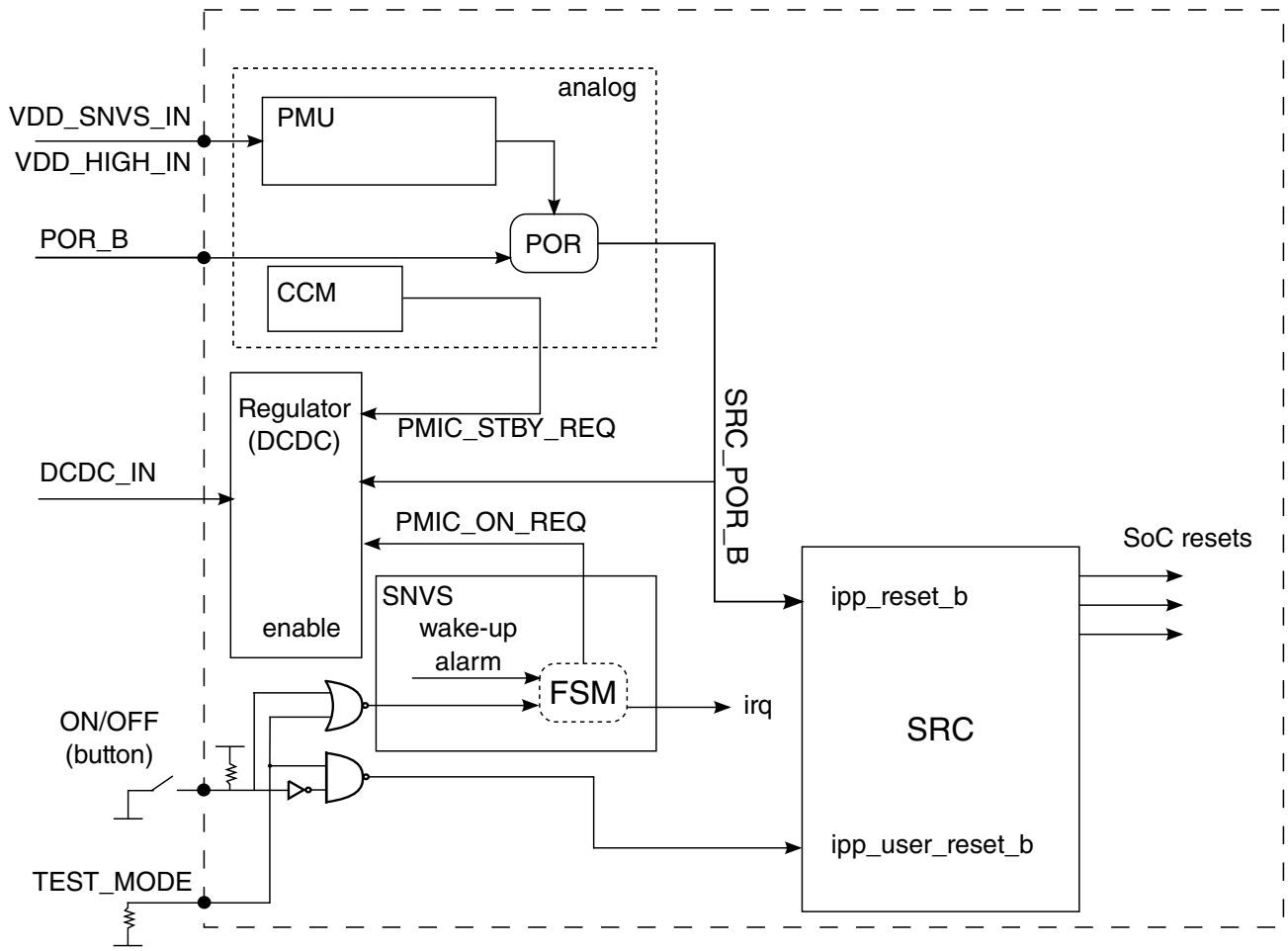
- Receives and handles the resets from all the reset sources
- Resets the appropriate domains based upon the resets sources and the nature of the reset
- Latches the SRC_BOOT_MODE pins and common configuration signals from the internal fuse

21.3 Functional Description

The reset controller determines the source and the type of reset, such as POR, COLD, and performs the necessary reset qualification and stretching sequences. Based on the type of reset, the reset logic generates the reset sequence for the entire IC. Whenever the chip is powered on, the reset is issued through SRC_ONOFF signal and the entire chip is reset.

21.3.1 Reset and Power-up Flow

The chip presumes the following reset and power-up flow:



Note: PMIC_ON_REQ acts as an active high-signal
1 or high Z = ON
0 = OFF

Figure 21-1. Chip reset scheme under PMU control

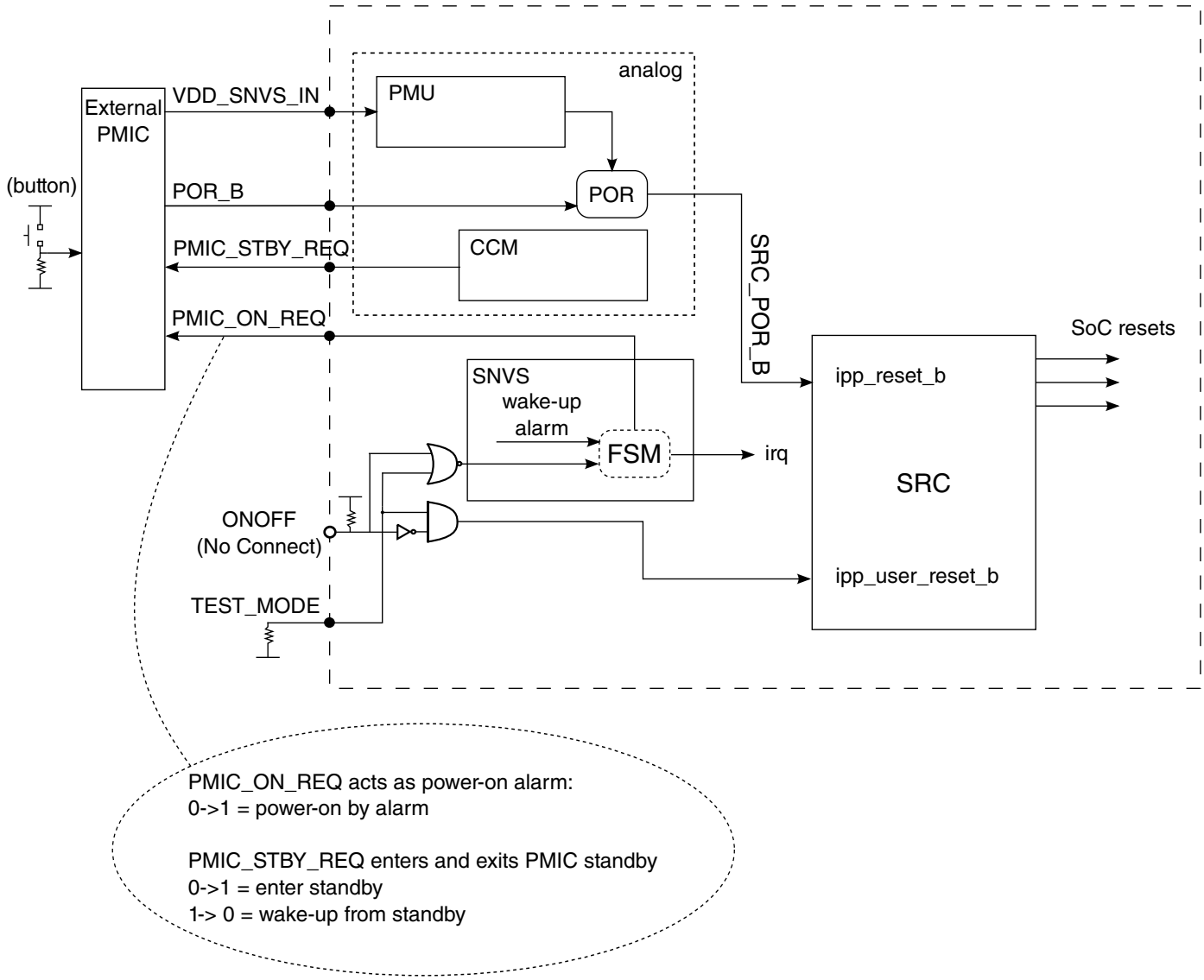


Figure 21-2. Chip reset scheme under external PMIC control

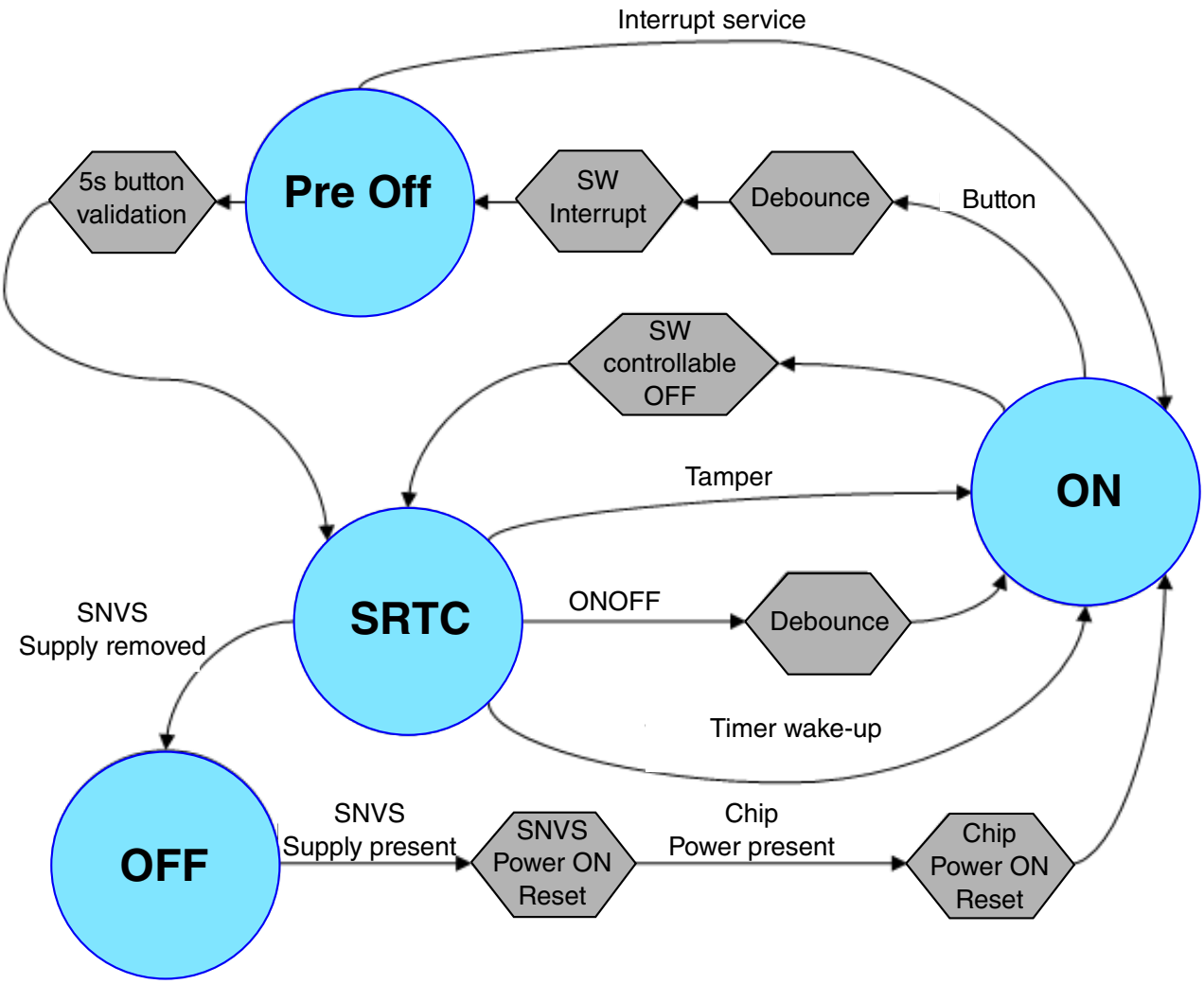


Figure 21-3. Chip on/off state flow diagram

21.3.2 Finite-State Machine (FSM)

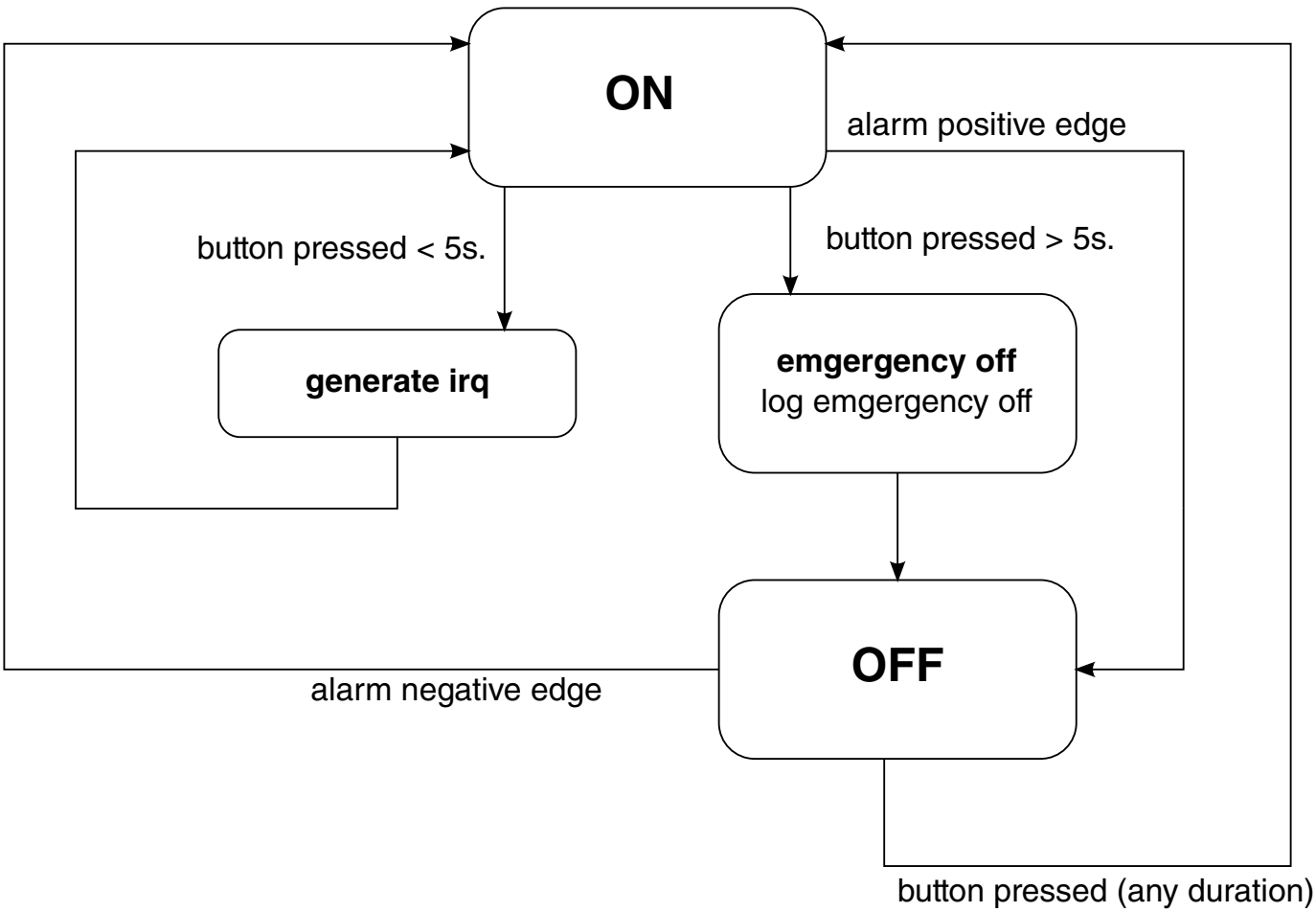


Figure 21-4. FSM

21.3.3 Power mode transitions

Table 21-2. Power mode transitions

Power mode	Configuration with external PMIC	Configuration with internal PMIC
ON, first time	1. Either coin cell or SoC power supply is connected to SNVS. 2. When button is pressed, PMIC powers on.	1. Either coin cell or SoC power supply is connected to SNVS, PMIC_ON_REQ goes to '1'. 2. Drive DCDC_PSWITCH to high, 1 ms after DCDC_IN is powered. 3. DCDC regulator is enabled.
Normal ON to OFF, by button	1. Button is pressed for a short duration on the external PMIC. 2. Interrupt request (irq) is sent to SoC from external PMIC.	1. SoC button is pressed for a short duration. 2. Interrupt request (irq) is sent to SoC from FSM. 3. Alarm timer is set up by software routine and started.

Table continues on the next page...

Table 21-2. Power mode transitions (continued)

Power mode	Configuration with external PMIC	Configuration with internal PMIC
	3. SoC is programming PMIC for power off when standby is asserted. 4. In CCM STOP mode, Standby is asserted, PMIC gates SoC supplies.	4. Upon alarm_in assertion to '1', PMIC_ON_REQ goes '0'. 5. DCDC regulator goes OFF.
Emergency ON to OFF, by button	1. Button is pressed for an extended time on the external PMIC. 2. PMIC is powering off.	1. Button is pressed for longer than 5 seconds on the SoC. 2. FSM validates button pressed for 5 seconds. 3. Emergency power off is logged, PMIC_ON_REQ goes '0', alarm_mask goes '1'. 4. DCDC regulator goes OFF.
OFF to ON, by button	1. Button is pressed on the external PMIC. 2. PMIC powers ON.	1. Button is pressed on the SoC. 2. PMIC_ON_REQ goes '1', alarm_mask goes '0'. 3. DCDC regulator powers ON.
OFF to ON, by timer alarm	1. Timer alarm in SNVS is programmed by software before SoC goes OFF. 2. SoC enters OFF mode. 3. Upon timer limit, wake up alarm goes '0'. PMIC_ON_REQ goes '1'. 4. PMIC receives assertion of PMIC_ON_REQ and wakes up.	1. Timer alarm in SNVS is programmed by software before SoC goes OFF. 2. SoC enters OFF mode. 3. Upon timer limit, wake up alarm goes '0'. PMIC_ON_REQ goes '1'. 4. DCDC regulator is enabled by PMIC_ON_REQ = 1.

21.3.4 Reset Control

This section details the reset control of this device.

21.3.4.1 Reset inputs and outputs

The reset control logic receives reset requests from all potential reset sources. All the immediate sources of reset are directly passed to the reset stretching block, whereas the resets requiring qualification are passed on to the reset qualification logic before they are sent to the reset stretching block.

All reset inputs and outputs are described in the following figure:

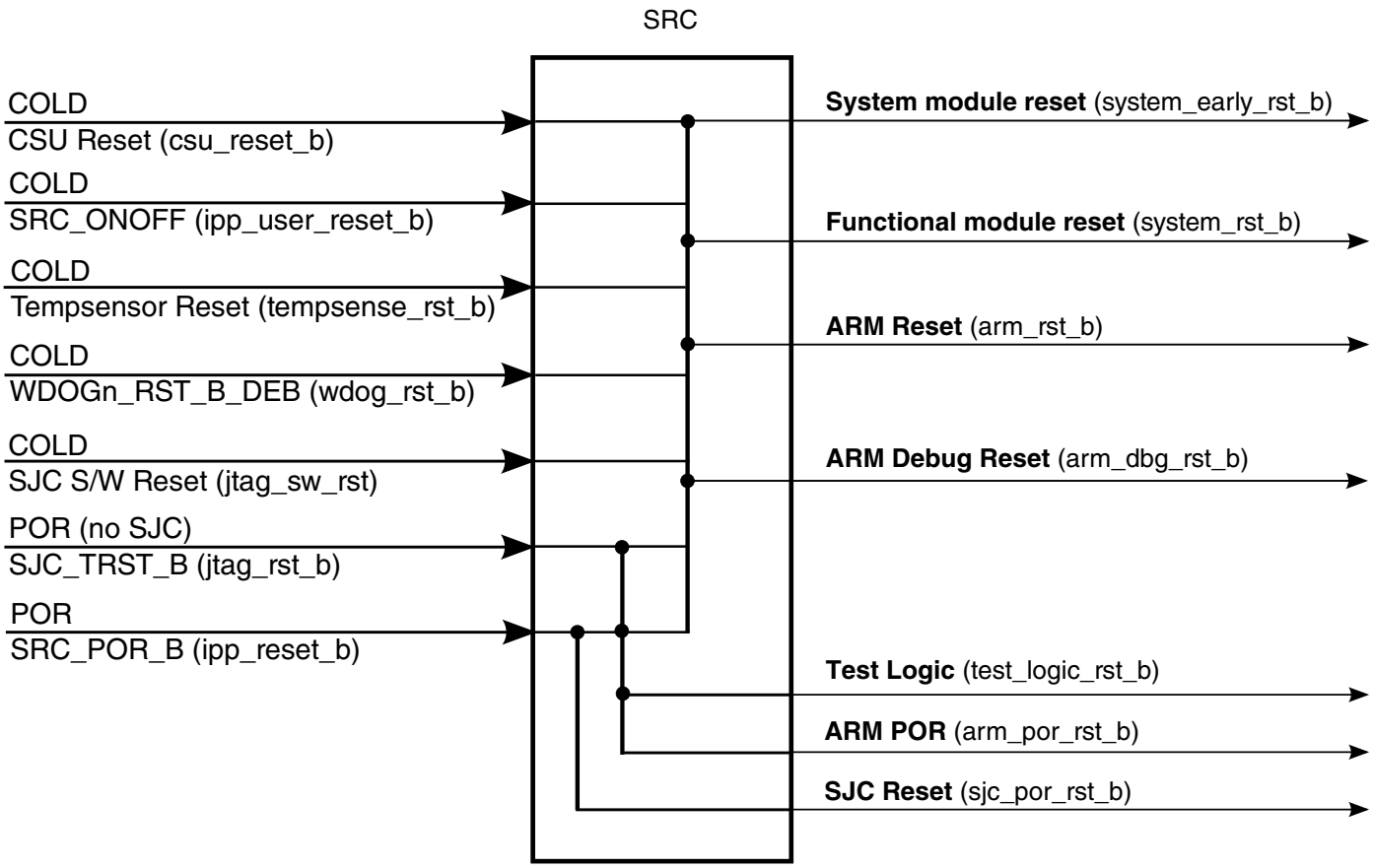


Figure 21-5. SRC inputs and outputs

The reset types and modules they affect are shown in Table 21-3.

NOTE

All resets are expected to be active low except jtag_sw_rst.

Table 21-3. SRC reset functionality

SoC Modules	POR	COLD
System modules (PLLs, fuses, etc)	yes	yes
Functional modules	yes	yes
Arm	yes	yes
IOMUXC	yes	no
Arm debug	yes	no
SJC	yes	no

The reset priorities are POR (strongest) and COLD (weakest). If a stronger reset is asserted during the sequence of a weaker reset, then the weaker sequence will be overridden, and the stronger reset sequence will commence. There is no priority within a reset type (POR, etc). If a reset is asserted during the reset sequence of the same type, the reset sequence will be interrupted and restarted.

The following lists the functionality of each of these reset outputs:

- system_early_rst_b - Resets the system modules that need to start first as CCM, OCOTP_CTRL, FUSEBOX, etc.
- system_rst_b - Resets functional modules
- arm_rst_b - Resets Arm module (on regular system reset)
- arm_por_rst_b - Resets Arm POR input
- arm_dbg_rst_b - Reset debug logic of Arm
- test_logic_rst_b - Reset test logic (IOMUXC, DAP)
- sjc_por_rst_b - Reset to SJC

NOTE

It is assumed that each reset source will deassert after its assertion, either due to reset generated to the system from SRC, or by negation of the reset source (if it came from an external source to the chip). In the latter case, the reset source is assumed to be held for at least 2 XTALI clocks so it can be sampled by SRC.

21.3.4.2 Reset Handling

21.3.4.2.1 POR (SRC_POR_B)

SRC_POR_B is an external reset signal of the SRC module. When the chip is powered up, the reset signal is passed through the POR_B pin indicating power-up sequence. The SRC resets the entire chip including the JTAG (SJC) module. All SRC registers will be reset during the POR sequence.

As soon as SRC_POR_B occurs, all resets are asserted and the entire chip is reset by SRC. The SRC_POR_B is stretched for 2 XTALI cycles and the stretching sequence takes place after 2 XTALI clocks of POR_B pin deassertion.

The sjc_por_rst_b signal is deasserted together with SRC_POR_B signal. The output is also deasserted after the stretching of SRC_POR_B has deasserted.

After the above resets deassert, system_early_rst_b reset is deasserted after 2 XTALI clocks. The system_early_rst_b is used for the CCM and PLL-IPs to start generating PLL clock ouputs and the system root clocks.

When the system root clocks are ready, the CCM will assert system_clk_ready signal. This signal is generated during the start sequence in the CCM and it involves the preparation of the PLLs to generate clock roots for functional operation.

SRC then enables OCOTP_CTRL and fusebox clocks, so that fuses can be loaded to OCOTP_CTRL.

- SRC will prepare the boot information
- After 8 ipg cycles, resets to all modules will be de-asserted
- After 8 ipg cycles, system clocks will be enabled (en_system_clk).

21.3.4.2.2 COLD RESET

The sequence is similar to SRC_POR_B except the IOMUXC, SJC and debug are not reset.

After the reset source deasserts, system_early_rst_b reset is deasserted after at least 2 XTALI clocks. The system_early_rst_b is used for the CCM and PLL-IPs to start generating PLL clock outputs and the system root clocks.

After the system root clocks are ready, the CCM will assert system_clk_ready signal. This signal is generated during the start sequence in the CCM and it involves the preparation of the PLLs to generate clock roots for functional operation. See CCM for more information.

After system_clk_ready arrives at the SRC, it will enable OCOTP_CTRL and fusebox clocks, so that fuses can be loaded to OCOTP_CTRL. OCOTP_CTRL will notify with iim_ready_flag when the fusebox loading finishes.

- SRC will prepare the boot information
- After 8 ipg cycles resets to all modules will be deasserted
- After 8 ipg cycles, system clocks will be enabled (en_system_clk).

21.3.5 Parallel Reset Requests

SRC will follow the following rules in the case of parallel reset requests:

1. The order of strength of resets is POR - strongest, COLD - weakest
2. If a stronger reset is asserted during weaker reset sequence, then the stronger reset will take over and the stronger reset process will commence. The following cases fall into this category:
    - POR reset request in the middle of cold reset process - the cold will be stopped and the POR sequence will start.

3. If a weaker reset is asserted during stronger reset sequence, then the stronger reset sequence will continue without interference. If at the end of the stronger reset process the weaker request is still asserted then the weaker sequence will commence. The following cases fall into this category:
   - COLD reset requests in the middle of POR reset process - the POR process will continue without interference.
4. If a similar reset request is asserted during the process of reset handling, then the process of reset handling will start over (with the same process). The following cases fall into this category:
   - POR reset request in the middle of POR reset process - the POR process will start over.
   - COLD reset request in the middle of COLD reset process - the COLD process will start over.

21.3.6 Boot Mode Control

21.3.6.1 BOOT_MODE Pin Latching

The exact boot sequence is controlled by the values of the BOOT_MODE pins on this device.

The value of the BOOT_MODE pins will be latched on de-assertion of POR reset. After latching, the values of the BOOT_MODE pins are used to determine the booting options of the core as described in the SRC_SBMRx registers.

The boot mode general purpose bits can be provided to the SRC from either e-fuses or GPIO signals. The gpio_bt_sel e-fuse defines the source to be used to derive the boot information. When gpio_bt_sel is set, e-fuses are used. When cleared, GPIO signals are used.

The boot information is provided in SRC_SBMR1 and SRC_SBMR2 registers. The figure below shows the selection of boot mode information.

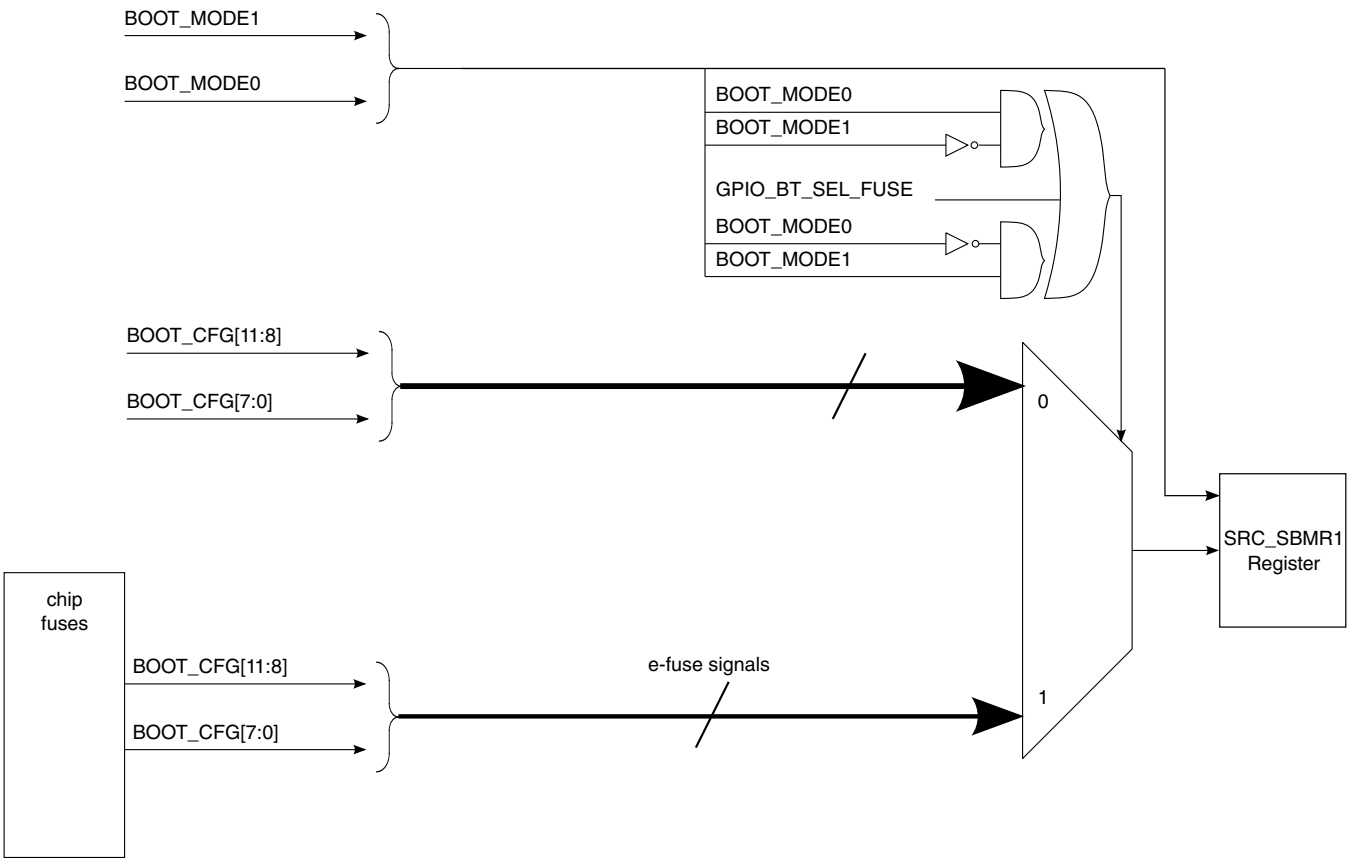


Figure 21-6. Boot mode information

21.3.7 Clocks

The table found here describes the clock sources for SRC.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 21-4. SRC Clocks

Clock name	Clock Root	Description
ipg_clk	ipg_clk_root	Peripheral clock
ipg_clk_s	ipg_clk_root	Peripheral access clock

21.4 External Signals

The following table describes the external signals of SRC.

Table 21-5. SRC External Signals

Signal	Description	Pad	Mode	Direction
SRC_BT_CFG0	Boot configuration signals	GPIO_B0_04	ALT6	I
SRC_BT_CFG1		GPIO_B0_05	ALT6	I
SRC_BT_CFG2		GPIO_B0_06	ALT6	I
SRC_BT_CFG3		GPIO_B0_07	ALT6	I
SRC_BT_CFG4		GPIO_B0_08	ALT6	I
SRC_BT_CFG5		GPIO_B0_09	ALT6	I
SRC_BT_CFG6		GPIO_B0_10	ALT6	I
SRC_BT_CFG7		GPIO_B0_11	ALT6	I
SRC_BT_CFG8		GPIO_B0_12	ALT6	I
SRC_BT_CFG9		GPIO_B0_13	ALT6	I
SRC_BT_CFG10		GPIO_B0_14	ALT6	I
SRC_BT_CFG11		GPIO_B0_15	ALT6	I
SRC_POR_B	Power on reset signal	POR_B	No Muxing (ALT0)	I
SRC_ONOFF	ON/OFF signal	ONOFF	No Muxing (ALT0)	I
SRC_BOOT_MODE0	Boot mode signals	GPIO_AD_B0_04	ALT0	I
SRC_BOOT_MODE1		GPIO_AD_B0_05	ALT0	I

SRC_BT_CFG pins are not used with Teensy because BT_FUSE_SEL is programmed

21.5 Initialization

21.5.1 Power-On Reset and power sequencing

The SRC module generates an internal POR_B signal that is logically AND'ed with any externally applied SRC_POR_B signal. The internal POR_B signal will be held low until all of the following conditions are met:

- 4ms after the external power supply VDDHIGH_IN is valid
- 1ms after the VDD_SOC_IN supply is valid

The 4ms and 1ms delays are derived from counting the 32 kHz RTC clock cycles; the accuracy depends on the accuracy of the RTC. When the RTC crystal is either absent or in the process of powering up, an internal ring oscillator will be the source of RTC, which is not as accurate as the crystal.

21.5.1.1 External POR using SRC_POR_B

If the external SRC_POR_B signal is used to control the processor POR, SRC_POR_B must remain low (asserted) until the VDD_SOC supplies are stable.

21.5.1.2 Internal POR

If the external SRC_POR_B signal is not used (always held high or left unconnected), the processor defaults to the internal POR function (PMU controls generation of the POR based on the power supplies).

21.6 SRC Memory Map/Register Definition

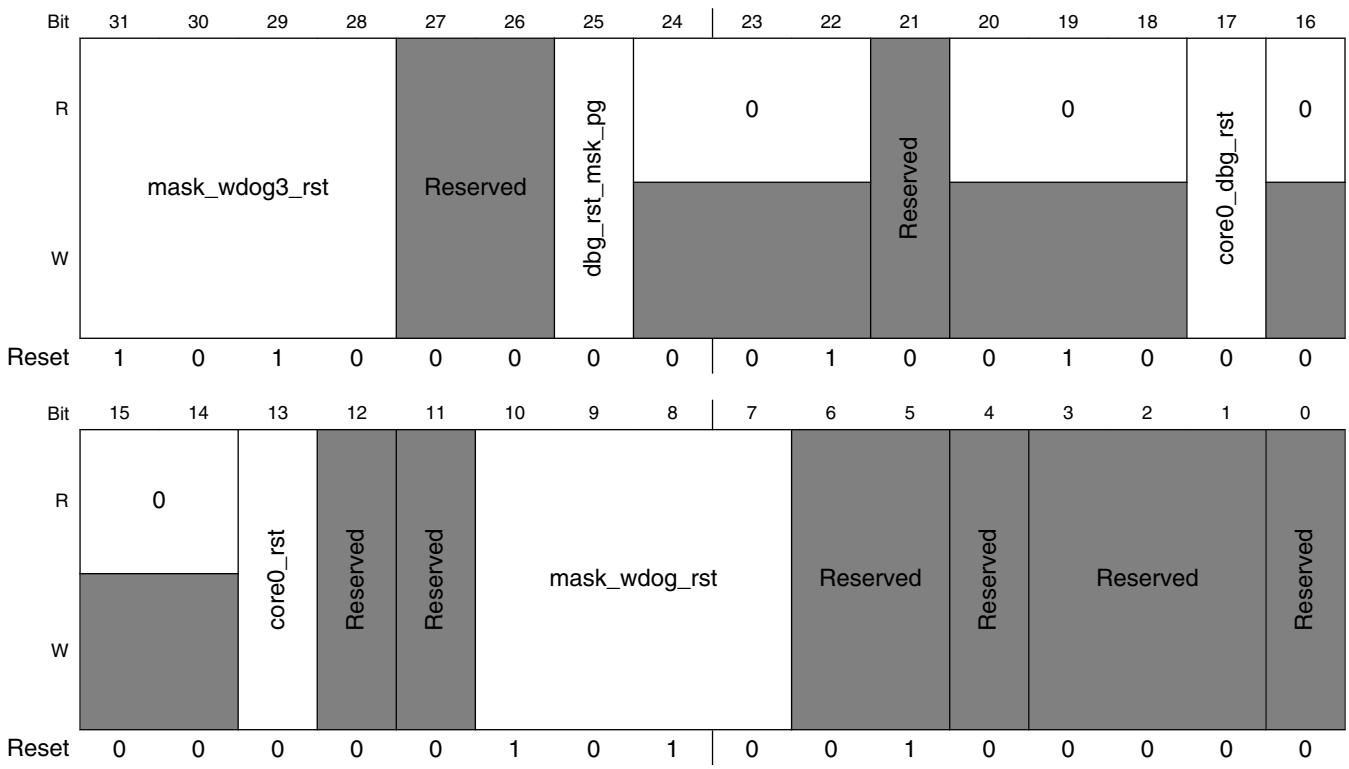
SRC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400F_8000	SRC Control Register (SRC_SCR)	32	R/W	A048_0520h	21.6.1/1274
400F_8004	SRC Boot Mode Register 1 (SRC_SBMR1)	32	R	0000_0000h	21.6.2/1276
400F_8008	SRC Reset Status Register (SRC_SRSR)	32	R/W	0000_0001h	21.6.3/1276
400F_801C	SRC Boot Mode Register 2 (SRC_SBMR2)	32	R	0000_0000h	21.6.4/1279
400F_8020	SRC General Purpose Register 1 (SRC_GPR1)	32	R/W	0000_0000h	21.6.5/1280
400F_8024	SRC General Purpose Register 2 (SRC_GPR2)	32	R/W	0000_0000h	21.6.6/1281
400F_8028	SRC General Purpose Register 3 (SRC_GPR3)	32	R/W	0000_0000h	21.6.7/1281
400F_802C	SRC General Purpose Register 4 (SRC_GPR4)	32	R/W	0000_0000h	21.6.8/1282
400F_8030	SRC General Purpose Register 5 (SRC_GPR5)	32	R/W	0000_0000h	21.6.9/1282
400F_8034	SRC General Purpose Register 6 (SRC_GPR6)	32	R/W	0000_0000h	21.6.10/ 1283
400F_8038	SRC General Purpose Register 7 (SRC_GPR7)	32	R/W	0000_0000h	21.6.11/ 1283
400F_803C	SRC General Purpose Register 8 (SRC_GPR8)	32	R/W	0000_0000h	21.6.12/ 1284
400F_8040	SRC General Purpose Register 9 (SRC_GPR9)	32	R/W	0000_0000h	21.6.13/ 1284
400F_8044	SRC General Purpose Register 10 (SRC_GPR10)	32	R/W	0000_0000h	21.6.14/ 1285

21.6.1 SRC Control Register (SRC_SCR)

The Reset control register (SCR), contains bits that control operation of the reset controller.

Address: 400F_8000h base + 0h offset = 400F_8000h



SRC_SCR field descriptions

Field	Description
31–28 mask_wdog3_rst	Mask wdog3_rst_b source. If these 4 bits are coded from A to 5 then, the wdog3_rst_b input to SRC will be masked and the wdog3_rst_b will not create a reset to the chip. **NOTE:** Any other code than 0101 will be coded to 1010 i.e. wdog3_rst_b is not masked 0101 wdog3_rst_b is masked 1010 wdog3_rst_b is not masked
27–26 -	This field is reserved. Reserved
25 dbg_rst_msk_pg	Do not assert debug resets after power gating event of core 0 do not mask core debug resets (debug resets will be asserted after power gating event) 1 mask core debug resets (debug resets won't be asserted after power gating event)

Table continues on the next page...

SRC_SCR field descriptions (continued)

Field	Description
24–22 Reserved	This read-only field is reserved and always has the value 0.
21 -	This field is reserved. Reserved
20–18 Reserved	This read-only field is reserved and always has the value 0.
17 core0_dbg_rst	Software reset for core0 debug only. **NOTE:** This is a self clearing bit. After it is set to 1, the reset process will begin, and when it finishes, this bit will be self cleared. 0 do not assert core0 debug reset 1 assert core0 debug reset
16–14 Reserved	This read-only field is reserved and always has the value 0.
13 core0_rst	Software reset for core0 only. **NOTE:** This is a self clearing bit. After it is set to 1, the reset process will begin, and when it finishes, this bit will be self cleared. 0 do not assert core0 reset 1 assert core0 reset
12 -	This field is reserved. Reserved
11 -	This field is reserved. Reserved
10–7 mask_wdog_rst	Mask wdog_rst_b source. If these 4 bits are coded from A to 5 then, the wdog_rst_b input to SRC will be masked and the wdog_rst_b will not create a reset to the chip. **NOTE:** During the time the WDOG event is masked using SRC logic, it is likely that the WDOG Reset Status Register (WRSR) bit 1 (which indicates a WDOG timeout event) will get asserted. software / OS developer must prepare for this case. Re-enabling the WDOG is possible, by un-masking it in SRC, though it must be preceded by servicing the WDOG. However, for the case that the event has been asserted, the status bit (WRSR bit-1) will remain asserted, regardless of servicing the WDOG module. (Hardware reset is the only way to cause the de-assertion of that bit). any other code will be coded to 1010 i.e. wdog_rst_b is not masked 0101 wdog_rst_b is masked 1010 wdog_rst_b is not masked (default)
6–5 -	This field is reserved. Reserved
4 -	This field is reserved. Reserved
3–1 -	This field is reserved. Reserved
0 -	This field is reserved. Reserved

21.6.2 SRC Boot Mode Register 1 (SRC_SBMR1)

The Boot Mode register (SBMR) contains bits that reflect the status of Boot Mode Pins of the chip. The reset value is configuration dependent (depending on boot/fuses/IO pads).

If SRC_GPR10[28] bit is set, this bit instructs the ROM code to use the SRC_GPR9 register as if it is SBMR1. This allows software to override the fuse bits and boot from an alternate boot source.

Address: 400F_8000h base + 4h offset = 400F_8004h

Bit	31–24	23–16	15–8	7–0
R	BOOT_CFG4[7:0]	BOOT_CFG3[7:0]	BOOT_CFG2[7:0]	BOOT_CFG1[7:0]
W				
Reset	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0	0 0 0 0 0 0 0 0

SRC_SBMR1 field descriptions

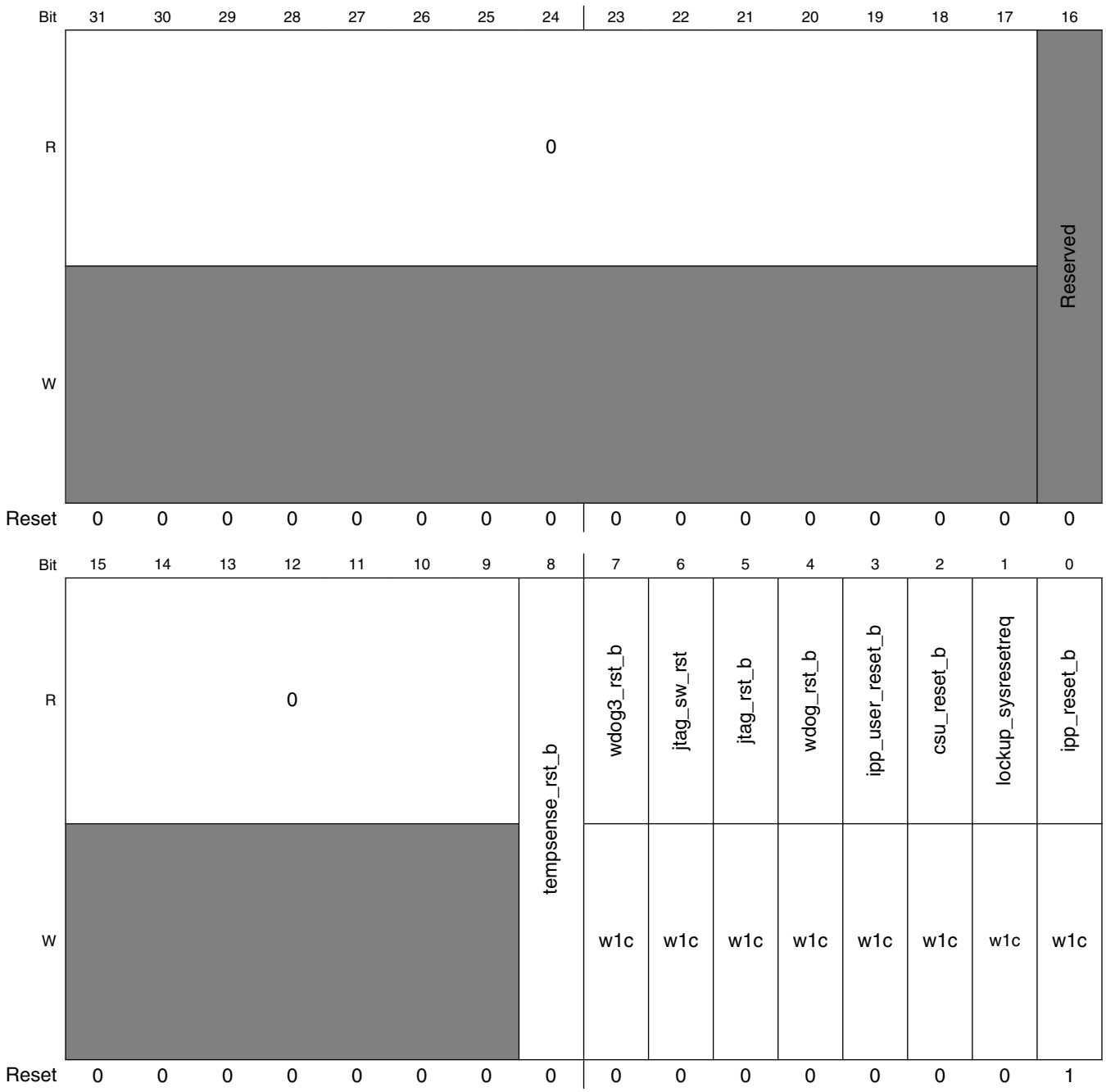
Field	Description
31–24 BOOT_ CFG4[7:0]	Refer to fusemap.
23–16 BOOT_ CFG3[7:0]	Refer to fusemap.
15–8 BOOT_ CFG2[7:0]	Refer to fusemap.
BOOT_ CFG1[7:0]	Refer to fusemap.

21.6.3 SRC Reset Status Register (SRC_SRSR)

The SRSR is a write to one clear register which records the source of the reset events for the chip. The SRC reset status register will capture all the reset sources that have occurred. This register is reset on ipp_reset_b. This is a read-write register.

For bit[6-0] - writing zero does not have any effect. Writing one will clear the corresponding bit. The individual bits can be cleared by writing one to that bit. When the system comes out of reset, this register will have bits set corresponding to all the reset sources that occurred during system reset. Software has to take care to clear this register by writing one after every reset that occurs so that the register will contain the information of recently occurred reset.

Address: 400F_8000h base + 8h offset = 400F_8008h



SRC_SRSR field descriptions

Field	Description
31–17 Reserved	This read-only field is reserved and always has the value 0.
16 -	This field is reserved. Reserved
15–9 Reserved	This read-only field is reserved and always has the value 0.

Table continues on the next page...

SRC_SRSR field descriptions (continued)

Field	Description
8 tempsense_rst_b	Temper Sensor software reset. Indicates whether the reset was the result of software reset from on-chip Temperature Sensor. 0 Reset is not a result of software reset from Temperature Sensor. 1 Reset is a result of software reset from Temperature Sensor.
7 wdog3_rst_b	IC Watchdog3 Time-out reset. Indicates whether the reset was the result of the watchdog3 time-out event. 0 Reset is not a result of the watchdog3 time-out event. 1 Reset is a result of the watchdog3 time-out event.
6 jtag_sw_rst	JTAG software reset. Indicates whether the reset was the result of software reset from JTAG. 0 Reset is not a result of the mentioned case. 1 Reset is a result of the mentioned case.
5 jtag_rst_b	HIGH - Z JTAG reset. Indicates whether the reset was the result of HIGH-Z reset from JTAG. 0 Reset is not a result of HIGH-Z reset from JTAG. 1 Reset is a result of HIGH-Z reset from JTAG.
4 wdog_rst_b	IC Watchdog Time-out reset. Indicates whether the reset was the result of the watchdog time-out event. 0 Reset is not a result of the watchdog time-out event. 1 Reset is a result of the watchdog time-out event.
3 ipp_user_reset_b	Indicates whether the reset was the result of the ipp_user_reset_b qualified reset. 0 Reset is not a result of the ipp_user_reset_b qualified as COLD reset event. 1 Reset is a result of the ipp_user_reset_b qualified as COLD reset event.
2 csu_reset_b	Indicates whether the reset was the result of the csu_reset_b input. 0 Reset is not a result of the csu_reset_b event. 1 Reset is a result of the csu_reset_b event.
1 lockup_sysresetreq	Indicates a reset has been caused by CPU lockup or software setting of SYSRESETREQ bit in Application Interrupt and Reset Control Register of the Arm core. SW needs to write a value to SRC_GPR5 before writing the SYSRESETREQ bit and use the SRC_GPR5 value to distinguish if the reset is caused by SYSRESETREQ or CPU lockup. 0 Reset is not a result of the mentioned case. 1 Reset is a result of the mentioned case.
0 ipp_reset_b	Indicates whether reset was the result of ipp_reset_b pin (Power-up sequence) 0 Reset is not a result of ipp_reset_b pin. 1 Reset is a result of ipp_reset_b pin.

21.6.4 SRC Boot Mode Register 2 (SRC_SBMR2)

The Boot Mode register (SBMR), contains bits that reflect the status of Boot Mode Pins of the chip. The default values for those bits depends on the values of pins/fuses during reset sequence, hence the question mark on their default value.

Address: 400F_8000h base + 1Ch offset = 400F_801Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0						BMOD[1:0]		0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0											BT_FUSE_SEL	-	-	0	SEC_CONFIG[1:0]
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRC_SBMR2 field descriptions

Field	Description
31–26 Reserved	This read-only field is reserved and always has the value 0.
25–24 BMOD[1:0]	BMOD[1:0] shows the latched state of the BOOT_MODE1 and BOOT_MODE0 signals on the rising edge of POR_B. See the Boot mode pin settings section of System Boot.
23–5 Reserved	This read-only field is reserved and always has the value 0.
4 BT_FUSE_SEL	BT_FUSE_SEL (connected to gpio bt_fuse_sel) shows the state of the BT_FUSE_SEL fuse. See Fusemap for additional information on this fuse.
3 -	Reserved
3 -	Reserved
2 Reserved	This read-only field is reserved and always has the value 0.
SEC_ CONFIG[1:0]	SECONFIG[1] shows the state of the SECONFIG[1] fuse. See Fusemap for additional information on this fuse. SECONFIG[0] shows the state of the SECONFIG[0] fuse. This fuse is shown as reserved in Fusemap (address 0x440[1]) because it does not have a user-relevant function.

21.6.5 SRC General Purpose Register 1 (SRC_GPR1)

NOTE

This register is used by the ROM code and should not be used by application software.

Address: 400F_8000h base + 20h offset = 400F_8020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PERSISTENT_ENTRY0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRC_GPR1 field descriptions

Field	Description
PERSISTENT_ ENTRY0	Holds entry function for core0 for waking-up from low power mode. The SRC ensures that the register value will persist across system resets.

21.6.6 SRC General Purpose Register 2 (SRC_GPR2)

NOTE

This register is used by the ROM code and should not be used by application software.

Address: 400F_8000h base + 24h offset = 400F_8024h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PERSISTENT_ARG0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRC_GPR2 field descriptions

Field	Description
PERSISTENT_ARG0	Holds argument of entry function for core0 for waking-up from low power mode. The SRC ensures that the register value will persist across system resets.

21.6.7 SRC General Purpose Register 3 (SRC_GPR3)

NOTE

This register is used by the ROM code and should not be used by application software.

Address: 400F_8000h base + 28h offset = 400F_8028h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRC_GPR3 field descriptions

Field	Description
-	Read/write general purpose bits used to store an arbitrary value.

21.6.8 SRC General Purpose Register 4 (SRC_GPR4)

NOTE

This register is used by the ROM code and should not be used by application software.

Address: 400F_8000h base + 2Ch offset = 400F_802Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRC_GPR4 field descriptions

Field	Description
-	Read/write general purpose bits used to store an arbitrary value.

21.6.9 SRC General Purpose Register 5 (SRC_GPR5)

NOTE

SW needs to write a value to SRC_GPR5 before writing the SYSRESETREQ bit and use the SRC_GPR5 value to distinguish if the reset is caused by SYSRESETREQ or CPU lockup

Address: 400F_8000h base + 30h offset = 400F_8030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRC_GPR5 field descriptions

Field	Description
-	Read/write general purpose bits used to store an arbitrary value.

21.6.10 SRC General Purpose Register 6 (SRC_GPR6)

NOTE

This register is used by the ROM code and should not be used by application software.

Address: 400F_8000h base + 34h offset = 400F_8034h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRC_GPR6 field descriptions

Field	Description
-	Read/write general purpose bits used to store an arbitrary value.

21.6.11 SRC General Purpose Register 7 (SRC_GPR7)

NOTE

This register is used by the ROM code and should not be used by application software.

Address: 400F_8000h base + 38h offset = 400F_8038h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRC_GPR7 field descriptions

Field	Description
-	Read/write general purpose bits used to store an arbitrary value.

21.6.12 SRC General Purpose Register 8 (SRC_GPR8)

NOTE

This register is used by the ROM code and should not be used by application software.

Address: 400F_8000h base + 3Ch offset = 400F_803Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	-																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRC_GPR8 field descriptions

Field	Description
-	Read/write general purpose bits used to store an arbitrary value.

21.6.13 SRC General Purpose Register 9 (SRC_GPR9)

NOTE

This register is used by the ROM code and should not be used by application software.

Address: 400F_8000h base + 40h offset = 400F_8040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

SRC_GPR9 field descriptions

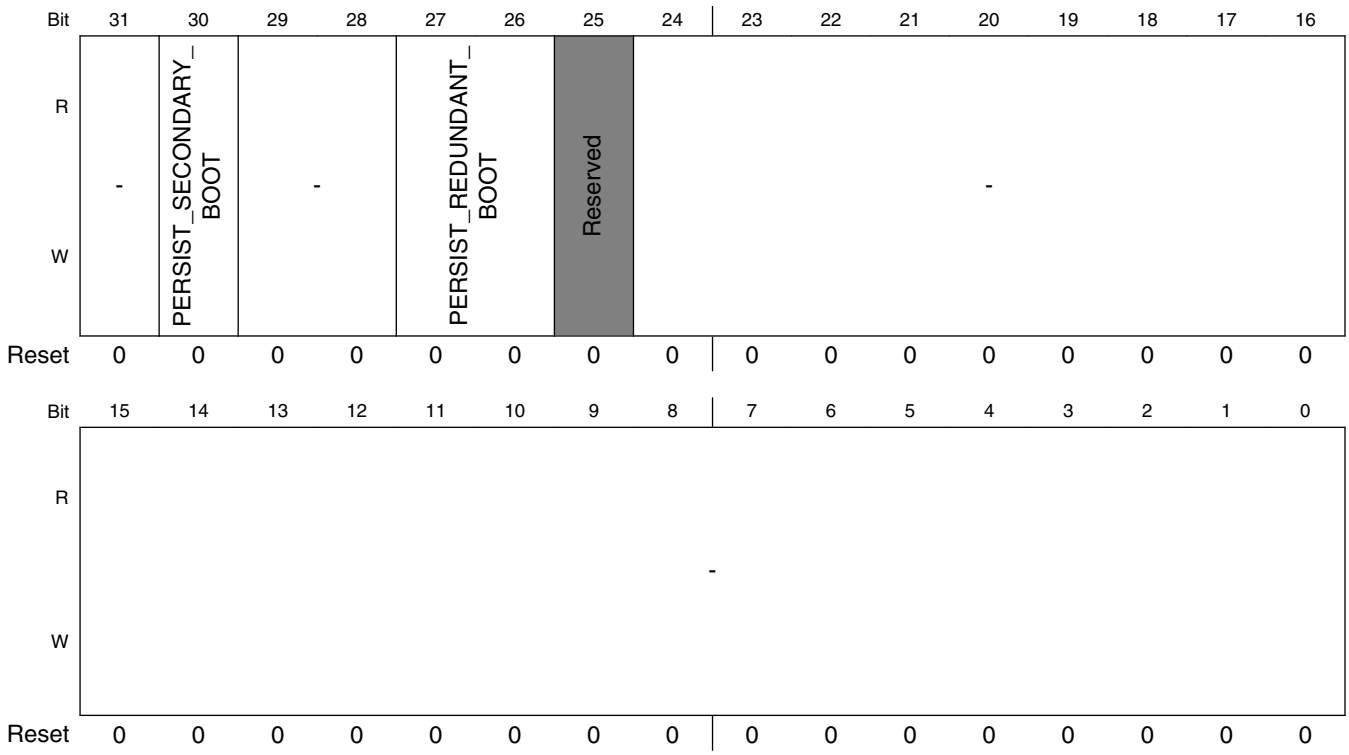
Field	Description
-	This field is reserved. Reserved.

21.6.14 SRC General Purpose Register 10 (SRC_GPR10)

NOTE

This register is used by the ROM code and should not be used by application software.

Address: 400F_8000h base + 44h offset = 400F_8044h



SRC_GPR10 field descriptions

Field	Description
31 -	Read/write bit, for general purpose **NOTE:** Reset only by POR
30 PERSIST_ SECONDARY_ BOOT	This bit identifies which image must be used - primary and secondary. Used only for eMMC/SD/FlexSPI NOR boot.
29–28 -	Read/write bits, for general purpose **NOTE:** Reset only by POR
27–26 PERSIST_ REDUNDANT_ BOOT	This field identifies which image must be used - 0/1/2/3. Used for both SPI NAND and SLC raw NAND devices.

Table continues on the next page...

SRC_GPR10 field descriptions (continued)

Field	Description
25 -	This field is reserved. Reserved.
-	Read/write bits, for general purpose **NOTE:** Reset only by POR

Chapter 22
Fusemap

22.1 Boot Fusemap

This section describes the various modes and the selection of the required boot devices.

A separate map is given for each and every boot device. The device select is specified by the BOOT_CFG1[7:4] fuses.

Table 22-1. Boot device select

Boot Device	BOOT_CFG1[7]	BOOT_CFG1[6]	BOOT_CFG1[5]	BOOT_CFG1[4]
FlexSPI1 (Serial NOR)	0	0	0	0
SD	0	1	x	x
MMC/eMMC	1	0	x	x
SEMC (NAND)	0	0	1	x
SEMC (NOR)	0	0	0	1
FlexSPI1 (Serial NAND)	1	1	x	x

NOTE

The fuses marked as “Reserved” are reserved for NXP internal (and future) use only, or are security related. Customers **must NOT** attempt to burn these, because the IC behavior may be unpredictable. The reserved fuses can be read as either '0' or '1'.

Security-related fuses are described in the Security Reference Manual (SRM) for this device.

Table 22-2. FlexSPI (Serial NOR) boot fusemap

Address	7	6	5	4	3	2	1	0
0x450[7:0] (BOOT_CFG1)	0	0	0	0	xSPI FLASH Auto Probe Type 00 - QuadSPI NOR		EncryptedXIP 0 - Disabled	xSPI FLASH Auto Probe

Table continues on the next page...

Table 22-2. FlexSPI (Serial NOR) boot fusemap (continued)

Address	7	6	5	4	3	2	1	0
					01 - MXIC Octal 10 - Micron Octal 11 - Adesto Octal		1 - Enabled	0 - Disabled 1 - Enabled
0x450[15:8] (BOOT_CFG2)	Reserved	Reserved	Reserved	Reserved	Reserved	FLASH TYPE: 000 - Device supports 3B read by default 001 - Device supports 4B read by default 010 - HyperFlash 1V8 011 - HyperFlash 3V3 100 - MXIC Octal DDR 101 - Micron Octal DDR 111 - QSPI device supports 3B read by default (on secondary pinmux option)		
0x450[23:16] (BOOT_CFG3)	Reserved							
0x450[31:24] (BOOT_CFG4)	Reserved							
0x460[7:0]	Reserved		FORCE_COLD_BOOT(SBMR)	BT_FUSE_SEL	Reserved	BOOT_FREQ (ARM/BUS) 0 - 396/132 MHz 1 - 528/132 MHz	Reserved	Reserved
0x460[15:8]	Reserved		Reserved		Reserved (SDR config)			
0x460[23:16]	Reserved		WDOG_ENABLE '0' - Disabled '1' - Enabled	Reserved>	DAP_SJC_SWD_SEL	SDP_READ_DISABLE	SDP_DISABLE	FORCE_INTERNAL_BOOT
0x460[31:24]	SD_PWR_CYCLE_SELECTION: '00' - 20ms '01' - 10ms '10' - 5ms '11' - 2.5ms		PWR_STABLE_CYCLE_SELECTION: '0' - 5ms '1' - 2.5ms	Reserved	Reserved	Reserved	NAND_ECC_DISABLE 0 - NAND ECC is enabled 0 - NAND ECC is disabled	DLL_ENABLE 0 - Disable DLL for SD/eMMC 1 - Enable DLL for SD/Emmc
0x470[7:0]	DLL Override:	SD1_RST_POLARITY_SELECT:	SD2 VOLTAGE SELECTION	UART Serial Download Disable:	Disable SDMMC Manufacture mode:	L1 I-Cache DISABLE	BT_MPU_DISABLE	Override SD Pad Settings

Table continues on the next page...

Table 22-2. FlexSPI (Serial NOR) boot fusemap (continued)

Address	7	6	5	4	3	2	1	0
	0 - DLL Slave Mode for SD/ eMMC 1 - DLL Override Mode for SD/eMMC	0 - Reset active low 1 - Reset active high	0 - 3.3V 1 - 1.8V	'0' - Not Disable '1' - Disable	0 - Enable 1 - Disable		0 - MPU is enabled during boot 1 - MPU is disabled during boot	(using PAD_SETTINGS value)
0x470[15:8]	SD2_RST_POLARITY_SELECT: 0 - Reset active low 1 - Reset active high	eMMC 4.4 - RESET TO PRE-IDLE STATE	Override HYS bit for SD/MMC pads	USDHC_PAD_PULL_DOWN: 0 - no action 1 - pull down	ENABLE_EMMC_22K_PULLUP: 0 - 47K pullup 1 - 22K pullup	Boot Failure Indicator Pin Select[4]	USDHC_IOMUX_SION_BIT_ENABLE: 0 - Disable 1 - Enable	USDHC IOMUX SRE Enable: 0 - Disable 1 - Enable
0x470[23:16]	Reserved	LPB_BOOT: (Core / Bus) '00' - Div by 1 '01' - Div by 2 '10' - Div by 4 '11' - Div by 8		Reserved	Boot Failure Indicator Pin Select[3:0] 00000 - gpio1.IO00 00001 - gpio1.IO01 ... 11111 - gpio1.IO31 Note: Please refer to the Reference Manual for pins which gpio1.IOxx will be muxed to by default			
0x470[31:24]	Override NAND Pad Settings (using PAD_SETTINGS value)	MMC_DLL_DLY[6:0]: Delay target for SD/eMMC DLL, it is applied to slave mode target delay or override mode target delay depends on DLL Override fuse bit value. DELAY_CELL_NUM (FlexSPI NOR): "000" - DLL Override feature is disabled "001-111" - DLL Override feature is enabled See OVRDVAL in FLEXSPI chapter in the Reference Manual for more details						

Table 22-3. SD boot fusemap Not used with Teensy

Address	7	6	5	4	3	2	1	0
0x450[7:0] (BOOT_CFG1)	0	1	SD/SDXC Speed: 00 - Normal/SDR12 01 - High/SDR25 10 - SDR50 11 - SDR104		SD Power Cycle Enable: '0' - No power cycle '1' - Enabled via USDHC_RST pad	SD Loop back Clock Source Select: (for SDR50 and SDR104 only) '0' - through SD pad '1' - direct	Port Select: 0 - eSDHC1 1 - eSDHC2	Fast Boot: 0 - Regular 1 - Fast Boot
0x450[15:8] (BOOT_CFG2)	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Bus Width: 0 - 1-bit	SD1 Voltage Selection:

Table continues on the next page...

Table 22-3. SD boot fusemap (continued) Not used with Teensy

Address	7	6	5	4	3	2	1	0
							1 - 4-bit	0 - 3.3V 1 - 1.8V
0x450[23:16] (BOOT_CFG3)	Reserved							
0x450[31:24] (BOOT_CFG4)	Reserved							
0x460[7:0]	Reserved		FORCE_COLD_BOOT(SBMR)	BT_FUSE_SEL	Reserved	BOOT_FREQ (ARM/ BUS) 0 - 396/132 MHz 1 - 528/132 MHz	Reserved	Reserved
0x460[15:8]	Reserved		Reserved		Reserved (SDR config)			
0x460[23:16]	Reserved		WDOG_ENABLE '0' - Disabled '1' - Enabled	Reserved	DAP_SJC_SWD_SEL	SDP_READ_DISABLE	SDP_DISABLE	FORCE_INTERNAL_BOOT
0x460[31:24]	SD_PWR_CYCLE_SELECTION: '00' - 20ms '01' - 10ms '10' - 5ms '11' - 2.5ms		PWR_STABLE_CYCLE_SELECTION: '0' - 5ms '1' - 2.5ms	Reserved	Reserved	Reserved	NAND_ECC_DISABLE 0 - NAND ECC is enabled 0 - NAND ECC is disabled	DLL_ENABLE 0 - Disable DLL for SD/ eMMC 1 - Enable DLL for SD/ Emmc
0x470[7:0]	DLL Override: 0 - DLL Slave Mode for SD/ eMMC 1 - DLL Override Mode for SD/eMMC	SD1_RST_POLARITY_SELECT: 0 - Reset active low 1 - Reset active high	SD2 VOLTAGE SELECTION 0 - 3.3V 1 - 1.8V	UART Serial Download Disable: '0' - Not Disable '1' - Disable	Disable SDMMC Manufacture mode: 0 - Enable 1 - Disable	L1 I-Cache DISABLE	BT_MPU_DISABLE 0 - MPU is enabled during boot 1 - MPU is disabled during boot	Override Pad Settings (using PAD_SETTINGS value)
0x470[15:8]	SD2_RST_POLARITY_SELECT: 0 - Reset active low 1 - Reset active high	eMMC 4.4 - RESET TO PRE-IDLE STATE	Override HYS bit for SD/MMC pads	USDHC_PAD_PULL_DOWN: 0 - no action 1 - pull down	ENABLE_EMMC_22K_PULLUP: 0 - 47K pullup 1 - 22K pullup	Boot Failure Indicator Pin Select[4]	USDHC_IOMUX_SION_BIT_ENABLE: 0 - Disable 1 - Enable	USDHC IOMUX SRE Enable: 0 - Disable 1 - Enable
0x470[23:16]	Reserved	LPB_BOOT: (Core / Bus)		Reserved	Boot Failure Indicator Pin Select[3:0]			

Table continues on the next page...

Table 22-3. SD boot fusemap (continued) Not used with Teensy

Address	7	6	5	4	3	2	1	0
		'00' - Div by 1 '01' - Div by 2 '10' - Div by 4 '11' - Div by 8			00000 - gpio1.IO00 00001 - gpio1.IO01 ... 11111 - gpio1.IO31 Note: Please refer RM for pins which gpio1.IOxx will be muxed to by default			
0x470[31:24]	Override NAND Pad Settings (using PAD_SETTINGS value)	MMC_DLL_DLY[6:0]: Delay target for SD/eMMC DLL, it is applied to slave mode target delay or override mode target delay depends on DLL Override fuse bit value. DELAY_CELL_NUM (FlexSPI NOR): "000" - DLL Override feature is disabled "001-111" - DLL Override feature is enabled, See OVRDVAL in FLEXSPI chapter in RM for more details						

Table 22-4. MMC/eMMC boot fusemap Not used with Teensy

Address	7	6	5	4	3	2	1	0
0x450[7:0] (BOOT_CFG1)	1	0	SD/MMC Speed: 0 - Normal 1 - High	Fast Boot Acknowledge Enable: 0 - Boot Ack Disabled 1 - Boot Ack Enabled	SD Power Cycle Enable: '0' - No power cycle '1' - Enabled via USDHC_RST pad	SD Loop back Clock Source Select: (for SDR50 and SDR104 only) '0' - through SD pad '1' - direct	Port Select: 0 - eSDHC1 1 - eSDHC2	Fast Boot: 0 - Regular 1 - Fast Boot
0x450[15:8] (BOOT_CFG2)	Reserved	Reserved	Reserved	Reserved	Reserved	Bus Width: 01 - 4-bit 01 - 8-bit 10 - 4-bit DDR(MMC 4.4) 11 - 8-bit DDR(MMC 4.4)		SD1 Voltage Selection: 0 - 3.3V 1 - 1.8V
0x450[23:16] (BOOT_CFG3)	Reserved							
0x450[31:24] (BOOT_CFG4)	Reserved							
0x460[7:0]	Reserved		FORCE_COLD_BOOT(SBMR)	BT_FUSE_SEL	Reserved	BOOT_FREQ (ARM/BUS) 0 - 396/132 MHz	Reserved	Reserved

Table continues on the next page...

Table 22-4. MMC/eMMC boot fusemap (continued) Not used with Teensy

Address	7	6	5	4	3	2	1	0
						1 - 528/132 MHz		
0x460[15:8]	Reserved		Reserved		Reserved (SDR config)			
0x460[23:16]	Reserved		WDOG_ENABLE '0' - Disabled '1' - Enabled	Reserved	DAP_SJC_SWD_SEL	SDP_READ_DISABLE	SDP_DISABLE	FORCE_INTERNAL_BOOT
0x460[31:24]	SD_PWR_CYCLE_SELECTION: '00' - 20ms '01' - 10ms '10' - 5ms '11' - 2.5ms		PWR_STABLE_CYCLE_SELECTION: '0' - 5ms '1' - 2.5ms	Reserved	Reserved	Reserved	NAND_ECC_DISABLE 0 - NAND ECC is enabled 0 - NAND ECC is disabled	DLL_ENABLE 0 - Disable DLL for SD/eMMC 1 - Enable DLL for SD/Emmc
0x470[7:0]	DLL Override: 0 - DLL Slave Mode for SD/eMMC 1 - DLL Override Mode for SD/eMMC	SD1_RST_POLARITY_SELECT: 0 - Reset active low 1 - Reset active high	SD2 VOLTAGE SELECTION 0 - 3.3V 1 - 1.8V	UART Serial Download Disable: '0' - Not Disable '1' - Disable	Disable SDMMC Manufacture mode: 0 - Enable 1 - Disable	L1 I-Cache DISABLE	BT_MPU_DISABLE 0 - MPU is enabled during boot 1 - MPU is disabled during boot	Override Pad Settings (using PAD_SETTINGS value)
0x470[15:8]	SD2_RST_POLARITY_SELECT: 0 - Reset active low 1 - Reset active high	eMMC 4.4 - RESET TO PRE-IDLE STATE	Override HYS bit for SD/MMC pads	USDHC_PAD_PULL_DOWN: 0 - no action 1 - pull down	ENABLE_EMMC_22K_PULLUP: 0 - 47K pullup 1 - 22K pullup	Boot Failure Indicator Pin Select[4]	USDHC_IOMUX_SION_BIT_ENABLE: 0 - Disable 1 - Enable	USDHC IOMUX SRE Enable: 0 - Disable 1 - Enable
0x470[23:16]	Reserved	LPB_BOOT: (Core / Bus) '00' - Div by 1 '01' - Div by 2 '10' - Div by 4 '11' - Div by 8		Reserved	Boot Failure Indicator Pin Select[3:0] 00000 - gpio1.IO00 00001 - gpio1.IO01 ... 11111 - gpio1.IO31 Note: Please refer RM for pins which gpio1.IOxx will be muxed to by default			
0x470[31:24]	Override NAND Pad Settings (using PAD_SETTINGS value)	MMC_DLL_DLY[6:0]: Delay target for SD/eMMC DLL, it is applied to slave mode target delay or override mode target delay depends on DLL Override fuse bit value. DELAY_CELL_NUM (FlexSPI NOR): "000" - DLL Override feature is disabled						

Table 22-4. MMC/eMMC boot fusemap Not used with Teensy

Address	7	6	5	4	3	2	1	0
		"001-111" - DLL Override feature is enabled, See OVRDVAL in FLEXSPI chapter in RM for more details						

Table 22-5. SEMC (NAND) boot fusemap Not used with Teensy

Address	7	6	5	4	3	2	1	0
0x450[7:0] (BOOT_CFG1)	0	0	1	BOOT_SEARCH_STRIDE Search stride for FCB and DBBT Search strides in terms of page 0000 - 64 Other value - 2^(BOOT_SEACH_STRIDE)				BOOT_SEARCH_COUNT 0 - 1 1 - 2
0x450[15:8] (BOOT_CFG2)	Reserved	Reserved	Reserved	Reserved	Reserved	DQS disable: 0 - disabled; 1 - enabled	ECC_ALG_SEL Disable: 0 - SW ECC selected; 1 - Device ECC selected	ONFI compliant: 0 - Yes, ONFI 1 - No, spec
0x450[23:16] (BOOT_CFG3)	Reserved							
0x450[31:24] (BOOT_CFG4)	Reserved							
0x460[7:0]	Reserved		FORCE_COLD_BOOT(SBMR)	BT_FUSE_SEL	Reserved	BOOT_FREQ (ARM/ BUS) 0 - 396/132 MHz 1 - 528/132 MHz	Reserved	Reserved
0x460[15:8]	Reserved		Reserved		Reserved (SDR config)			
0x460[23:16]	Reserved		WDOG_ENABLE '0' - Disabled '1' - Enabled	Reserved	DAP_SJC_SWD_SEL	SDP_READ_DISABLE	SDP_DISABLE	FORCE_INTERNAL_BOOT
0x460[31:24]	SD_PWR_CYCLE_SELECTION: '00' - 20ms '01' - 10ms '10' - 5ms '11' - 2.5ms		PWR_STABLE_CYCLE_SELECTION: '0' - 5ms '1' - 2.5ms	Reserved	Reserved	Reserved	NAND_ECC_DISABLE 0 - NAND ECC is enabled 0 - NAND ECC is disabled	DLL_ENABLE 0 - Disable DLL for SD/ eMMC 1 - Enable DLL for SD/ Emmc

Table continues on the next page...

Table 22-5. SEMC (NAND) boot fusemap (continued) Not used with Teensy

Address	7	6	5	4	3	2	1	0
0x470[7:0]	DLL Override: 0 - DLL Slave Mode for SD/ eMMC 1 - DLL Override Mode for SD/eMMC	SD1_RST_ POLARITY_ SELECT: 0 - Reset active low 1 - Reset active high	SD2 VOLTAGE SELECTIO N 0 - 3.3V 1 - 1.8V	UART Serial Download Disable: '0' - Not Disable '1' - Disable	Disable SDMMC Manufactur e mode: 0 - Enable 1 - Disable	L1 I-Cache DISABLE	BT_MPU_D ISABLE 0 - MPU is enabled during boot 1 - MPU is disabled during boot	Override Pad Settings (using PAD_SETTI NGS value)
0x470[15:8]	SD2_RST_ POLARITY_ SELECT: 0 - Reset active low 1 - Reset active high	eMMC 4.4 - RESET TO PRE-IDLE STATE	Override HYS bit for SD/MMC pads	USDHC_PA D_PULL_D OWN: 0 - no action 1 - pull down	ENABLE_E MMC_22K_ PULLUP: 0 - 47K pullup 1 - 22K pullup	Boot Failure Indicator Pin Select[4]	USDHC_IO MUX_SION _BIT_ENAB LE: 0 - Disable 1 - Enable	USDHC IOMUX SRE Enable: 0 - Disable 1 - Enable
0x470[23:16]	Reserved	LPB_BOOT: (Core / Bus) '00' - Div by 1 '01' - Div by 2 '10' - Div by 4 '11' - Div by 8		Reserved	Boot Failure Indicator Pin Select[3:0] 00000 - gpio1.IO00 00001 - gpio1.IO01 ... 11111 - gpio1.IO31 Note: Please refer RM for pins which gpio1.IOxx will be muxed to by default			
0x470[31:24]	Override NAND Pad Settings (using PAD_SETTI NGS value)	MMC_DLL_DLY[6:0]: Delay target for SD/eMMC DLL, it is applied to slave mode target delay or override mode target delay depends on DLL Override fuse bit value. DELAY_CELL_NUM (FlexSPI NOR): "000" - DLL Override feature is disabled "001-111" - DLL Override feature is enabled, See OVRDVAL in FLEXSPI chapter in RM for more details						

Table 22-6. SEMC (NOR) boot fusemap Not used with Teensy

Address	7	6	5	4	3	2	1	0
0x450[7:0] (BOOT_CFG1)	0	0	0	1	Reserved	Clock Freq: 000 - 33MHz 001 - 66MHz 010 - 108MHz 011 - 133MHz 1xx - 166MHz		

Table continues on the next page...

Table 22-6. SEMC (NOR) boot fusemap (continued) Not used with Teensy

Address	7	6	5	4	3	2	1	0
0x450[15:8] (BOOT_CFG2)	Reserved	Reserved	Reserved	Reserved	Reserved	DQS Disable: 0 - disabled 1 - enabled	Data Port Size: 0 - 16 bit 1 - 8 bit	AC Timing 0 - Default 1 - Fuse (0x6E0)
0x450[23:16] (BOOT_CFG3)	Reserved							
0x450[31:24] (BOOT_CFG4)	Reserved							
0x460[7:0]	Reserved		FORCE_COLD_BOOT(SBMR)	BT_FUSE_SEL	Reserved	BOOT_FREQ (ARM/ BUS) 0 - 396/132 MHz 1 - 528/132 MHz	Reserved	Reserved
0x460[15:8]	Reserved		Reserved		Reserved (SDR config)			
0x460[23:16]	Reserved		WDOG_ENABLE '0' - Disabled '1' - Enabled	Reserved	DAP_SJC_SWD_SEL	SDP_READ_DISABLE	SDP_DISABLE	FORCE_INTERNAL_BOOT
0x460[31:24]	SD_PWR_CYCLE_SELECTION: '00' - 20ms '01' - 10ms '10' - 5ms '11' - 2.5ms		PWR_STABLE_CYCLE_SELECTION: '0' - 5ms '1' - 2.5ms	Reserved	Reserved	Reserved	NAND_ECC_DISABLE 0 - NAND ECC is enabled 0 - NAND ECC is disabled	DLL_ENABLE 0 - Disable DLL for SD/ eMMC 1 - Enable DLL for SD/ Emmc
0x470[7:0]	DLL Override: 0 - DLL Slave Mode for SD/ eMMC 1 - DLL Override Mode for SD/eMMC	SD1_RST_POLARITY_SELECT: 0 - Reset active low 1 - Reset active high	SD2 VOLTAGE SELECTION 0 - 3.3V 1 - 1.8V	UART Serial Download Disable: '0' - Not Disable '1' - Disable	Disable SDMMC Manufacture mode: 0 - Enable 1 - Disable	L1 I-Cache DISABLE	BT_MPU_DISABLE 0 - MPU is enabled during boot 1 - MPU is disabled during boot	Override Pad Settings (using PAD_SETTINGS value)
0x470[15:8]	SD2_RST_POLARITY_SELECT: 0 - Reset active low	eMMC 4.4 - RESET TO PRE-IDLE STATE	Override HYS bit for SD/MMC pads	USDHC_PAD_PULL_DOWN: 0 - no action	ENABLE_EMMC_22K_PULLUP: 0 - 47K pullup	Boot Failure Indicator Pin Select[4]	USDHC_IOMUX_SION_BIT_ENABLE: 0 - Disable 1 - Enable	USDHC IOMUX SRE Enable: 0 - Disable 1 - Enable

Table continues on the next page...

Table 22-6. SEMC (NOR) boot fusemap (continued) Not used with Teensy

Address	7	6	5	4	3	2	1	0
	1 - Reset active high			1 - pull down	1 - 22K pullup			
0x470[23:16]	Reserved	LPB_BOOT: (Core / Bus) '00' - Div by 1 '01' - Div by 2 '10' - Div by 4 '11' - Div by 8		Reserved	Boot Failure Indicator Pin Select[3:0] 00000 - gpio1.IO00 00001 - gpio1.IO01 ... 11111 - gpio1.IO31 Note: Please refer RM for pins which gpio1.IOxx will be muxed to by default			
0x470[31:24]	Override NAND Pad Settings (using PAD_SETTINGS value)	MMC_DLL_DLY[6:0]: Delay target for SD/eMMC DLL, it is applied to slave mode target delay or override mode target delay depends on DLL Override fuse bit value. DELAY_CELL_NUM (FlexSPI NOR): "000" - DLL Override feature is disabled "001-111" - DLL Override feature is enabled, See OVRDVAL in FLEXSPI chapter in RM for more details						

Table 22-7. FlexSPI1 (Serial NAND) boot fusemap Not used with Teensy

Address	7	6	5	4	3	2	1	0
0x450[7:0] (BOOT_CFG1)	1	1	SAFE FREQ: (Default safe communication frequency) 0 – High Speed (50MHz) 1 – Low Speed (30MHZ)	COL_ADDRESS_WIDTH: 0 – 12bits 1 – 13bits	SPI NAND HOLD TIME: 00 - 0us 01 - 500us 10 - 1ms 11 - 3ms		BOOT_SEARCH_STRIDE: Search stride for FCB and DBBT (in terms of page) 0 - 64 1 - 128 2 - 256 3- -32	
0x450[15:8] (BOOT_CFG2)	Reserved	Reserved	Reserved	Reserved	Reserved	CS_INTERVAL: CS de-asserted interval between two commands 0 – 100ns 1 – 200ns 2 – 400ns 3 – 50ns		BOOT_SEARCH_COUNT: 0 - 1 1 - 2
0x450[23:16] (BOOT_CFG3)	Reserved							
0x450[31:24]	Reserved							

Table continues on the next page...

Table 22-7. FlexSPI1 (Serial NAND) boot fusemap (continued) Not used with Teensy

Address	7	6	5	4	3	2	1	0
(BOOT_CFG4)								
0x460[7:0]	Reserved		FORCE_COLD_BOOT(SBMR)	BT_FUSE_SEL	Reserved	BOOT_FREQ (ARM/ BUS) 0 - 396/132 MHz 1 - 528/132 MHz	Reserved	Reserved
0x460[15:8]	Reserved		Reserved		Reserved (SDR config)			
0x460[23:16]	Reserved		WDOG_ENABLE '0' - Disabled '1' - Enabled	Reserved	DAP_SJC_SWD_SEL	SDP_READ_DISABLE	SDP_DISABLE	FORCE_INTERNAL_BOOT
0x460[31:24]	SD_PWR_CYCLE_SELECTION: '00' - 20ms '01' - 10ms '10' - 5ms '11' - 2.5ms		PWR_STABLE_CYCLE_SELECTION: '0' - 5ms '1' - 2.5ms	Reserved	Reserved	Reserved	NAND_ECC_DISABLE 0 - NAND ECC is enabled 0 - NAND ECC is disabled	DLL_ENABLE 0 - Disable DLL for SD/ eMMC 1 - Enable DLL for SD/ Emmc
0x470[7:0]	DLL Override: 0 - DLL Slave Mode for SD/ eMMC 1 - DLL Override Mode for SD/eMMC	SD1_RST_POLARITY_SELECT: 0 - Reset active low 1 - Reset active high	SD2 VOLTAGE SELECTION 0 - 3.3V 1 - 1.8V	UART Serial Download Disable: '0' - Not Disable '1' - Disable	Disable SDMMC Manufacture mode: 0 - Enable 1 - Disable	L1 I-Cache DISABLE	BT_MPU_DISABLE 0 - MPU is enabled during boot 1 - MPU is disabled during boot	Override Pad Settings (using PAD_SETTINGS value)
0x470[15:8]	SD2_RST_POLARITY_SELECT: 0 - Reset active low 1 - Reset active high	eMMC 4.4 - RESET TO PRE-IDLE STATE	Override HYS bit for SD/MMC pads	USDHC_PAD_PULL_DOWN: 0 - no action 1 - pull down	ENABLE_EMMC_22K_PULLUP: 0 - 47K pullup 1 - 22K pullup	Boot Failure Indicator Pin Select[4]	USDHC_IOMUX_SION_BIT_ENABLE: 0 - Disable 1 - Enable	USDHC IOMUX SRE Enable: 0 - Disable 1 - Enable
0x470[23:16]	Reserved	LPB_BOOT: (Core / Bus) '00' - Div by 1 '01' - Div by 2 '10' - Div by 4 '11' - Div by 8		Reserved	Boot Failure Indicator Pin Select[3:0] 00000 - gpio1.IO00 00001 - gpio1.IO01 ... 11111 - gpio1.IO31			

Table continues on the next page...

Table 22-7. FlexSPI1 (Serial NAND) boot fusemap (continued) Not used with Teensy

Address	7	6	5	4	3	2	1	0
					Note: Please refer RM for pins which gpio1.IOxx will be muxed to by default			
0x470[31:24]	Override NAND Pad Settings (using PAD_SETTINGS value)	MMC_DLL_DLY[6:0]: Delay target for SD/eMMC DLL, it is applied to slave mode target delay or override mode target delay depends on DLL Override fuse bit value. DELAY_CELL_NUM (FlexSPI NOR): "000" - DLL Override feature is disabled "001-111" - DLL Override feature is enabled, See OVRDVAL in FLEXSPI chapter in RM for more details						

22.2 Lock Fusemap

Table 22-8. Lock fuses

Address	7	6	5	4	3	2	1	0
0x400[7:0]	Reserved	SJC_RESP_LOCK 0- Unlock '1' - WP,OP,RP	Reserved		BOOT_CFG_LOCK 00- Unlock '1x' - OP 'x1' - WP		Reserved	
0x400[15:8]	Reserved		GP2_LOCK 00- Unlock '1x' - OP 'x1' - WP		GP1_LOCK 00- Unlock '1x' - OP 'x1' - WP		MAC_ADDR_LOCK 00- Unlock '1x' - OP 'x1' - WP	
0x400[23:16]	Reserved	MISC_CONF_LOCK '1' - WP + OP	Reserved		ANALOG_LOCK 00- Unlock '1x' - OP 'x1' - WP		Reserved	SW_GP1_LOCK 0- Unlock '1' - WP + OP of SW_GP1 fuses
0x400[31:24]	Reserved				GP3_LOCK '1x' - OP 'x1' - WP		Reserved	

NOTE

Do NOT program *Reserved* bits that are not security related (mentioned in the SRM).

22.3 Fusemap Descriptions Table

This section describes the chip fusemap descriptions.

Table 22-9. Fusemap Descriptions

Fuse Address	Fuses Name	Number of Fuses	Fuses Function	Setting	Locked by
0x400[1:0]	Reserved	2	Reserved	Reserved	Reserved
0x400[3:2]	BOOT_CFG_LOCK	2	Perform lock on BOOT related fuses.	00 - Unlock (The controlled field can be read, sensed, burned or overridden in the corresponded IIM register) x1 - Write Lock (The controlled field can be read or sensed only) 1x - Operation Lock (The controlled field can't be overridden)	N/A
0x400[5:4]	Reserved	2	Reserved	Reserved	Reserved
0x400[6]	SJC_RESP_LOCK	1	SJC response lock	0 - Unlock (The controlled field can be read, sensed, burned or overridden in the corresponded IIM register) 1 - Lock (The controlled field can't be read, sensed, burned or overridden in the corresponded IIM register)	N/A
0x400[7]	Reserved	1	Reserved	Reserved	Reserved
0x400[9:8]	MAC_ADDR_LOCK	2	Lock MAC_ADDR fuses.	00 - Unlock (The controlled field can be read, sensed, burned or overridden in the corresponded IIM register) x1 - Write Lock (The controlled field can be read or sensed only) 1x - Operation Lock (The controlled field can't be overridden)	N/A
0x400[11:10]	GP1_LOCK	2	Lock for General Purpose fuse register #1 (GP1)	00 - Unlock (The controlled field can be read, sensed, burned or overridden in the corresponded IIM register) x1 - Write Lock (The controlled field can be read or sensed only) 1x - Operation Lock (The controlled field can't be overridden)	N/A

Table continues on the next page...

Table 22-9. Fusemap Descriptions (continued)

Fuse Address	Fuses Name	Number of Fuses	Fuses Function	Setting	Locked by
0x400[13:12]	GP2_LOCK	2	Lock for General Purpose fuse register #2 (GP2)	00 - Unlock (The controlled field can be read, sensed, burned or overridden in the corresponded IIM register) x1 - Write Lock (The controlled field can be read or sensed only) 1x - Operation Lock (The controlled field can't be overridden)	N/A
0x400[14]	Reserved	1	Reserved	Reserved	Reserved
0x400[15]	Reserved	1			Reserved
0x400[16]	SW_GP1_LOCK	1	Lock for SW_GP1 fuse.	0 - Unlock (The controlled field can be read, sensed, burned or overridden in the corresponded IIM register) 1 - Lock (The controlled field can't be sensed, burned or overridden in the corresponded IIM register)	N/A
0x400[17]	Reserved	1	Reserved	Reserved	Reserved
0x400[19:18]	ANALOG_LOCK	2	Lock for analog related fuse.	00 - Unlock (The controlled field can be read, sensed, burned or overridden in the corresponded IIM register) x1 - Write Lock (The controlled field can be read or sensed only) 1x - Operation Lock (The controlled field can't be overridden)	N/A
0x400[20]	Reserved	1	Reserved	Reserved	Reserved
0x400[21]	Reserved	1	Reserved	Reserved	Reserved
0x400[22]	MISC_CONF_LOCK	1			N/A
0x400[23]	Reserved	1	Reserved	Reserved	Reserved
0x400[25:24]	Reserved	2	Reserved	Reserved	Reserved
0x400[27:26]	GP3_LOCK	2	Lock for GP3 fuse	0 - Unlock (The controlled field can be read, sensed, burned, or overridden in the corresponding IIM register) x1 - Write Lock (The controlled field can be read or sensed only) 1x - Operation Lock (The controlled field cannot be overridden)	N/A
0x400[29:28]	Reserved	2	Reserved	Reserved	Reserved
0x400[30]	Reserved	1	Reserved	Reserved	Reserved
0x400[31]	Reserved	1	Reserved	Reserved	Reserved

Table continues on the next page...

Table 22-9. Fusemap Descriptions (continued)

Fuse Address	Fuses Name	Number of Fuses	Fuses Function	Setting	Locked by
0x420 - 0x410	SJC_CHALL/ UNIQUE_ID[63:0]	64	SJC CHALLENGE / Unique ID		N/A
0x430[7:0]	Reserved	8	Reserved	Reserved	Reserved
0x430[15:8]	Reserved	8	Reserved	Reserved	Reserved
0x430[19:16]	SI_REV[3:0]	4	Silicon Revision number		N/A
0x430[20]	Reserved	1	Reserved	Reserved	Reserved
0x430[21]	Reserved	1	Reserved	Reserved	Reserved
0x430[22]	Reserved	1	Reserved	Reserved	Reserved
0x430[23]	Reserved	1	Reserved	Reserved	Reserved
0x430[24]	Reserved	1	Reserved	Reserved	Reserved
0x430[25]	Reserved	1	Reserved	Reserved	Reserved
0x430[26]	Reserved	1	Reserved	Reserved	Reserved
0x430[27]	Reserved	1	Reserved	Reserved	Reserved
0x430[28]	Reserved	1	Reserved	Reserved	Reserved
0x430[29]	Reserved	1	Reserved	Reserved	Reserved
0x430[30]	Reserved	1	Reserved	Reserved	Reserved
0x430[31]	Reserved	1	Reserved	Reserved	Reserved
0x440[0]	Reserved	1	Reserved	Reserved	Reserved
0x440[1]	Reserved	1	Reserved	Reserved	Reserved
0x440[2]	Reserved	1	Reserved	Reserved	Reserved
0x440[3]	Reserved	1	Reserved	Reserved	Reserved
0x440[4]	Reserved	1	Reserved	Reserved	Reserved
0x440[5]	Reserved	1	Reserved	Reserved	Reserved
0x440[6]	Reserved	1	Reserved	Reserved	Reserved
0x440[7]	Reserved	1	Reserved	Reserved	Reserved
0x440[8]	Reserved	1	Reserved	Reserved	Reserved
0x440[9]	Reserved	1	Reserved	Reserved	Reserved
0x440[10]	Reserved	1	Reserved	Reserved	Reserved
0x440[11]	Reserved	1	Reserved	Reserved	Reserved
0x440[12]	Reserved	1	Reserved	Reserved	Reserved
0x440[13]	Reserved	1	Reserved	Reserved	Reserved
0x440[14]	Reserved	1	Reserved	Reserved	Reserved
0x440[15]	Reserved	1	Reserved	Reserved	Reserved
0x440[17:16]	SPEED_GRADING[1:0]	2	Burned by tester program, for indicating IC core speed	FRAL[0:1] MHz P/N Code 00 Reserved Reserved 01 500 05 10 600 06 11 Reserved Reserved	TESTER_LOCK
0x440[19:18]	Reserved	2	Reserved	Reserved	Reserved

Table continues on the next page...

Table 22-9. Fusemap Descriptions (continued)

Fuse Address	Fuses Name	Number of Fuses	Fuses Function	Setting	Locked by
0x440[20]	Reserved	1	Reserved	Reserved	Reserved
0x440[21]	Reserved	1	Reserved	Reserved	Reserved
0x440[23:22]	Reserved	2	Reserved	Reserved	Reserved
0x440[24]	Reserved	1	Reserved	Reserved	Reserved
0x440[25]	Reserved	1	Reserved	Reserved	Reserved
0x440[26]	Reserved	1	Reserved	Reserved	Reserved
0x440[27]	Reserved	1	Reserved	Reserved	Reserved
0x440[28]	Reserved	1	Reserved	Reserved	Reserved
0x440[29]	Reserved	1	Reserved	Reserved	Reserved
0x440[30]	Reserved	1	Reserved	Reserved	Reserved
0x440[31]	Reserved	1	Reserved	Reserved	Reserved
0x450[7:0]	BOOT_CFG1	8	BOOT configuration register #1, Usage varies, depending on selected boot device.	0x0000 - FlexSPI (NOR) boot 0x01xx - SD boot 0x10xx - MMC/eMMC boot 0x0001 - SEMC (NAND) boot 0x001x - SEMC (NOR) boot 0x10xx - FlexSPI (serial NAND) boot Others - Reserved Refer to Fuse Map for details.	BOOT_CFG_LOCK
0x450[15:8]	BOOT_CFG2	8	BOOT configuration register #2, Usage varies, depending on selected boot device.	See fuse-map tab for details.	BOOT_CFG_LOCK
0x450[23:16]	BOOT_CFG3	8	BOOT configuration register #3	See fuse-map tab for details.	BOOT_CFG_LOCK
0x450[31:24]	BOOT_CFG4	8	BOOT configuration register #4	See fuse-map tab for details.	BOOT_CFG_LOCK
0x460[0]	Reserved	1	Reserved	Reserved	Reserved
0x460[1]	Reserved	1	Reserved	Reserved	Reserved
0x460[2]	BOOT_FREQ	1	Determines, ARM Core and Bus frequencies during boot	0 - (ARM) 396 / (Bus) 132 MHz 1 - (ARM) 528 / (Bus) 132 MHz	BOOT_CFG_LOCK
0x460[3]	Reserved	1	Reserved	Reserved	Reserved
0x460[4]	BT_FUSE_SEL	1	Determines, whether using fuses for boot configuration, or GPIO / Serial loader	If boot_mode="00" (Development) 0=Boot mode configuration is taken from GPIOs. 1=Boot mode configuration is taken from fuses. If boot_mode="10" (Production)	BOOT_CFG_LOCK

Table continues on the next page...

Table 22-9. Fusemap Descriptions (continued)

Fuse Address	Fuses Name	Number of Fuses	Fuses Function	Setting	Locked by
				0 - Boot using serial downloader mode 1- Boot mode configuration is taken from fuses.	
0x460[5]	FORCE_COLD_BOOT(SBMR)	1	Force cold boot when core comes out of reset. Reflected in SBMR register of SRC	Fuse Function: 0 – Default behavior equivalent to the rest of the product family allowing a fast recovery from low power modes. That is, the ROM is allowed to jump to the address previously programmed in the SRC persistent register. 1 – Fast recovery path in the ROM is not allowed and a cold boot is always performed. Customers wanting a higher level of security should burn this fuse.	BOOT_CFG_LOCK
0x460[6]	Reserved	1	Reserved	Reserved	Reserved
0x460[7]	Reserved	1	Reserved	Reserved	Reserved
0x460[11:8]	SDRAM_CONFIG[7:0]	4	(SDRAM config options)		BOOT_CFG_LOCK
0x460[13:12]	Reserved	2	Reserved	Reserved	Reserved
0x460[15:14]	Reserved	2	Reserved	Reserved	Reserved
0x460[16]	FORCE_INTERNAL_BOOT	1	After this fuse is blown, the external BT_MODE[1:0] pins will be ignored, BootROM will take Boot Mode as “internal boot.”	0 - Boot Mode from BT_MODE pins 1 - BT_MODE pins be ignored, Boot Mode forced to "internal boot"	BOOT_CFG_LOCK
0x460[17]	SDP_DISABLE	1	Disable/Enable serial download support	0 - Serial download supported 1 - No serial download support	BOOT_CFG_LOCK
0x460[18]	SDP_READ_DISABLE	1	Disable/Enable serial download READ_REGISTER command.	0 - SDP READ_REGISTER command is enabled 1 - SDP READ_REGISTER is disabled	BOOT_CFG_LOCK
0x460[19]	DAP_SJC_SWD_SEL	1	Control DAP works in JTAG or SWD mode	0 - DAP works in SWD mode 1 - DAP works in JTAG mode	BOOT_CFG_LOCK
0x460[20]	Reserved	1	Reserved	Reserved	Reserved
0x460[21]	WDOG_ENABLE	1	Watchdog Enable	Used to specify whether to enable / not watchdog at boot. '0' - Watch-Dog is disabled. '1' - Watch-Dog is enabled.	BOOT_CFG_LOCK
0x460[23:22]	Reserved	2	Reserved	Reserved	Reserved

Table continues on the next page...

Table 22-9. Fusemap Descriptions (continued)

Fuse Address	Fuses Name	Number of Fuses	Fuses Function	Setting	Locked by
0x460[24]	DLL_ENABLE	1	Controls the enable/ disable of the DLL for SD/eMMC boot	0 - Disable DLL for SD/eMMC 1 - Enable DLL for SD/eMMC	BOOT_CFG_LOCK
0x460[25]	NAND_ECC_DISABLE	1	Indicate whether to enabled ECC (done by SW/Device HW) for Raw Nand device	0 - NAND ECC is enabled 0 - NAND ECC is disabled	BOOT_CFG_LOCK
0x460[26]	Reserved	1	Reserved	Reserved	Reserved
0x460[27]	Reserved	1	Reserved	Reserved	Reserved
0x460[28]	Reserved	1			BOOT_CFG_LOCK
0x460[29]	PWR_STABLE_CYCLE_SELECTION	1	Select Power Stable Cycle	0 - 5ms 1 - 2.5ms	BOOT_CFG_LOCK
0x460[31:30]	SD_PWR_CYCLE_SELECTION	2	Select SD Power Cycle	00 - 20ms 01 - 10ms 10 - 5ms 11 - 2.5ms	BOOT_CFG_LOCK
0x470[0]	Override SD Pad Settings	1	Overrides ROM default value for SD PAD control register. When set ROM will override SD pad control register with value programmed into PAD_SETTINGS fuse bits.		BOOT_CFG_LOCK
0x470[1]	BT_MPU_DISABLE	1	The fuse bit is used for ROM to not enable MPU	0 - MPU is enabled during boot 1 - MPU is disabled during boot	BOOT_CFG_LOCK
0x470[2]	L1 I-Cache DISABLE	1	The fuse bit is used for ROM to not enable L1 I-Cache	0 - L1 I-Cache is enabled during boot 1 - L1 I-Cache is disabled during boot	BOOT_CFG_LOCK
0x470[3]	Disable SDMMC Manufacture mode	1	The fuse bit is used to disable ROM feature “SD/MMC Manufacturing Mode”.	0 - Enable 1 – Disable	BOOT_CFG_LOCK
0x470[4]	UART Serial Download Disable	1	Disable UART Serial Download	0 - Disable 1 - Enable	BOOT_CFG_LOCK
0x470[5]	SD2 VOLTAGE SELECTION	1	Fuse bit to change voltage selection for SD2 pads. When set ROM will select 1.8V for SD3 pads otherwise 3.3V	0 - 3.3V 1 - 1.8V	BOOT_CFG_LOCK
0x470[6]	SD1_RST_POLARITY_SELECT	1	Select reset polarity for SD1	0 - Reset active low 1 - Reset active high	BOOT_CFG_LOCK

Table continues on the next page...

Table 22-9. Fusemap Descriptions (continued)

Fuse Address	Fuses Name	Number of Fuses	Fuses Function	Setting	Locked by
0x470[7]	DLL Override	1	Select the DLL mode for SD/eMMC.	0 - DLL Slave Mode for SD/eMMC 1 - DLL Override Mode for SD/eMMC	BOOT_CFG_LOCK
0x470[8]	USDHC IOMUX SRE Enable	1	The fuse bit is used for ROM to enable SRE bit for SD pads	0 - Disable 1 - Enable	BOOT_CFG_LOCK
0x470[9]	USDHC_IOMUX_SION_BIT_ENABLE	1	The fuse bit is used for ROM to enable SION bit for MUX control register.	0 - Disable 1 - Enable	BOOT_CFG_LOCK
0x470[10]	Boot Failure Indicator Pin Select[4]	1	Indicate Boot Failure	00000 - gpio1.IO00 00001 - gpio1.IO01 ... 11111 - gpio1.IO31 Note: Please refer RM for pins which gpio1.IOxx will be muxed to by default	BOOT_CFG_LOCK
0x470[11]	ENABLE_EMMC_22K_PULLUP	1	The fuse bit is used for ROM to enable 22K pullup for SD pads.	0 - 47K pullup 1 - 22K pullup	BOOT_CFG_LOCK
0x470[12]	USDHC_PAD_PULL_DOWN	1	The fuse bit is used for ROM to enable pull down bit for SD pads.	0 - no action 1 - pull down	BOOT_CFG_LOCK
0x470[13]	Override HYS bit for SD/MMC pads	1	After this fuse is blown, the [HYS] bit of IOMUXC_SW_PAD_CTL_PAD_SDx_CLK(x is the SD port which the ROM boot from), IOMUXC_SW_PAD_CTL_PAD_SDx_CMD, IOMUXC_SW_PAD_CTL_PAD_SDx_DAT0-n(n will be 0, 3, or 7, depends on the bus width selected) will be set.		BOOT_CFG_LOCK
0x470[14]	eMMC 4.4 - RESET TO PRE-IDLE STATE	1	After this fuse is blown, the CMD0 with argument 0xf0f0f0f0 will be sent to put the eMMC card into pre-IDLE state so that eMMC card's fast boot can work properly. This is useful for the warm boot, such as boot due to the WDOG reset.		BOOT_CFG_LOCK
0x470[15]	SD2_RST_POLARITY_SELECT	1	Select reset polarity for SD2	0 - Reset active low	BOOT_CFG_LOCK

Table continues on the next page...

Table 22-9. Fusemap Descriptions (continued)

Fuse Address	Fuses Name	Number of Fuses	Fuses Function	Setting	Locked by
				1 - Reset active high	
0x470[19:16]	Boot Failure Indicator Pin Select[3:0]	4	Misscellanious power management configuration bits.	00000 - gpio1.IO00 00001 - gpio1.IO01 ... 11111 - gpio1.IO31 Note: Please refer RM for pins which gpio1.IOxx will be muxed to by default	BOOT_CFG_LOCK
0x470[20]	Reserved	1	Reserved	Reserved	Reserved
0x470[22:21]	LPB_BOOT	2	Divide (Core / Bus) based on Boot Frequencies	'00' - Div by 1; '01' - Div by 2; '10' - Div by 4; '11' - Div by 8	BOOT_CFG_LOCK
0x470[23]	Reserved	1	Reserved	Reserved	Reserved
0x470[30:24]	MMC_DLL_DLY[6:0]	7	eMMC 4.4 delay line default value (set by boot rom), used in conjunction with "DLL Override" = 1 (BOOT_CFG3[3])	Conneted to LVDS module.	BOOT_CFG_LOCK
0x470[31]	Override NAND Pad Settings	1	Override pad settings for NAND boot.		BOOT_CFG_LOCK
0x480[5:0]	Reserved	6	Reserved	Reserved	Reserved
0x480[7:6]	Reserved	2	Reserved	Reserved	Reserved
0x480[15:8]	Reserved	8	Reserved	Reserved	Reserved
0x480[23:16]	Reserved	8	Reserved	Reserved	Reserved
0x480[26:24]	Reserved	3	Reserved	Reserved	Reserved
0x480[27]	Reserved	1	Reserved	Reserved	Reserved
0x480[29:28]	Reserved	2	Reserved	Reserved	Reserved
0x480[30]	Reserved	1	Reserved	Reserved	Reserved
0x480[31]	Reserved	1	Reserved	Reserved	Reserved
0x490[9:0]	Reserved	10	Reserved	Reserved	Reserved
0x490[23:10]	Reserved	14	Reserved	Reserved	Reserved
0x490[27:24]	Reserved	4	Reserved	Reserved	Reserved
0x490[31:28]	Reserved	4	Reserved	Reserved	Reserved
0x4A0[7:0]	Reserved	8	Reserved	Reserved	Reserved
0x4A0[19:8]	Reserved	76	Reserved	Reserved	Reserved
0x4C0[31:20]	Reserved	12	Reserved	Reserved	Reserved
0x4D0[31:0]	Reserved	32	Reserved	Reserved	Reserved
0x4E0[31:0]	Reserved	32	Reserved	Reserved	Reserved
0x4F0[15:0]	USB_VID[31:0]	16	USB VID		ANALOG_LOCK

Table continues on the next page...

Table 22-9. Fusemap Descriptions (continued)

Fuse Address	Fuses Name	Number of Fuses	Fuses Function	Setting	Locked by
0x4F0[31:16]	USB_PID[31:0]	16	USB PID		ANALOG_LOCK
0x500[31:0]	Reserved	256	Reserved	Reserved	Reserved
0x580[31:0]	Reserved	256	Reserved	Reserved	Reserved
0x600[23:0]	SJC_RESP[55:0]	56	Response reference value for the secure JTAG controller		SJC_RESP_LOCK (locks also for read and explicit sense)
0x600[31:24]	Reserved	8	Reserved	Reserved	Reserved
0x620[15:0]	MAC1_ADDR[47:0]	48	Ethernet MAC Address	PJRC programs unique mac address	MAC_ADDR_LOCK
0x630[31:16]	MAC2_ADDR[47:0]	48	Ethernet2 MAC Address		MAC_ADDR_LOCK
0x650[31:0]	Reserved	32	Reserved	Reserved	Reserved
0x660[31:0]	GP1[31:0]	32	General Purpose fuse register #1		GP1_LOCK
0x670[31:0]	GP2[31:0]	32	General Purpose fuse register #2		GP2_LOCK
0x680[31:0]	SW_GP1[31:0]	32	SW general purpose key		SW_GP1_LOCK
0x690[31:0]	Reserved	128	Reserved	Reserved	Reserved
0x6D0[5:0]	PAD_SETTINGS	6	Used with conjuction of MMC/SD/Nand "Override Pad Settings" fuse value, as follow: '0' - Use IO default settings for boot device IO pads. '1' - Use "Override" value, as set by this register.	IO pads settings of selected boot interface, are override with this fuses, as follow: [0] - Slew Rate [3:1] Drive Strength [5:4] - Speed Settings. Refer to IO PAD chapter for "Settings" fields value	MISC_CONF_LOCK
0x6D0[6]	USB_VBUS_EVENT_HANDLER_EN	1	Rom handle USB VBUS attach/detach event	0 - ROM not handle USB VBUS attach/detach event 1 - ROM handle USB VBUS attach/detach event	MISC_CONF_LOCK
0x6D0[7]	Enable Boot Failure Indicator Pin	1	Enable Boot Failure Indicator Pin	0 - disabled 1 - enabled	MISC_CONF_LOCK
0x6D0[11:8]	READ_RETRY_SEQ_ID	4	Choose Read Retry Sequence	0000 - don't use read retry(RR) sequence embedded in ROM 0001 - Micron 20nm RR sequence 0010 - Toshiba A19nm RR sequence	MISC_CONF_LOCK

Table continues on the next page...

Table 22-9. Fusemap Descriptions (continued)

Fuse Address	Fuses Name	Number of Fuses	Fuses Function	Setting	Locked by
				0011 - Toshiba 19nm RR sequence 0100 - SanDisk 19nm RR sequence 0101 - SanDisk 1ynmRR sequence Others - Reserved	
0x6D0[12]	Reserved	1	Unallocated	Reserved	Reserved
0x6D0[15:13]	WDOG Timeout Select	3	Seletct WDOG Timeout	000 - 64s 001 - 32s 010 - 16s 011 - 8s 100 - 4s Others - Reserved	MISC_CONF_LOCK
0x6D0[19:16]	Default_FlexRAM_Part	4	Default FlexRAM RAM bank partioning	CFG DTCM ITCM ORAM RAM_Bank0~15_CFG (D=DTCM, I=ITCM, O=OCRAM) 0000: 128KB 128KB 256KB {O O O O D D I I I I D D O O O O} 0001: 128KB 64KB 320KB {O O O O D D I I D D O O O O O O} 0010: 128KB 256KB 128KB {O O D D I I I I I I I I D D O O} 0011: 128KB 32KB 352KB {O O O D D D D I O O O O O O O O} 0100: 64KB 128KB 320KB {O O O O D D I I I I O O O O O O} 0101: 64KB 64KB 384KB {O O O O D D I I O O O O O O O O} 0110: 64KB 256KB 192KB {O O D D I I I I I I I I O O O O} 0111: 0KB 442KB 64KB {O O I I I I I I I I I I I I I I} 1000: 256KB 128KB 128KB {O O D D D D D I I I I D D D D O O} 1001: 256KB 64KB 192KB {O O D D D D I I D D D D O O O O} 1010: 192KB 256KB 64KB {O O D D I I I I I I I I D D D D} 1011: 448KB 0KB 64KB {O O D D D D D D D D D D D D D D}	MISC_CONF_LOCK

Table continues on the next page...

Table 22-9. Fusemap Descriptions (continued)

Fuse Address	Fuses Name	Number of Fuses	Fuses Function	Setting	Locked by
				1100: 0KB 128KB 384KB {O O O O I I I I O O O O O O O O} 1101: 32KB 32KB 448KB {O O O D I O O O O O O O O O O O O} 1110: 0KB 256KB 256KB {O O I I I I I I I I O O O O O O} 1111: 0KB 0KB 512KB {O O O O O O O O O O O O O O O O}	
0x6D0[21:20]	Reserved	2	Reserved	Reserved	Reserved
0x6D0[22]	Reserved	1	Reserved	Reserved	Reserved
0x6D0[23]	Reserved	1	Reserved	Reserved	Reserved
0x6D0[24]	EEPROM_RECOVERY_EN	1	EEPROM Recovery Enable	0' - Disabled '1' - Enabled	MISC_CONF_LOCK
0x6D0[26:25]	LPSPI_PORT_SEL	2	LPSPI Port Select	00 - LPSPI1 SPI2 01 - LPSPI2 10 - LPSPI3 SPI1 11 - LPSPI4 SPI	MISC_CONF_LOCK
0x6D0[27]	LPSPI_ADDR	1	SPI Addressing	0 - 3-bytes (24-bit) 1 - 2-bytes (16-bit)	MISC_CONF_LOCK
0x6D0[29:28]	LPSPI_SPEED	2	LPSPI Speed select	00 - 20 MHz (default) 01 - 10 MHz 10 - 5 MHz 11 - 2 MHz	MISC_CONF_LOCK
0x6E0[7:0]	BOOT_CONFIG_MISC0	8	boot configuration misc		MISC_CONF_LOCK
0x6E0[15:8]	BOOT_CONFIG_MISC1	8	boot configuration misc		MISC_CONF_LOCK
0x6E0[23:16]	BOOT_CONFIG_MISC2	8	boot configuration misc		MISC_CONF_LOCK
0x6E0[31:24]	BOOT_CONFIG_MISC3	8	boot configuration misc		MISC_CONF_LOCK
0x6F0[2:0]	Reserved	3	Reserved	Reserved	Reserved
0x6F0[3]	Reserved	1	Reserved	Reserved	Reserved
0x6F0[7:4]	Reserved	4	Reserved	Reserved	Reserved
0x6F0[15:8]	Reserved	8	Reserved	Reserved	Reserved
0x6F0[31:16]	Reserved	16	Reserved	Reserved	Reserved
0x800[31:0]	Reserved	256	Reserved	Reserved	Reserved
0x880[31:0]	GP3[127:0]	128	General Purpose fuse register #3		GP3_LOCK
0x8C0[31:0]	Reserved	128	Reserved	Reserved	Reserved

Chapter 23
On-Chip OTP Controller (OCOTP_CTRL)

23.1 Chip-specific OCOTP_CTRL information

Table 23-1. Reference links to related information

Topic	Related module(s)	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
Interrupts, DMA Events and XBAR Assignments	-	Interrupts, DMA Events and XBAR Assignments

NOTE

Security-related fuses are shown as reserved here. For details, refer to the Security Reference Manual.

23.2 Overview

This section contains information describing the On-Chip OTP controller (OCOTP_CTRL) along with details about the block functionality and implementation.

In this document, the words "eFuse" and "OTP" are interchangeable. OCOTP refers to the hardware block itself.

23.2.1 Block Diagram

The system level block diagram for OCOTP is provided below.

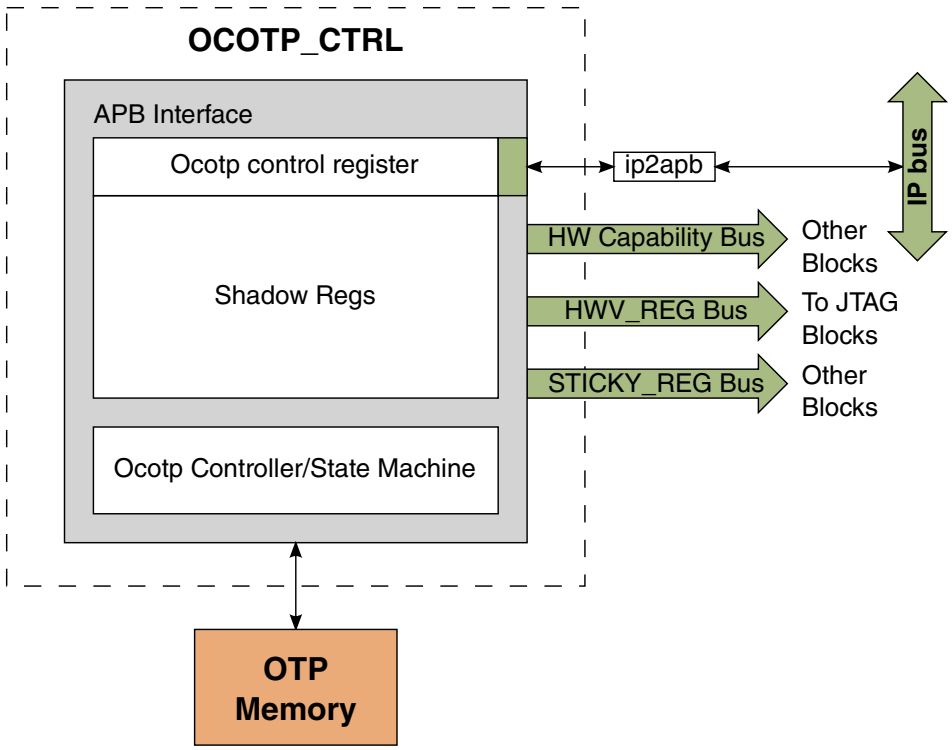


Figure 23-1. OCOTP System Level Block Diagram

23.2.2 Features

The OCOTP provides the following features :

- 32-bit word restricted program and read to 2 kbit (128 x 8) of eFuse.
- Loading and housing of eFuse content into shadow registers
- Memory-mapped (restricted) access to shadow registers
- Provides program-protect and read-protect of eFuse
- Provides override and read protection of shadow register

23.3 Functional Description

23.3.1 Operations

The IP bus interface of the OCOTP has the following functions.

- Configure control registers for programming and reading all the fuse words
- Override and read shadow registers

OCOTP configuration for programs and reads are performed on 32-bit words for software (SW) convenience. For writes, the 32-bit word reflects the "write-mask". Bit fields with 0 will not be programmed and bit fields with 1 will be programmed. OCOTP will program bit field with 1 in the fuse word one bit by one bit. For reads, OCOTP will read 4 times to get 4 bytes in the fuse word in order.

In this document, 2 kbit fuse are divided into 8 banks by function. Each bank has 8 fuse words. Physically, 2 kbits fuse are in two 128x8 efusebox.

23.3.1.1 Shadow Register Reload

All fuse words in efuse box are shadowed. This means that at reset, fuse values are read and then copied into memory-mapped shadow registers. If fuses are subsequently programmed, the shadow registers should be reloaded to keep them coherent with the fuse bank arrays.

The "reload shadows" feature allows the user to force a reload of the shadow registers without having to reset the device. To force a reload, complete the following steps:

1. Set the TIMING[STROBE_READ] field value appropriately.
2. Set the TIMING[RELAX] field value appropriately.
3. Check that CTRL[BUSY] and CTRL[ERROR] are clear. Overlapped accesses are not supported by the controller. Any pending write, read, or reload must be completed before a new access can be requested.
4. Set the CTRL[RELOAD_SHADOWS] bit. OCOTP will read all the fuses one by one and put it into corresponding shadow register.
5. Wait for CTRL[BUSY] and CTRL[RELOAD_SHADOWS] to be cleared by the controller.

The controller will automatically clear the CTRL[RELOAD_SHADOWS] bit after the successful completion of the operation.

23.3.1.2 Fuse and Shadow Register Read

All shadow registers are always readable through the IPS bus. When their corresponding fuse lock bits are set, the shadow registers also become read locked. After read locking, reading from these registers will return 0xBADABADA.

In addition, CTRL[ERROR] will be set. It must be cleared by software before any new write, read, or reload access can be issued. Subsequent reads to unlocked shadow registers will still work successfully.

To read fuse words directly from the fusebox instead of the shadow registers, complete the following steps:

1. Program TIMING[STROBE_READ] and TIMING[RELAX] fields with timing values to match the current frequency of the ipg_clk. OTP read will work at maximum bus frequencies as long as the TIMING parameters are set correctly.
2. Check that CTRL[BUSY] and CTRL[ERROR] are clear. Overlapped accesses are not supported by the controller. Any pending write, read, or reload must be completed before a read access can be requested.
3. Write the requested fuse address to CTRL[ADDR].
4. Set READ_CTRL[READ_FUSE] to start the read operation.
5. Wait for the controller to clear CTRL[BUSY]. A read request to a protected or locked region will result in no OTP access and no setting of CTRL[BUSY]. In addition, CTRL[ERROR] will be set. It must be cleared by software before any new access can be issued.
6. Read READ_FUSE_DATA to get fuse word value. READ_FUSE_DATA will be 0xBADABADA when CTRL[ERROR] is set.

23.3.1.3 Fuse and Shadow Register Writes

Shadow register bits can be overridden by software until the corresponding fuse lock bit for the region is set. When the lock shadow bit is set, the shadow registers for that lock region become write locked. The LOCKn register also has no overwrite or fuse lock bits, but it is always read-only.

In order to avoid "rogue" code performing erroneous writes to OTP, a special unlocking sequence is required for writes to the fuse banks. To program fuse bank complete the following steps:

1. Program the following fields with timing values to match the frequency of ipg_clk:
   - TIMING[STROBE_PROG]
   - TIMING[RELAX]

OTP writes will work at maximum bus frequencies as long as the TIMING parameters are set correctly.

2. Check that CTRL[BUSY] and CTRL[ERROR] are clear. Overlapped accesses are not supported by the controller. Any pending write, read, or reload must be completed before a write access can be requested.
3. Write the requested address to CTRL[ADDR] and program the unlock code into CTRL[WR_UNLOCK]. The unlock code must be programmed for each write access. The unlock code is documented in the register description. Both the unlock code and address can be written in the same operation.
4. Write the data to the DATA register. This will automatically set CTRL[BUSY] and clear CTRL[WR_UNLOCK]. Bit fields with 1's will result in that OTP bit being programmed. Bit fields with 0's will be ignored. At the same time that the write is accepted, the controller makes an internal copy of CTRL[ADDR] which cannot be updated until the next write sequence is initiated. This copy guarantees that erroneous writes to CTRL[ADDR] will not affect an active write operation. During the write operation, DATA cannot be modified.
5. Wait for the controller to clear CTRL[BUSY]. A write request to a protected or locked region will result in no OTP access and no setting of CTRL[BUSY]. In addition CTRL[ERROR] will be set. It must be cleared by software before any new write access can be issued.

It should be noted that write latencies to OTP are numbers of 10 micro-seconds per word. The write latency is based on the number of "1" bits being written. For example, to program half the fuse bits in one 32-bt word requires 10 us x 16.

23.3.1.4 Error Conditions

The CTRL[ERROR] bit will be set under the following conditions:

- A write is performed to a shadow register during a shadow reload (CTRL[RELOAD_SHADOWS] is set). In addition, the contents of the shadow register shall not be updated.
- A write is performed to a shadow register which has been locked.
- A read is performed to a shadow register which has been read locked.
- A program is performed to a fuse word which has been locked.
- A read is performed to a fuse word which has been read locked.
- Overlapping access - a read, write, or reload is attempted while a read, write, or reload is already in progress.

23.3.1.5 Write Postamble

Due to internal electrical characteristics of the OTP during writes, all OTP operations (direct reads, shadow reloads, or other writes) following a write must wait at least 2 µs after CTRL[BUSY] clears. The delay guarantees programming voltages on-chip reach a steady state when exiting a write sequence.

A recommended software sequence to meet the postamble requirements is as follows:

1. Issue the write and poll for CTRL[BUSY].
2. After CTRL[BUSY] is clear, wait 2 µs.
3. Perform the next OTP operation.

23.3.2 Fuse Shadow Memory Footprint

The OTP memory footprint is shown in the following figure. The registers are grouped by lock region. Their names correspond to the fusemap names.

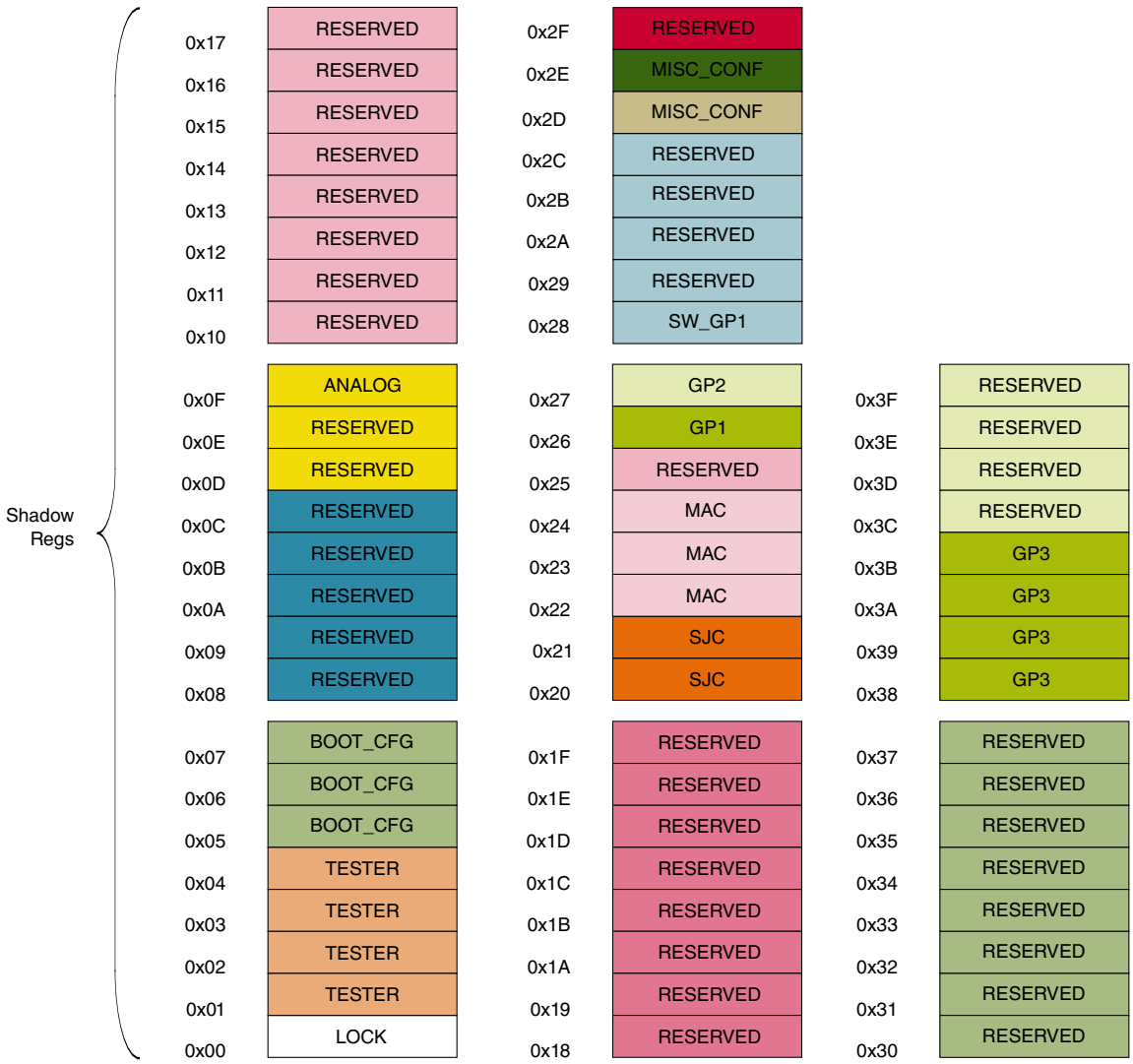


Figure 23-2. OTP Memory Footprint

23.3.3 OTP Read/Write Timing Parameters

The timing fields contained in the HW_OCOTP_TIMING register that specify counter limit values, which are used to time how long the state machine remains in the various states, as well as specify the STROBE signal timing.

The timing parameters are specified in ipg_clk cycles. Since the ipg_clk frequency can be set to a range of values, these parameters must be adjusted with the clock to yield the appropriate delay.

Register Field	Description

Table continues on the next page...

HW_OCOTP_TIMING[WAIT]	"WAIT" specifies time interval between auto read and write access in one time program. t_{SP_RD}=(WAIT+1)/ipg_clk_freq, should be ≧ 150 ns.
HW_OCOTP_TIMING[RELAX]	"RELAX" is used to configure the setup/hold time for certain timing margin. t_{SP_PGM}=t_{HP_PGM}=(RELAX+1)/ipg_clk_freq, should be ≧ 100 ns.
HW_OCOTP_TIMING[STROBE_PROG]	t_{PGM} = [(STROBE_PROG+1) – 2×(RELAX_PROG+1)]/ ipg_clk_freq. The t_{PGM} should be configured within the range of 9000 ns < t_{PGM} < 11000 ns, while its recommended value is 10000 ns.
HW_OCOTP_TIMING[STROBE_READ]	t_{RD}= [(STROBE_READ+1) – 2×(RELAX_READ+1)]/ ipg_clk_freq. The t_{RD} is required to be larger than 45 ns. t_{AEN}= (STROBE_READ+1)/ipg_clk_freq. The t_{AEN} is required to be larger than 75 ns.
HW_OCOTP_TIMING2[RELAX_READ]	"RELAX_READ" is used to configure the setup/hold time for certain timing margin. (RELAX_READ+1)/ipg_clk_freq should be ≧ 10 ns.
HW_OCOTP_TIMING2[RELAX_PROG]	"RELAX_PROG" is used to configure the setup/hold time for certain timing margin. (t_{AEN} – t_{PGM})/2 = (RELAX_PROG+1)/ipg_clk_freq, should be ≧ 950 ns.

23.3.4 Resets

The OCOTP is always active. The shadow registers automatically load the appropriate OTP contents after reset is deasserted. During this load-time CTRL[BUSY] is set.

23.3.5 Clocks

The table found here describes the clock sources for OCOTP. Please see the chip-specific clocking section for clock setting, configuration and gating information.

Table 23-2. OCOTP Clocks

Clock name	Description
ipg_clk	Peripheral clock
ipg_clk_s	Peripheral access clock

23.4 External Signals

There are no external signals pinned out for OCOTP.

23.5 Fuse Map

See the Fusemap chapter of this reference manual for more information.

23.6 Memory Map/Register Definition

The OCOTP Memory Map/Register Definition can be found here.

NOTE

Writing or reading an unimplemented register address in the OCOTP Controller will not send a bus error and read data will be 0.

23.6.1 OCOTP register descriptions

OCOTP Hardware Register Format Summary

23.6.1.1 OCOTP memory map

OCOTP base address: 401F_4000h

Offset	Register	Width (In bits)	Access	Reset value
0h	OTP Controller Control and Status Register (CTRL)	32	RW	0000_0000h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
4h	OTP Controller Control and Status Register (CTRL_SET)	32	RW	0000_0000h
8h	OTP Controller Control and Status Register (CTRL_CLR)	32	RW	0000_0000h
Ch	OTP Controller Control and Status Register (CTRL_TOG)	32	RW	0000_0000h
10h	OTP Controller Timing Register (TIMING)	32	RW	0C0D_9755h
20h	OTP Controller Write Data Register (DATA)	32	RW	0000_0000h
30h	OTP Controller Write Data Register (READ_CTRL)	32	RW	0000_0000h
40h	OTP Controller Read Data Register (READ_FUSE_DATA)	32	RW	0000_0000h
60h	Software Controllable Signals Register (SCS)	32	RW	0000_0000h
64h	Software Controllable Signals Register (SCS_SET)	32	RW	0000_0000h
68h	Software Controllable Signals Register (SCS_CLR)	32	RW	0000_0000h
6Ch	Software Controllable Signals Register (SCS_TOG)	32	RW	0000_0000h
90h	OTP Controller Version Register (VERSION)	32	RO	0300_0000h
100h	OTP Controller Timing Register 2 (TIMING2)	32	RW	01C3_0092h
400h	Value of OTP Bank0 Word0 (Lock controls) (LOCK)	32	RW	4012_8003h
410h	Value of OTP Bank0 Word1 (Configuration and Manufacturing Info.) (CFG0)	32	RW	0000_0000h
420h	Value of OTP Bank0 Word2 (Configuration and Manufacturing Info.) (CFG1)	32	RW	0000_0000h
430h	Value of OTP Bank0 Word3 (Configuration and Manufacturing Info.) (CFG2)	32	RW	0000_0000h
440h	Value of OTP Bank0 Word4 (Configuration and Manufacturing Info.) (CFG3)	32	RW	0000_0000h
450h	Value of OTP Bank0 Word5 (Configuration and Manufacturing Info.) (CFG4)	32	RW	0000_0000h
460h	Value of OTP Bank0 Word6 (Configuration and Manufacturing Info.) (CFG5)	32	RW	0000_0000h
470h	Value of OTP Bank0 Word7 (Configuration and Manufacturing Info.) (CFG6)	32	RW	0000_0000h
4F0h	Value of OTP Bank1 Word7 (Analog Info.) (ANA2)	32	RW	0000_0000h
600h	Value of OTP Bank4 Word0 (Secure JTAG Response Field) (SJC_RESP0)	32	RW	0000_0000h
610h	Value of OTP Bank4 Word1 (Secure JTAG Response Field) (SJC_RESP1)	32	RW	0000_0000h
620h	Value of OTP Bank4 Word2 (MAC Address) (MAC0)	32	RW	0000_0000h
630h	Value of OTP Bank4 Word3 (MAC Address) (MAC1)	32	RW	0000_0000h
640h	Value of OTP Bank4 Word4 (MAC2 Address) (MAC2)	32	RW	0000_0000h
660h	Value of OTP Bank4 Word6 (General Purpose Customer Defined Info) (GP1)	32	RW	0000_0000h
670h	Value of OTP Bank4 Word7 (General Purpose Customer Defined Info) (GP2)	32	RW	0000_0000h
680h	Value of OTP Bank5 Word0 (SW GP1) (SW_GP1)	32	RW	0000_0000h
6D0h	Value of OTP Bank5 Word5 (Misc Conf) (MISC_CONF0)	32	RW	0000_0000h
6E0h	Value of OTP Bank5 Word6 (Misc Conf) (MISC_CONF1)	32	RW	0000_0000h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
880h	Value of OTP Bank7 Word0 (GP3) (GP30)	32	RW	0000_0000h
890h	Value of OTP Bank7 Word1 (GP3) (GP31)	32	RW	0000_0000h
8A0h	Value of OTP Bank7 Word2 (GP3) (GP32)	32	RW	0000_0000h
8B0h	Value of OTP Bank7 Word3 (GP3) (GP33)	32	RW	0000_0000h

23.6.1.2 OTP Controller Control and Status Register (CTRL)

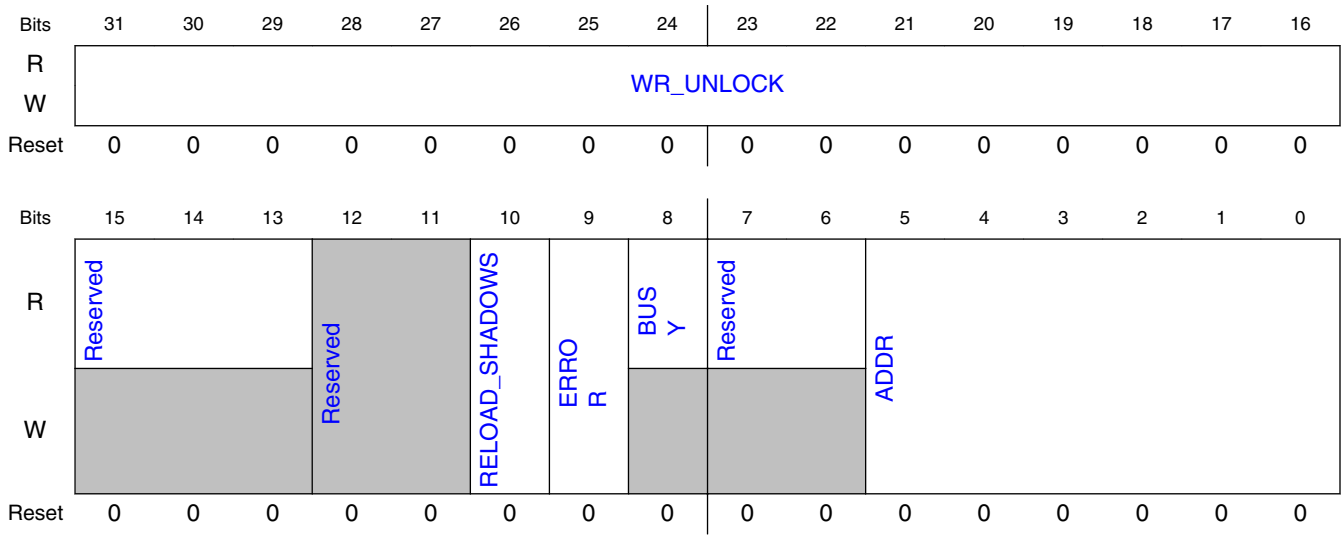
23.6.1.2.1 Offset

Register	Offset	Description
CTRL	0h	OTP Controller Control and Status Register
CTRL_SET	4h	Writing a 1 to a bit in this register sets the corresponding bit in CTRL
CTRL_CLR	8h	Writing a 1 to a bit in this register clears the corresponding bit in CTRL
CTRL_TOG	Ch	Writing a 1 to a bit in this register toggles the corresponding bit in CTRL

23.6.1.2.2 Function

The OCOTP Control and Status Register provides the necessary software interface for performing read and write operations to the On-Chip OTP (One-Time Programmable ROM). The control fields such as WR_UNLOCK, ADDR, BUSY and ERROR may be used in conjuction with the DATA register to perform write operations. Read operations to the On-Chip OTP involve ADDR, BUSY and ERROR bit fields and the READ_CTRL register. The read value is saved in the READ_FUSE_DATA register.

23.6.1.2.3 Diagram



23.6.1.2.4 Fields

Field	Function
31-16 WR_UNLOCK	Write Unlock Write 0x3E77 to enable OTP write accesses. NOTE: This register must be unlocked on a write-by-write basis (a write is initiated when DATA is written), so the UNLOCK bitfield must contain the correct key value during all writes to DATA, otherwise a write shall not be initiated. This field is automatically cleared after a successful write completion (clearing of BUSY). Values not listed are reserved. 0000000000000000b - OTP write access is locked. 0011111001110111b - OTP write access is unlocked.
15-13 —	Reserved
12-11 —	Reserved
10 RELOAD_SHADOWS	Reload Shadow Registers This bit can be set to force re-loading of the shadow registers. This operation will automatically set BUSY. After the shadow registers have been re-loaded, BUSY and RELOAD_SHADOWS are automatically cleared by the controller. 0b - Do not force shadow register re-load. 1b - Force shadow register re-load. This bit is cleared automatically after shadow registers are re-loaded.
9 ERROR	Locked Region Access Error This bit is set by the controller when an access to a locked region (OTP or shadow register) is requested. Reset this bit by writing a one to the CTRL_CLR[ERROR] bit and not by writing to the CTRL register. 0b - No error. 1b - Error - access to a locked region requested.

Table continues on the next page...

Field	Function
8 BUSY	OTP controller status bit This bit is automatically set by the controller when a write or read access to the OTP is started. When set, no new write access or read access to OTP (including RELOAD_SHADOWS) can be performed. This bit is cleared by the controller when an access completes. After reset (or after setting RELOAD_SHADOWS), this bit is set by the controller until the HW/SW and LOCK registers are successfully copied, after which time it is automatically cleared by the controller. 0b - No write or read access to OTP started. 1b - Write or read access to OTP started.
7-6 —	Reserved
5-0 ADDR	OTP write and read access address register Specifies one of 128 word address locations (0x00 - 0x3F).

23.6.1.3 OTP Controller Timing Register (TIMING)

23.6.1.3.1 Offset

Register	Offset
TIMING	10h

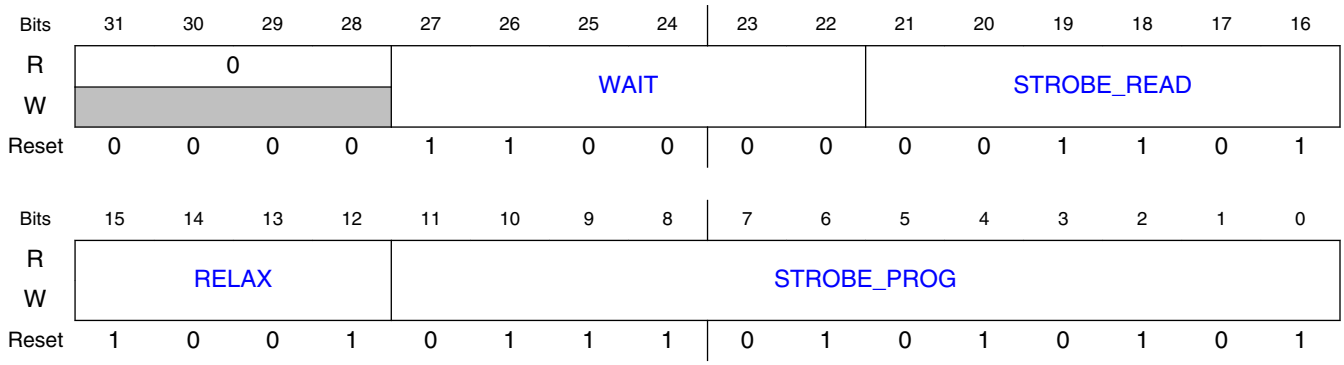
23.6.1.3.2 Function

The OCOTP Data Register is used for OTP Programming

This register specifies timing parameters for programming and reading the OTP fuse array.

See OTP Read/Write Timing Parameters for more details.

23.6.1.3.3 Diagram



23.6.1.3.4 Fields

Field	Function
31-28 —	Reserved
27-22 WAIT	Wait Interval This count value specifies the time interval between auto read and write access in one time program to the OTP. It is given in number of ipg_clk periods. The OTP controller performs a read action before each program action. This is the "auto" read. The purpose of the auto read is to avoid multiple programming to the same bits.
21-16 STROBE_READ	Read Strobe Period This count value specifies the strobe period in one time read OTP (Trd). Trd = ((STROBE_READ+1)- 2*(RELAX_READ+1)) /ipg_clk_freq. It is given in number of ipg_clk periods.
15-12 RELAX	Relax Count Value This count value specifies the time to add to all default timing parameters other than the Tpgm and Trd. It is given in number of ipg_clk periods.
11-0 STROBE_PROG	Write Strobe Period This count value controls the strobe period in one time to write OTP (Tpgm). Tpgm = ((STROBE_PROG +1)- 2*(RELAX_PROG+1)) /ipg_clk_freq. It is given in number of ipg_clk periods.

23.6.1.4 OTP Controller Write Data Register (DATA)

23.6.1.4.1 Offset

Register	Offset
DATA	20h

23.6.1.4.2 Function

This register is used in conjuction with the CTRL register to perform one-time writes to the OTP. See the Fuse and Shadow Register Writes section for operating details.

23.6.1.4.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

23.6.1.4.4 Fields

Field	Function
31-0 DATA	Used to initiate a write to OTP. See the Fuse and Shadow Register Writes section for operating details.

23.6.1.5 OTP Controller Write Data Register (READ_CTRL)

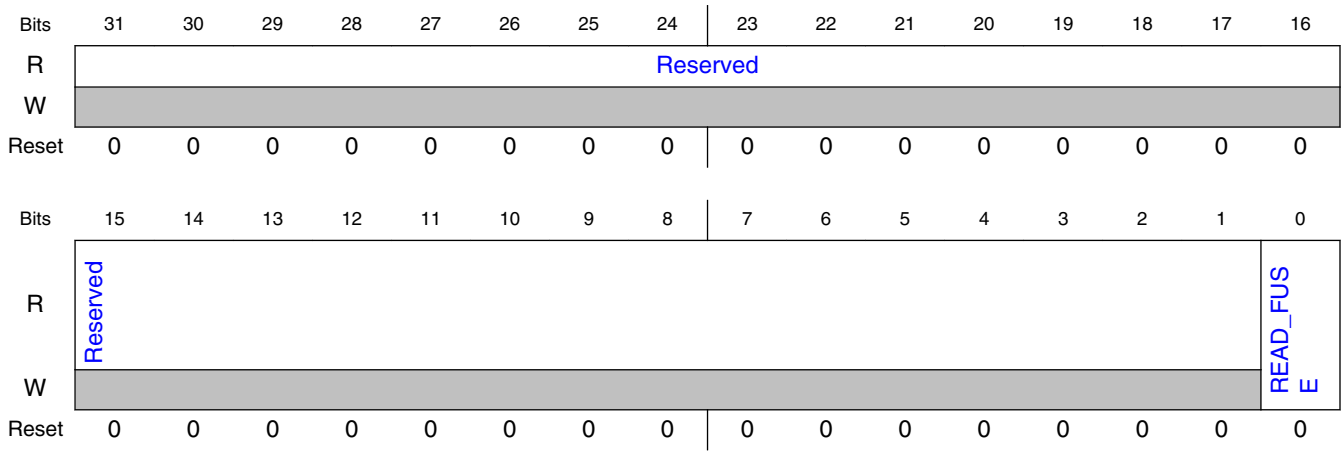
23.6.1.5.1 Offset

Register	Offset
READ_CTRL	30h

23.6.1.5.2 Function

This register is used in conjuction with CTRL to perform reads of the OTP. See the Fuse and Shadow Register Read section for operating details.

23.6.1.5.3 Diagram



23.6.1.5.4 Fields

Field	Function
31-1 —	Reserved
0 READ_FUSE	READ_FUSE Used to initiate a read from OTP. See the Fuse and Shadow Register Read section for operating details.

23.6.1.6 OTP Controller Read Data Register (READ_FUSE_DATA)

23.6.1.6.1 Offset

Register	Offset
READ_FUSE_DATA	40h

23.6.1.6.2 Function

The data read from OTP.

23.6.1.6.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

23.6.1.6.4 Fields

Field	Function
31-0 DATA	The data read from OTP. See the Fuse and Shadow Register Read section for operating details.

23.6.1.7 Software Controllable Signals Register (SCS)

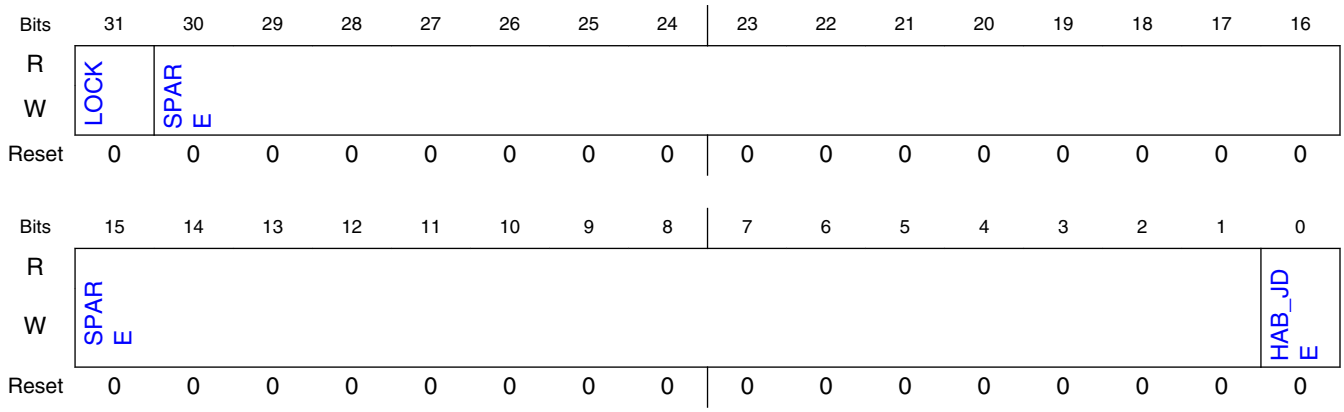
23.6.1.7.1 Offset

Register	Offset	Description
SCS	60h	Software Controllable Signals Register
SCS_SET	64h	Writing a 1 to a bit in this register sets the corresponding bit in SCS
SCS_CLR	68h	Writing a 1 to a bit in this register clears the corresponding bit in SCS
SCS_TOG	6Ch	Writing a 1 to a bit in this register toggles the corresponding bit in SCS

23.6.1.7.2 Function

This register holds volatile configuration values that can be set and locked by software.

23.6.1.7.3 Diagram



23.6.1.7.4 Fields

Field	Function
31 LOCK	Lock This bitfield locks the bits in this register. 0b - Bits in this register are unlocked. 1b - Bits in this register are locked. When set, all of the bits in this register are locked and can not be changed through SW programming. After this bit is set, it can only be cleared by a POR.
30-1 SPARE	Unallocated read/write bits for implementation specific software use.
0 HAB_JDE	HAB JTAG Debug Enable This bit can be used by the High Assurance Boot (HAB) portion of the Boot ROM to enable JTAG debugging, assuming that a properly signed command to do so is found and validated by the HAB. The HAB will lock the register before passing control to the application whether or not JTAG debugging has been enabled. Once JTAG is enabled by this bit, it cannot be disabled unless the system is reset by POR. 0b - JTAG debugging is not enabled by the HAB (it may still be enabled by other mechanisms). 1b - JTAG debugging is enabled by the HAB (though this signal may be gated off).

23.6.1.8 OTP Controller Version Register (VERSION)

23.6.1.8.1 Offset

Register	Offset
VERSION	90h

23.6.1.8.2 Function

This register indicates the RTL version in use.

23.6.1.8.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	MAJOR								MINOR							
W																
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	STEP															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

23.6.1.8.4 Fields

Field	Function
31-24 MAJOR	Major RTL Version Fixed read-only value reflecting the MAJOR field of the RTL version.
23-16 MINOR	Minor RTL Version Fixed read-only value reflecting the MINOR field of the RTL version.
15-0 STEP	RTL Version Steping Fixed read-only value reflecting the stepping of the RTL version.

23.6.1.9 OTP Controller Timing Register 2 (TIMING2)

23.6.1.9.1 Offset

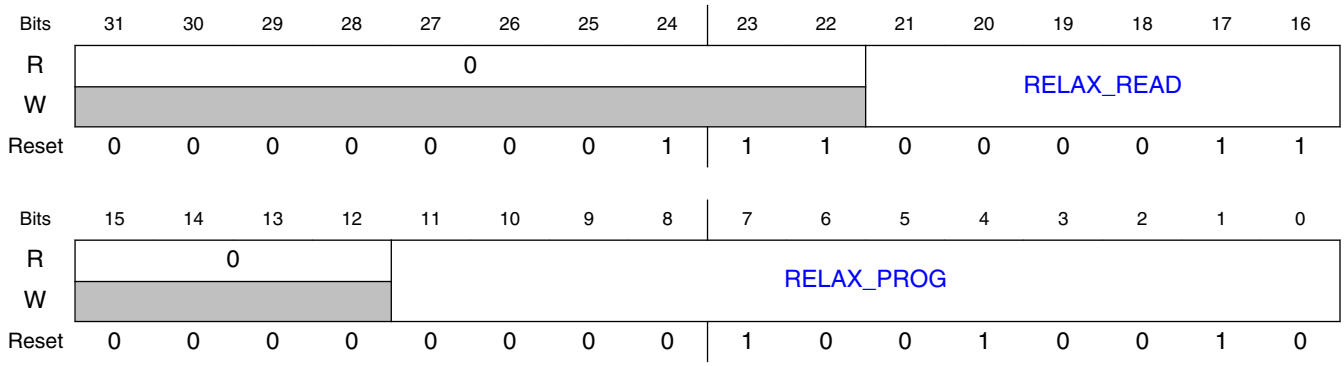
Register	Offset
TIMING2	100h

23.6.1.9.2 Function

This register specifies timing parameters for programming and reading the OCOTP fuse array.

See OTP Read/Write Timing Parameters for more details.

23.6.1.9.3 Diagram



23.6.1.9.4 Fields

Field	Function
31-22 —	Reserved
21-16 RELAX_READ	Relax Read count value This count value specifies the strobe period in one time read OTP. Trd = ((STROBE_READ+1)-2*(RELAX_READ+1)) /ipg_clk_freq. It is given in number of ipg_clk periods.
15-12 —	Reserved
11-0 RELAX_PROG	Relax Prog. count value This count value specifies the strobe period in one time write OTP. Tpgm = ((STROBE_PROG+1)-2*(RELAX_PROG+1)) /ipg_clk_freq. It is given in number of ipg_clk periods.

23.6.1.10 Value of OTP Bank0 Word0 (Lock controls) (LOCK)

23.6.1.10.1 Offset

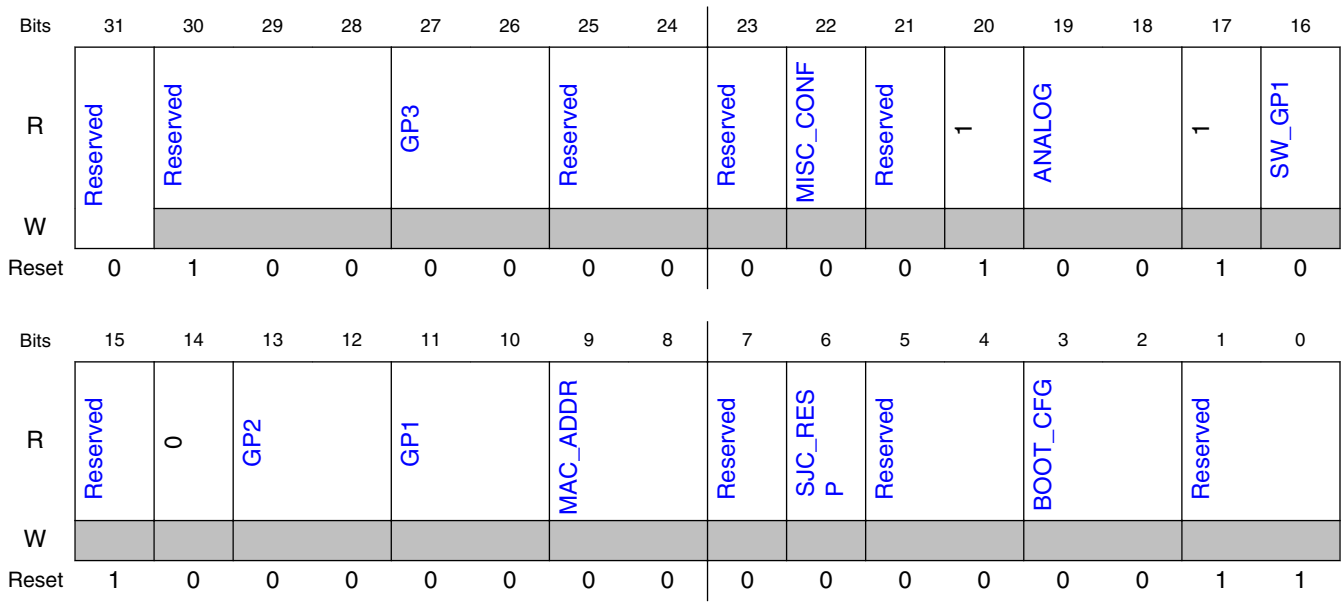
Register	Offset
LOCK	400h

23.6.1.10.2 Function

Shadowed memory mapped access to OTP Bank 0, word 0 (ADDR = 0x00).

Copied from the OTP automatically after reset. Can be re-loaded by setting CTRL[RELOAD_SHADOWS]

23.6.1.10.3 Diagram



23.6.1.10.4 Fields

Field	Function
31 —	Reserved
30-28 —	Reserved
27-26 GP3	GP3 Write Lock Status Status of shadow register and OTP write lock for GP3 region. When bit 1 is set, the writing of this region's shadow register is blocked. When bit 0 is set, the writing of this region's OTP fuse word is blocked.
25-24 —	Reserved
23 —	Reserved
22 MISC_CONF	MISC_CONF Write Lock Status Status of shadow register and OTP write lock for misc_conf region. 0b - Writing of this region's shadow register and OTP fuse word are not blocked. 1b - When set, the writing of this region's shadow register and OTP fuse word are blocked.
21 —	Reserved

Table continues on the next page...

Field	Function
20 —	Reserved
19-18 ANALOG	ANALOG Write Lock Status Status of shadow register and OTP write lock for analog region. When bit 1 is set, the writing of this region's shadow register is blocked. When bit 0 is set, the writing of this region's OTP fuse word is blocked.
17 —	Reserved
16 SW_GP1	SW_GP1 Write Lock Status Status of shadow register and OTP write lock for sw_gp1 region. 0b - Writing of this region's shadow register and OTP fuse word are not blocked. 1b - When set, the writing of this region's shadow register and OTP fuse word are blocked.
15 —	Reserved
14 —	Reserved
13-12 GP2	GP2 Write Lock Status Status of shadow register and OTP write lock for gp2 region. When bit 1 is set, the writing of this region's shadow register is blocked. When bit 0 is set, the writing of this region's OTP fuse word is blocked.
11-10 GP1	GP1 Write Lock Status Status of shadow register and OTP write lock for gp2 region. When bit 1 is set, the writing of this region's shadow register is blocked. When bit 0 is set, the writing of this region's OTP fuse word is blocked.
9-8 MAC_ADDR	MAC_ADDR Write Lock Status Status of shadow register and OTP write lock for mac_addr region. When bit 1 is set, the writing of this region's shadow register is blocked. When bit 0 is set, the writing of this region's OTP fuse word is blocked.
7 —	Reserved
6 SJC_RESP	SJC_RESP Lock Status Status of shadow register read and write, OTP read and write lock for sjc_resp region. 0b - The writing or reading of this region's shadow register and OTP fuse word are not blocked. 1b - When set, the writing of this region's shadow register and OTP fuse word are blocked. The read of this region's shadow register and OTP fuse word are also blocked
5-4 —	Reserved
3-2 BOOT_CFG	BOOT_CFG Write Lock Status Status of shadow register and OTP write lock for boot_cfg region. When bit 1 is set, the writing of this region's shadow register is blocked. When bit 0 is set, the writing of this region's OTP fuse word is blocked.
1-0 —	Reserved

23.6.1.11 Value of OTP Bank0 Word1 (Configuration and Manufacturing Info.) (CFG0)

23.6.1.11.1 Offset

Register	Offset
CFG0	410h

23.6.1.11.2 Function

Shadowed memory mapped access to OTP Bank 0, word 1 (ADDR = 0x01).

Copied from the OTP automatically after reset. Can be re-loaded by setting CTRL[RELOAD_SHADOWS]

23.6.1.11.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BITS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

23.6.1.11.4 Fields

Field	Function
31-0 BITS	BITS This register reflects the value of OTP Bank 0, word 1 (ADDR = 0x01). See the Fusemap Descriptions Table in the Fusemap chapter for more fuse details.

23.6.1.12 Value of OTP Bank0 Word2 (Configuration and Manufacturing Info.) (CFG1)

23.6.1.12.1 Offset

Register	Offset
CFG1	420h

23.6.1.12.2 Function

Shadowed memory mapped access to OTP Bank 0, word 2 (ADDR = 0x02).

Copied from the OTP automatically after reset. Can be re-loaded by setting CTRL[RELOAD_SHADOWS]

23.6.1.12.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BITS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

23.6.1.12.4 Fields

Field	Function
31-0 BITS	BITS This register reflects the value of OTP Bank 0, word 2 (ADDR = 0x02). See the Fusemap Descriptions Table in the Fusemap chapter for more fuse details.

23.6.1.13 Value of OTP Bank0 Word3 (Configuration and Manufacturing Info.) (CFG2)

23.6.1.13.1 Offset

Register	Offset
CFG2	430h

23.6.1.13.2 Function

Shadowed memory mapped access to OTP Bank 0, word 3 (ADDR = 0x03).

Copied from the OTP automatically after reset. Can be re-loaded by setting CTRL[RELOAD_SHADOWS]

23.6.1.13.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BITS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

23.6.1.13.4 Fields

Field	Function
31-0 BITS	BITS This register reflects the value of OTP Bank 0, word 3 (ADDR = 0x03). See the Fusemap Descriptions Table in the Fusemap chapter for more fuse details.

23.6.1.14 Value of OTP Bank0 Word4 (Configuration and Manufacturing Info.) (CFG3)

23.6.1.14.1 Offset

Register	Offset
CFG3	440h

23.6.1.14.2 Function

Non-shadowed memory mapped access to OTP Bank 0, word 4 (ADDR = 0x04).

23.6.1.14.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BITS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

23.6.1.14.4 Fields

Field	Function
31-0 BITS	BITS This register reflects the value of OTP Bank 0, word 4 (ADDR = 0x04). See the Fusemap Descriptions Table in the Fusemap chapter for more fuse details.

23.6.1.15 Value of OTP Bank0 Word5 (Configuration and Manufacturing Info.) (CFG4)

23.6.1.15.1 Offset

Register	Offset
CFG4	450h

23.6.1.15.2 Function

Shadowed memory mapped access to OTP Bank 0, word 5 (ADDR = 0x05).

Copied from the OTP automatically after reset. Can be re-loaded by setting CTRL[RELOAD_SHADOWS]

23.6.1.15.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BITS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

23.6.1.15.4 Fields

Field	Function
31-0 BITS	BITS This register reflects the value of OTP Bank 0, word 5 (ADDR = 0x05). See the Fusemap Descriptions Table in the Fusemap chapter for more fuse details.

23.6.1.16 Value of OTP Bank0 Word6 (Configuration and Manufacturing Info.) (CFG5)

23.6.1.16.1 Offset

Register	Offset
CFG5	460h

23.6.1.16.2 Function

Shadowed memory mapped access to OTP Bank 0, word 6 (ADDR = 0x06).

Copied from the OTP automatically after reset. Can be re-loaded by setting CTRL[RELOAD_SHADOWS]

23.6.1.16.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BITS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

23.6.1.16.4 Fields

Field	Function
31-0 BITS	BITS This register reflects the value of OTP Bank 0, word 6 (ADDR = 0x06). See the Fusemap Descriptions Table in the Fusemap chapter for more fuse details.

23.6.1.17 Value of OTP Bank0 Word7 (Configuration and Manufacturing Info.) (CFG6)

23.6.1.17.1 Offset

Register	Offset
CFG6	470h

23.6.1.17.2 Function

Shadowed memory mapped access to OTP Bank 0, word 7 (ADDR = 0x07).

Copied from the OTP automatically after reset. Can be re-loaded by setting CTRL[RELOAD_SHADOWS]

23.6.1.17.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BITS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

23.6.1.17.4 Fields

Field	Function
31-0 BITS	BITS This register reflects the value of OTP Bank 0, word 7 (ADDR = 0x07). See the Fusemap Descriptions Table in the Fusemap chapter for more fuse details.

23.6.1.18 Value of OTP Bank1 Word7 (Analog Info.) (ANA2)

23.6.1.18.1 Offset

Register	Offset
ANA2	4F0h

23.6.1.18.2 Function

Shadowed memory mapped access to OTP Bank 1, word 7 (ADDR = 0x0F).

Copied from the OTP automatically after reset. Can be re-loaded by setting CTRL[RELOAD_SHADOWS]

23.6.1.18.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BITS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

23.6.1.18.4 Fields

Field	Function
31-0 BITS	BITS This register reflects the value of OTP Bank 1, word 7 (ADDR = 0x0F). See the Fusemap Descriptions Table in the Fusemap chapter for more fuse details.

23.6.1.19 Value of OTP Bank4 Word0 (Secure JTAG Response Field) (SJC_RESP0)

23.6.1.19.1 Offset

Register	Offset
SJC_RESP0	600h

23.6.1.19.2 Function

Shadowed memory mapped access to OTP Bank 4, word 0 (ADDR = 0x20).

Copied from the OTP automatically after reset. Can be re-loaded by setting CTRL[RELOAD_SHADOWS]

23.6.1.19.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BITS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

23.6.1.19.4 Fields

Field	Function
31-0 BITS	BITS This register reflects the value of OTP Bank 4, word 0 (ADDR = 0x20). See the Fusemap Descriptions Table in the Fusemap chapter for more fuse details.

23.6.1.20 Value of OTP Bank4 Word1 (Secure JTAG Response Field) (SJC_RESP1)

23.6.1.20.1 Offset

Register	Offset
SJC_RESP1	610h

23.6.1.20.2 Function

Shadowed memory mapped access to OTP Bank 4, word 1 (ADDR = 0x21).

Copied from the OTP automatically after reset. Can be re-loaded by setting CTRL[RELOAD_SHADOWS]

23.6.1.20.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BITS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

23.6.1.20.4 Fields

Field	Function
31-0 BITS	BITS This register reflects the value of OTP Bank 4, word 1 (ADDR = 0x21). See the Fusemap Descriptions Table in the Fusemap chapter for more fuse details.

23.6.1.21 Value of OTP Bank4 Word2 (MAC Address) (MAC0)

23.6.1.21.1 Offset

Register	Offset
MAC0	620h

23.6.1.21.2 Function

Shadowed memory mapped access to OTP Bank 4, word 2 (ADDR = 0x22).

Copied from the OTP automatically after reset. Can be re-loaded by setting CTRL[RELOAD_SHADOWS]

23.6.1.21.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BITS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

23.6.1.21.4 Fields

Field	Function
31-0 BITS	BITS This register reflects the value of OTP Bank 4, word 2 (ADDR = 0x22). See the Fusemap Descriptions Table in the Fusemap chapter for more fuse details.

23.6.1.22 Value of OTP Bank4 Word3 (MAC Address) (MAC1)

23.6.1.22.1 Offset

Register	Offset
MAC1	630h

23.6.1.22.2 Function

Shadowed memory mapped access to OTP Bank 4, word 3 (ADDR = 0x23).

Copied from the OTP automatically after reset. Can be re-loaded by setting CTRL[RELOAD_SHADOWS]

23.6.1.22.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BITS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

23.6.1.22.4 Fields

Field	Function
31-0 BITS	BITS This register reflects the value of OTP Bank 4, word 3 (ADDR = 0x23). See the Fusemap Descriptions Table in the Fusemap chapter for more fuse details.

23.6.1.23 Value of OTP Bank4 Word4 (MAC2 Address) (MAC2)

23.6.1.23.1 Offset

Register	Offset
MAC2	640h

23.6.1.23.2 Function

Shadowed memory mapped access to OTP Bank 4, word 4 (ADDR = 0x24).

Copied from the OTP automatically after reset. Can be re-loaded by setting CTRL[RELOAD_SHADOWS]

23.6.1.23.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BITS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

23.6.1.23.4 Fields

Field	Function
31-0 BITS	BITS This register reflects the value of OTP Bank 4, word 4 (ADDR = 0x24). See the Fusemap Descriptions Table in the Fusemap chapter for more fuse details.

23.6.1.24 Value of OTP Bank4 Word6 (General Purpose Customer Defined Info) (GP1)

23.6.1.24.1 Offset

Register	Offset
GP1	660h

23.6.1.24.2 Function

Shadowed memory mapped access to OTP Bank 4, word 6 (ADDR = 0x26).

Copied from the OTP automatically after reset. Can be re-loaded by setting CTRL[RELOAD_SHADOWS]

23.6.1.24.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BITS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

23.6.1.24.4 Fields

Field	Function
31-0 BITS	BITS This register reflects the value of OTP Bank 4, word 6 (ADDR = 0x26). See the Fusemap Descriptions Table in the Fusemap chapter for more fuse details.

23.6.1.25 Value of OTP Bank4 Word7 (General Purpose Customer Defined Info) (GP2)

23.6.1.25.1 Offset

Register	Offset
GP2	670h

23.6.1.25.2 Function

Shadowed memory mapped access to OTP Bank 4, word 7 (ADDR = 0x27).

Copied from the OTP automatically after reset. Can be re-loaded by setting CTRL[RELOAD_SHADOWS]

23.6.1.25.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BITS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

23.6.1.25.4 Fields

Field	Function
31-0 BITS	BITS This register reflects the value of OTP Bank 4, word 7 (ADDR = 0x27). See the Fusemap Descriptions Table in the Fusemap chapter for more fuse details.

23.6.1.26 Value of OTP Bank5 Word0 (SW GP1) (SW_GP1)

23.6.1.26.1 Offset

Register	Offset
SW_GP1	680h

23.6.1.26.2 Function

Shadowed memory mapped access to OTP Bank 5, word 0 (ADDR = 0x28).

Copied from the OTP automatically after reset. Can be re-loaded by setting CTRL[RELOAD_SHADOWS]

23.6.1.26.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BITS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

23.6.1.26.4 Fields

Field	Function
31-0 BITS	BITS This register reflects the value of OTP Bank 5, word 0 (ADDR = 0x28). See the Fusemap Descriptions Table in the Fusemap chapter for more fuse details.

23.6.1.27 Value of OTP Bank5 Word5 (Misc Conf) (MISC_CONF0)

23.6.1.27.1 Offset

Register	Offset
MISC_CONF0	6D0h

23.6.1.27.2 Function

Shadowed memory mapped access to OTP Bank 5, word 5 (ADDR = 0x2D).

Copied from the OTP automatically after reset. Can be re-loaded by setting CTRL[RELOAD_SHADOWS]

23.6.1.27.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BITS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

23.6.1.27.4 Fields

Field	Function
31-0 BITS	BITS This register reflects the value of OTP Bank 5, word 5 (ADDR = 0x2D). See the Fusemap Descriptions Table in the Fusemap chapter for more fuse details.

23.6.1.28 Value of OTP Bank5 Word6 (Misc Conf) (MISC_CONF1)

23.6.1.28.1 Offset

Register	Offset
MISC_CONF1	6E0h

23.6.1.28.2 Function

Shadowed memory mapped access to OTP Bank 5, word 6 (ADDR = 0x2E).

Copied from the OTP automatically after reset. Can be re-loaded by setting CTRL[RELOAD_SHADOWS]

23.6.1.28.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BITS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

23.6.1.28.4 Fields

Field	Function
31-0 BITS	BITS This register reflects the value of OTP Bank 5, word 6 (ADDR = 0x2E). See the Fusemap Descriptions Table in the Fusemap chapter for more fuse details.

23.6.1.29 Value of OTP Bank7 Word0 (GP3) (GP30)

23.6.1.29.1 Offset

Register	Offset
GP30	880h

23.6.1.29.2 Function

Shadowed memory mapped access to OTP Bank 7, word 0 (ADDR = 0x38).

Copied from the OTP automatically after reset. Can be re-loaded by setting CTRL[RELOAD_SHADOWS]

23.6.1.29.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BITS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

23.6.1.29.4 Fields

Field	Function
31-0 BITS	BITS This register reflects the value of OTP Bank 7, word 0 (ADDR = 0x38). See the Fusemap Descriptions Table in the Fusemap chapter for more fuse details. Reflects value of OTP Bank 7, word 0 (ADDR = 0x38).

23.6.1.30 Value of OTP Bank7 Word1 (GP3) (GP31)

23.6.1.30.1 Offset

Register	Offset
GP31	890h

23.6.1.30.2 Function

Shadowed memory mapped access to OTP Bank 7, word 1 (ADDR = 0x39).

Copied from the OTP automatically after reset. Can be re-loaded by setting CTRL[RELOAD_SHADOWS]

23.6.1.30.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BITS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

23.6.1.30.4 Fields

Field	Function
31-0 BITS	BITS This register reflects the value of OTP Bank 7, word 1 (ADDR = 0x39). See the Fusemap Descriptions Table in the Fusemap chapter for more fuse details.

23.6.1.31 Value of OTP Bank7 Word2 (GP3) (GP32)

23.6.1.31.1 Offset

Register	Offset
GP32	8A0h

23.6.1.31.2 Function

Shadowed memory mapped access to OTP Bank 7, word 2 (ADDR = 0x3A).

Copied from the OTP automatically after reset. Can be re-loaded by setting CTRL[RELOAD_SHADOWS]

23.6.1.31.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BITS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

23.6.1.31.4 Fields

Field	Function
31-0 BITS	BITS This register reflects the value of OTP Bank 7, word 2 (ADDR = 0x3A). See the Fusemap Descriptions Table in the Fusemap chapter for more fuse details.

23.6.1.32 Value of OTP Bank7 Word3 (GP3) (GP33)

23.6.1.32.1 Offset

Register	Offset
GP33	8B0h

23.6.1.32.2 Function

Shadowed memory mapped access to OTP Bank 7, word 3 (ADDR = 0x3B).

Copied from the OTP automatically after reset. Can be re-loaded by setting CTRL[RELOAD_SHADOWS]

23.6.1.32.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BITS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BITS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

23.6.1.32.4 Fields

Field	Function
31-0 BITS	BITS This register reflects the value of OTP Bank 7, word 3 (ADDR = 0x3B). See the Fusemap Descriptions Table in the Fusemap chapter for more fuse details.

Chapter 24
External Memory Controllers

24.1 Overview

This chip has these external memory interfaces and controllers:

- Smart External Memory Controller (SEMC)
- eSD/eMMC/SDIO Interface
- Flexible Serial Peripheral Interface (FlexSPI)

24.2 Smart External Memory Controller (SEMC) Overview

The SEMC is a multi-standard memory controller optimized for both high-performance and low pin-count. It can support multiple external memories in the same application with shared address and data pins. The interface supported includes SDRAM, NOR Flash, SRAM and NAND Flash, as well as 8080 display interface.

The key features of the SEMC include:

- Support up to 8 memory regions up to 512Mbit per region which can be configured to be memory space of SDRAM, NOR Flash, NAND Flash, SRAM or 8080 display frame buffer
- Chip Select (CS) signals
- Ability to disable SDRAM and NAND controller by separate fuse bit
- Support Address Latch Enable (ALE)
- SDRAM interface
  - Supports both 8-bit, 16-bit modes
  - Up to 512Mb per each Chip Select (CS) and up to 4x CS
- NOR Flash and SRAM interface
  - Supports both 8-bit and 16-bit modes
  - Asynchronous mode
  - Address and Data Multiplexing (ADM)
  - Support NOR Flash program via IPBus configuration

- 8080 display interface
  - Supports both 8/16/24-bit modes
  - Up to 100MHz
  - Without tearing effect support
- NAND Flash
  - 8/16-bit NAND FLASH
  - Asynchronous mode
  - Only supports device which is capable of cache read/write operations
  - Only supports page based operation
- Dynamic I/O sharing which enables multiple external memory device in parallel
- Multi-device access pattern scheduler
  - 8 entries scheduler
  - QoS priority, latency and efficiency adjustable arbitration scheme
- SDRAM access pattern optimization
  - 8 entries command reordering queue
  - QoS priority, latency and efficiency adjustable arbitration scheme

24.3 eMMC/eSD/SDIO

This chip has two Ultra Secured Digital Host Controller (uSDHC) modules for SD/eMMC interface. It provides the interface between the host system and the SD/SDIO/MMC cards.

The key features include:

- Support SD/SDIO standard, up to version 3.0
- Support MMC standard, up to version 4.5
- Support 3.3V and 1.8V operation, but do not support 1.2V operation.
- Supports 1-bit / 4-bit SD and SDIO modes, 1-bit / 4-bit / 8-bit MMC modes
  - Up to 800 Mbps of data transfer for SDIO cards using 4 parallel data lines in SDR mode
  - Up to 400 Mbps of data transfer for SDIO card using 4 parallel data lines in DDR mode
  - Up to 800 Mbps of data transfer for SDXC cards using 4 parallel data lines in SDR mode
  - Up to 400 Mbps of data transfer for SDXC card using 4 parallel data lines in DDR mode
  - Up to 1600 Mbps of data transfer for MMC cards using 8 parallel data lines in SDR mode
  - Up to 800 Mbps of data transfer for MMC cards using 8 parallel data lines in DDR mode

24.4 Quad Serial Peripheral Interface

This chip has one Quad Serial Peripheral Interface block (named FlexSPI) acts as an interface to one or two external serial flash devices, each with up to four bidirectional data lines.

The key features includes:

- Each channel can be configured as 1/2/4-bit operation
- Support both dual-channel or single-channel operation
- Support both SDR mode and DDR mode
- Support up to 166 MHz SDR Mode and 166 MHz DDR Mode (with external Flash device DQS input)
- Support up to 133 MHz SDR Mode and 66 MHz DDR Mode (with internal DQS loopback mode)

Chapter 25
Smart External Memory Controller (SEMC)

25.1 Chip-specific SEMC information

Table 25-1. Reference links to related information

Topic	Related module(s)	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
Interrupts, DMA Events and XBAR Assignments	-	Interrupts, DMA Events and XBAR Assignments

NOTE

- For SEMC_CLKX0, see SEMC_CLK5 in IOMUXC chapter.
- For SEMC_CLKX1, see SEMC_CLK6 in IOMUXC chapter.

25.2 Overview

The SEMC is a multi-standard memory controller optimized for both high-performance and low pin-count. It can support multiple external memories in the same application with shared address and data pins. The interfaces supported include SDRAM, NOR Flash, SRAM, NAND Flash, and 8080 display interface.

25.2.1 Block Diagram

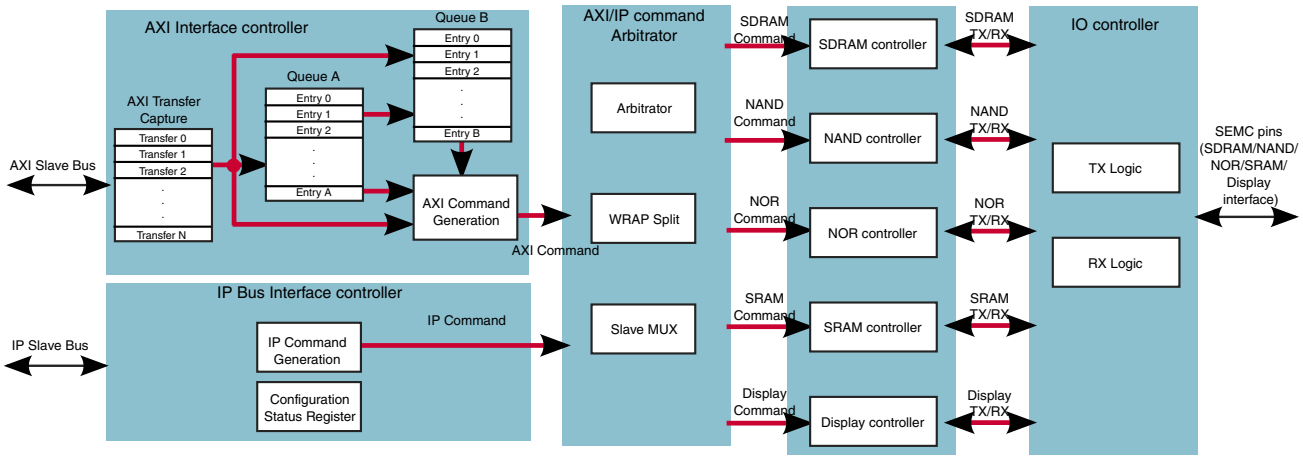


Figure 25-1. SEMC Block Diagram

25.2.2 Features

The SEMC includes the following features:

- Memory space:
  - Supports 9 memory regions:
    - Region #0 for SDRAM CS0 device
    - Region #1 for SDRAM CS1 device
    - Region #2 for SDRAM CS2 device
    - Region #3 for SDRAM CS3 device
    - Region #4 and #8 for NAND device (IP command and AXI command use different region to avoid large region address range on AXI interface)
    - Region #5 for NOR device
    - Region #6 for SRAM device
    - Region #7 for 8080 display frame buffer
  - Each region is configurable for base address, memory size and valid bit
- AXI slave interface
  - Supports 16 outstanding transfers (total including both reads and writes)
    - Supports 8 outstanding write transfers
    - Supports 16 outstanding read transfers
  - Multi-device access pattern optimization with Queue A
    - 8 entries command reordering in Queue A
    - Adjustable arbitration scheme based on QoS priority, latency and efficiency
  - SDRAM access pattern optimization with Queue B

        - 8 entries command reordering in Queue B
        - Adjustable arbitration scheme based on QoS priority, latency and efficiency
    - Memory-mapped AXI interface for read and write access
    - Supports 32/16/8 bit access
- IP Bus interface
    - Internal configuration register access
    - Trigger device access such as device initialization/configuration/read/write (IP command)
- SDRAM interface
    - Supports 8/16 bit modes
    - Supports 4 Chip Select (CS) and each CS up to 512Mb
- NAND Flash interface
    - Supports 8/16 bit modes
    - No hardware ECC
    - Supports device which is capable of cache read/write operations
    - Supports page based read/write operation
    - Supports 32 bit transfer size for AXI write operation
- NOR Flash interface
    - Supports 16 bit mode
    - Supports ADMUX and AADM modes
    - NOR Flash write access is supported only by IP command
    - Up to 512Mb memory size

NOTE

- Up to 512Mb memory size if SEMC_ADDR08 is used as address
- Up to 256Mb memory size if SEMC_ADDR08 is used as chip select

- SRAM interface
    - Supports SRAM and Pseudo SRAM
    - Supports 16 bit mode
    - Supports ADMUX and AADM modes
    - Up to 512Mb memory size

NOTE

- Up to 512Mb memory size if SEMC_ADDR08 is used as address
- Up to 256Mb memory size if SEMC_ADDR08 is used as chip select

- 8080 display interface

    - Supports 8/16 bit modes
    - No tearing effect support
- Misc
    - Dynamic I/O sharing which enables multiple external memory device in parallel
    - Up to 7 Chip Select (CS) signals
    - Supports Address Latch Enable (ALE) for SRAM/NOR device

25.3 Functional Description

The following sections describe functional details of the SEMC module.

25.3.1 Operations

The following sections describe operations of the SEMC module.

25.3.1.1 Modes of Operation

The SEMC supports the following operation modes:

- Module Disable mode

  In module disable mode, the internal clock is gated for low power, but access to configuration and status registers is available.

  This mode is entered by setting MCR[MDIS], and exited by clearing MCR[MDIS]. Software should poll STS0[IDLE] to make sure there are no pending transactions before entering module disable mode.

- Stop mode

  When the system requires the SEMC to enter stop mode, the SEMC waits for all transactions to complete (STS0[IDLE]=1) and returns acknowledge handshake to system. The system can gate off the SEMC clock after the stop acknowledge handshake is returned. There is no clock gating within the SEMC module.

  The SEMC exits this mode immediately when the system stop mode request is negated.

- Normal mode

Access to external devices and internal registers are available in normal mode. The SEMC clock is not gated internally.

25.3.1.2 Device Access by AXI Command

Devices can be accessed by AXI bus access directly. AXI command is the device access sequence triggered by AXI bus access. The device access sequence for AXI commands is described in the following sections.

NOTE

- The AXI bus memory map for each device is defined by Base Registers.
- There is no software configuration or polling needed for AXI command.
- AXI command is the first choice to access device memory. IP command is suitable for other access, such as device register/OTP space access, device initialization, etc.
- IP command can be used to access memory like AXI command, but it is not recommended if AXI command is available. IP command access is low efficiency.

25.3.1.2.1 Bus Master Control Registers (BMCRn) Configuration

BMCRn can be used to balance external memory access efficiency, urgency and latency based on the requirements of a specific application.

25.3.1.2.1.1 BMCR0 Registers Configuration

For Queue A controlled by BMCR0, the arbitration logic adopts a weight based algorithm that assigns each command in the reordering queue with a final score.

The arbitration logic calculates a SCORE for each command based on the formula below, where the weighting factor for each parameter is controlled by the BMCR0 configuration. The command with the highest SCORE is served first.

SCORE = QOS*WQOS + AGE*WAGE/4 + WSH + WRWS

1. QOS stands for AxQOS of AXI bus, and WQOS is the weight factor of QOS.
2. AGE stands for the wait period for each command in queue, and WAGE is the weight factor of AGE.
3. WSH stands for weight of slave hit without read/write switch scenario.
4. WRWS stands for weight of slave hit with read/write switch scenario.

Recommend to set BMCR0 with 0x0 for applications that require restrict sequence of transactions, such as stack is allocated in SDRAM. The write and read transactions to the same address might be reordered if transactions in the queue is reordered, and it might crash the routine. WQOS can be non-zero value if needed, meanwhile WAGE must be non-zero value as well. Anyway, WRWS and WSH should be 0x0.

25.3.1.2.1.2 BMCR1 Registers Configuration

For Queue B controlled by BMCR1, the arbitration logic adopts a weight based algorithm that assigns each command in the reordering queue with a final score.

The arbitration logic calculates SCORE for each SDRAM command based on the formula below. The command with the highest SCORE is served first.

SCORE = QOS*WQOS + AGE*WAGE/4 + WPH + WBR + WRWS

1. QOS stands for AxQOS of AXI bus, and WQOS is the weight factor of QOS.
2. AGE stands for the wait period for each command in queue, and WAGE is the weight factor of AGE.
3. WPH stands for weight of page hit scenario.
4. WBR stands for weight of bank rotation/switch scenario.
5. WRWS stands for weight of slave hit without read/write switch scenario.

Recommend to set BMCR1 with 0x0 for applications that require restrict sequence of transactions, such as stack is allocated in SDRAM. The write and read transactions to the same address might be reordered if transactions in the queue is reordered, and it might crash the routine. WQOS can be non-zero value if needed, meanwhile WAGE must be non-zero value as well. Anyway, WRWS and WSH should be 0x0.

25.3.1.3 Device Access by IP Command

Devices can be accessed (initialized/configured/read/written) by IP bus access also. IP command is the device access sequence triggered by IP bus access. The device access sequence for IP command is described in the following sections.

Device access can be triggered by IP command using the following steps:

1. Set IPCR0/IPCR1/IPCR2/IPTXDAT registers for device address, data size, write mask and write data.
2. Set IPCMD register with valid command code (IPCMD[CMD]) and command key (IPCMD[KEY]).
3. Wait for IP command done (INTR[IPCMDDONE] sets).
4. Read the device data from IPRXDAT register. This is for read access only.

NOTE

- If device address or command code setting is not valid, an IP command error interrupt is generated (INTR[IPCMDERR]) and no device access occurs.
- If another IP command is triggered when the previous IP command has not finished yet, an IP command error interrupt is generated.
- IPCR0/IPCR1/IPCR2/IPTXDAT registers are protected when an IP command is in progress. Any write access to these registers is ignored until the IP command completes.
- The IPCR0 slave address field must be configured even if the memory does not require an address for the selected command (such as WRITE ENABLE command). The SEMC determines the slave device by decoding the device address.
- The slave device is limited to NAND when IPCMD[KEY] is 0x5AA5.

25.3.1.4 SDRAM Controller Operations

This section describes the operations of SDRAM controller.

25.3.1.4.1 SDRAM Address Multiplexing

SDRAMs use a multiplexed address split into rows, columns, and banks. Figure 25-2 shows how the bits of the linear address sent to the SEMC are split up into the row, column, and bank addresses sent to the SDRAM memory device.

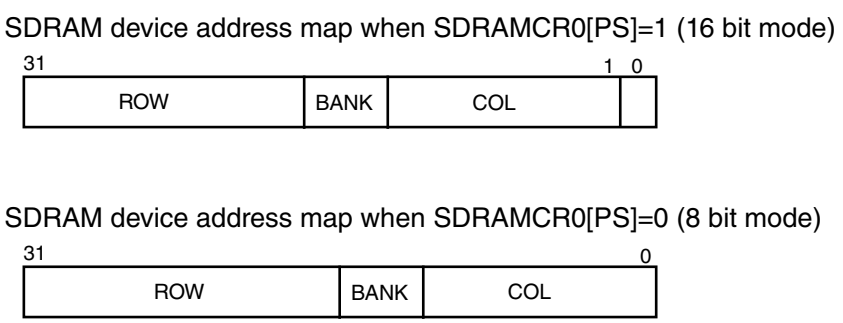


Figure 25-2. SDRAM Device Address Multiplexing

NOTE

- SDRAM address consists of row, bank and column address bits.

- Column address bits offset is bit 0 if SDRAMCR0[PS]=0, bit 1 if SDRAMCR0[PS]=1. Column address bit width is determined by SDRAMCR0[COL] and SDRAMCR0[COL8].
- Bank address bits offset is determined by SDRAMCR0[PS], SDRAMCR0[COL] and SDRAMCR0[COL8]. Bank address bit width is determined by SDRAMCR0[BANK2].
- Row address bits offset is determined by SDRAMCR0[PS], SDRAMCR0[COL], SDRAMCR0[COL8] and SDRAMCR0[BANK2]. Row address bit width is determined by external SDRAM device.
- SDRAM device base address on SoC is configured by BR0, BR1, BR2 and BR3 registers.

25.3.1.4.2 SDRAM Access by IP Command

Table 25-2 lists the SDRAM access types that are triggered by IP commands.

Table 25-2. SDRAM IP Command

Command	Code	Comment
READ	0x8	Reads from the SDRAM memory
WRITE	0x9	Writes to the SDRAM memory
MODE REGISTER SET	0xA	Writes the mode register inside the SDRAM memory
ACTIVE	0xB	Sends a new bank and row address to the memory
AUTO REFRESH	0xC	Auto refresh
SELF REFRESH	0xD	Puts the SDRAM into low power data retention mode. The memory cannot be accessed while in self refresh mode.
PRECHARGE	0xE	Closes the currently active row for the specified bank
PRECHARGE ALL	0xF	Closes the active rows for all banks

25.3.1.4.2.1 READ

When an IP command is triggered with command code - READ, the SEMC will send a READ command to the SDRAM device.

Figure 25-3 shows the sequence for IP command - READ.

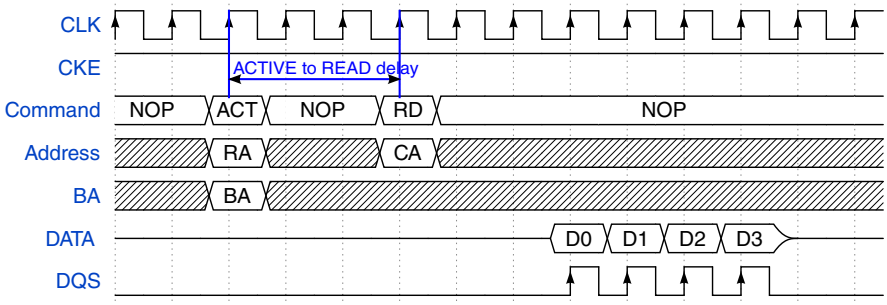


Figure 25-3. IP Command - SDRAM READ

NOTE

- It is supported but not recommended to send READ command with IP command trigger. The SEMC should read SDRAM device by AXI command.
- Read address is from IPCR0 register, read data is sent to IPRXDAT register.
- For this example:
  - ACTIVE to READ/WRITE delay is 3 (SDRAMCR1[ACT2RW]=2)

25.3.1.4.2.2 WRITE

When an IP command is triggered with command code - WRITE, the SEMC will send a WRITE command to the SDRAM device.

Figure 25-4 shows the sequence for IP command - WRITE.

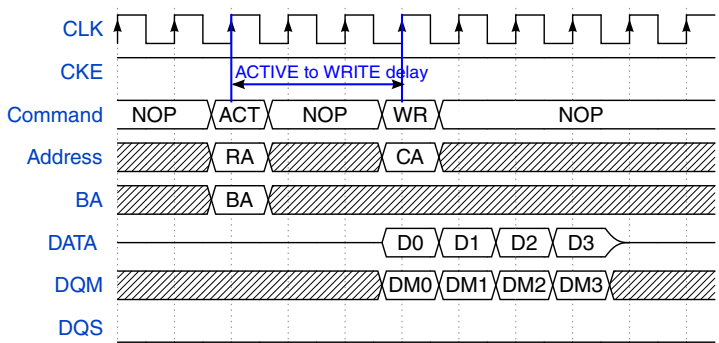


Figure 25-4. IP Command - SDRAM WRITE

NOTE

- It is supported but not recommended to send WRITE command with IP command trigger. The SEMC should write SDRAM device by AXI command.

- Write address is from IPCR0 register, write data is from IPTXDAT register, write mask is from IPCR2 register.
- For this example:
  - ACTIVE to READ/WRITE delay is 3 (SDRAMCR1[ACT2RW]=2)

25.3.1.4.2.3 MODE REGISTER SET

When an IP command is triggered with command code - MODE REGISTER SET, the SEMC will send a MODE REGISTER SET command to the SDRAM device.

Figure 25-5 shows the sequence for IP command - MODE REGISTER SET.

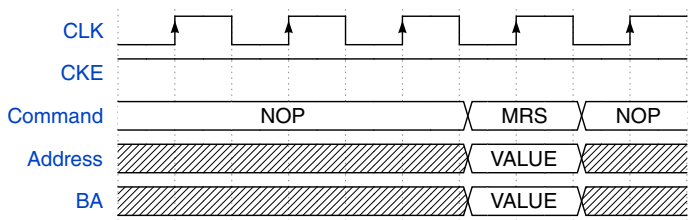


Figure 25-5. IP Command - SDRAM MODE REGISTER SET

NOTE

- MODE REGISTER SET command triggered by IP command should be used for SDRAM device initialization only.
- The "Value" of Address and BA comes from IPTXDAT register.

25.3.1.4.2.4 ACTIVE

When an IP command is triggered with command code - ACTIVE, the SEMC will send an ACTIVE command to the SDRAM device.

Figure 25-6 shows the sequence for IP command - ACTIVE.

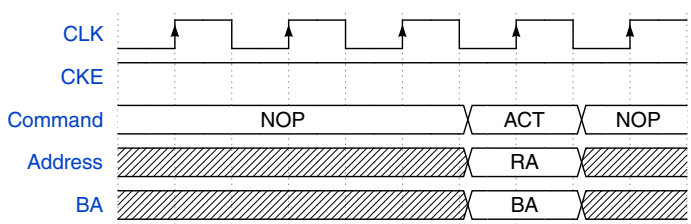


Figure 25-6. IP Command - SDRAM ACTIVE

NOTE

- It is supported but not recommended to send ACTIVE command with an IP command trigger. The SEMC can handle page management for READ/WRITE/AUTO REFRESH/SELF REFRESH etc.

25.3.1.4.2.5 AUTO REFRESH

When an IP command is triggered with command code - AUTO REFRESH, the SEMC will send an AUTO REFRESH command to the SDRAM device.

Figure 25-7 shows the sequence for IP command - AUTO REFRESH.

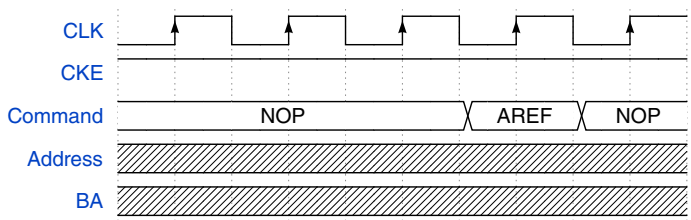


Figure 25-7. IP Command - SDRAM AUTO REFRESH

NOTE

- There is only one AUTO REFRESH command sent when triggered by IP command and only one device is accessed.
- The SEMC sends PRECHARGE ALL command to close any opened page on device before sending AUTO REFRESH command. There is no PRECHARGE ALL command if no pages on the selected device are open.
- AUTO REFRESH command triggered by IP command should be used for SDRAM device initialization only. For entering auto refresh during normal operation, set SDRAMCR3[REN] to 1 and the SEMC will send AUTO REFRESH command automatically.

25.3.1.4.2.6 SELF REFRESH

When a SELF REFRESH IP command is requested for any SDRAM device, the SEMC will send a SELF REFRESH command to all SDRAM devices. After the SELF REFRESH command is completed, all SDRAM devices should be in self refresh mode and the SEMC keeps this mode on SDRAM interface (SEMC_CKE=0 and SEMC_CLK=0).

The SEMC brings all SDRAM devices out of self refresh mode when there is any new IP command or AXI command triggered to an SDRAM device.

Figure 25-8 shows the sequence for self refresh entering and exiting by AXI command.

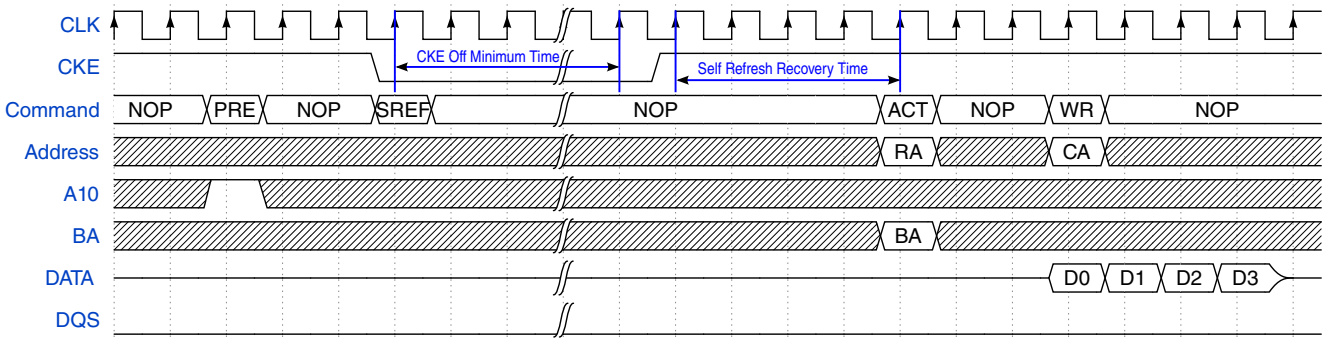


Figure 25-8. IP Command - SDRAM SELF REFRESH

NOTE

- The SEMC sends PRECHARGE ALL command to close any opened page on device before entering self refresh mode. There is no PRECHARGE ALL command if no page is opened at any device.
- SELF REFRESH command sends to all devices.
- It is not recommended to put device into self refresh mode by IP command triggering although it is supported. Recommend to put device into self refresh mode by the SEMC STOP mode instead.
- Disable the SEMC auto refresh enable bit (SDRAMCR3[REN]=0) before entering self refresh mode, otherwise the SEMC may bring devices out of self refresh mode.
- For this example:
  - Self refresh recovery time is 4 clock cycles (SDRAMCR2[SRRC]=3)
  - CKE off minimum time is 4 clock cycles (SDRAMCR1[CKEOFF]=3)
  - The SEMC brings SDRAM out of self refresh mode when an AXI write command is triggered.

25.3.1.4.2.7 PRECHARGE

When an IP command is triggered with command code - PRECHARGE, The SEMC will send a PRECHARGE command to the SDRAM device.

Figure 25-9 shows the sequence for IP command - PRECHARGE.

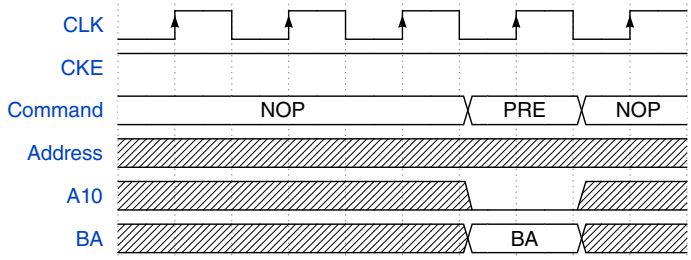


Figure 25-9. IP Command - SDRAM PRECHARGE

NOTE

- It is not recommended to send PRECHARGE command triggered by IP command although it is supported. The SEMC can handle page management for READ/WRITE/ AUTO REFRESH/SELF REFRESH etc.

25.3.1.4.2.8 PRECHARGE ALL

When an IP command is triggered with command code - PRECHARGE ALL, the SEMC will send a PRECHARGE ALL command to the SDRAM device.

Figure 25-10 shows the sequence for IP command - PRECHARGE ALL.

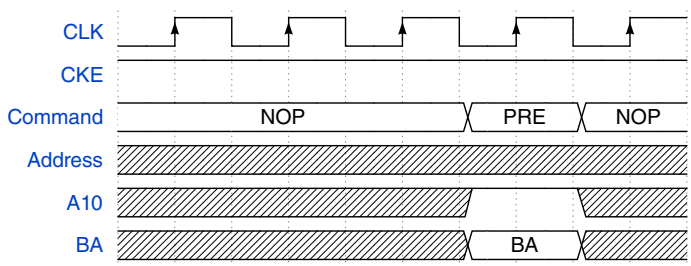


Figure 25-10. IP Command - SDRAM PRECHARGE ALL

NOTE

- It is not recommended to send PRECHARGE ALL command triggered by IP command although it is supported. The SEMC can handle page management for READ/WRITE/AUTO REFRESH/SELF REFRESH etc.
- PRECHARGE ALL command triggered by IP command should be used for SDRAM device initialization only.

25.3.1.4.2.9 Combination of IP Commands

Figure 25-11 shows a combination sequence of IP commands, that includes AUTO REFRESH, ACTIVE, WRITE and PRECHARGE IP commands.

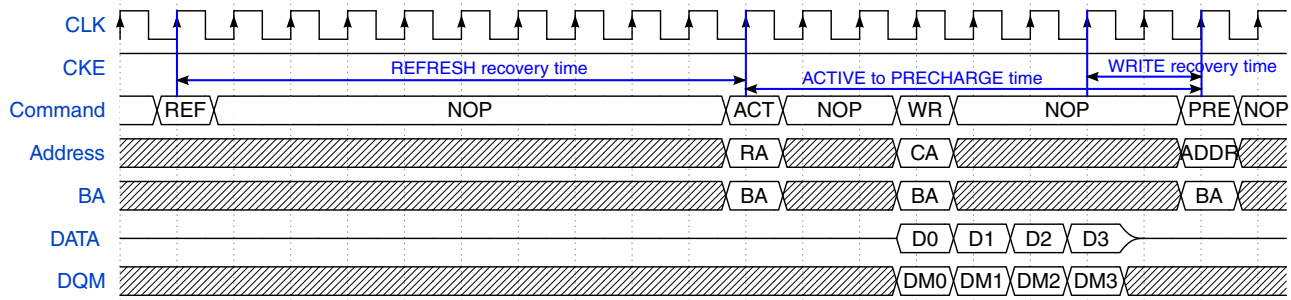


Figure 25-11. IP Command - AUTO REFRESH, ACTIVE, WRITE and PRECHARGE

NOTE

- For this example:
  - REFRESH recovery time is 10 clock cycles (SDRAMCR1[RFRC]=9)
  - WRITE recovery time is 2 clock cycles (SDRAMCR1[WRC]=1)
  - ACTIVE to PRECHARGE minimum time is 6 clock cycles (SDRAMCR1[ACT2PRE]=5). Current ACTIVE to PRECHARGE time is 8 clock cycles by the limitation of WRITE burst beats and recovery time.

25.3.1.4.3 SDRAM Device Access by AXI Command

SDRAM device read/write access can be triggered by AXI Command. For AXI read access, the SEMC will send READ commands to SDRAM device automatically and return data on AXI bus. For AXI write access, the SEMC will save AXI bus write data and send WRITE commands to the SDRAM device automatically

25.3.1.4.3.1 Read Access

Figure 25-12 shows an SDRAM read triggered by an AXI command.

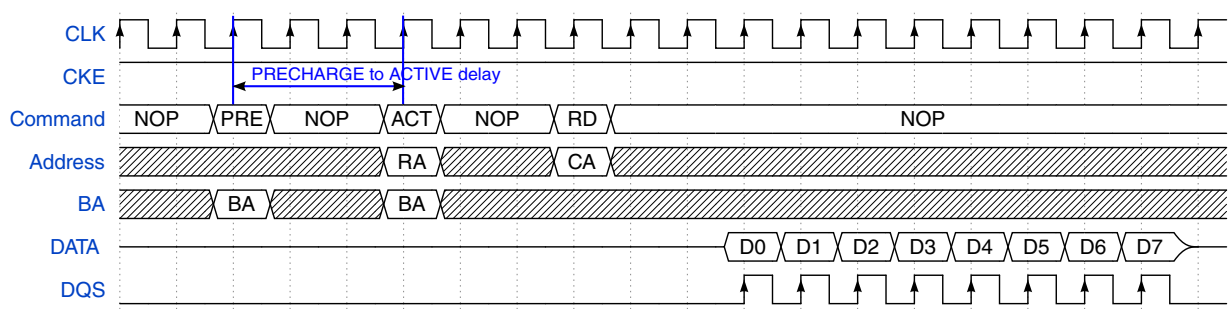


Figure 25-12. AXI Command - SDRAM READ

NOTE

- The SEMC opens/closes pages automatically for current AXI command.
- For this example:
  - CAS latency is 3 (SDRAMCR0[CL]=3)
  - Burst length is 8 (SDRAMCR0[BL]=3)
  - PRECHARGE to ACTIVE delay is 3 clock cycles (SDRAMCR1[PRE2ACT]=2)

25.3.1.4.3.2 Write Access

Figure 25-13 shows an SDRAM write triggered by an AXI command.

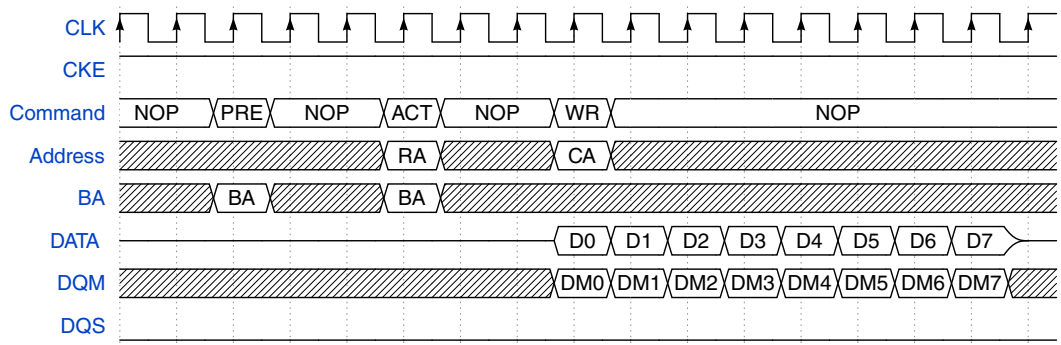


Figure 25-13. AXI Command - SDRAM WRITE

NOTE

- The SEMC opens/closes pages automatically for current AXI command.
- SDRAM device burst length is 8 in this example. The SEMC may send multiple WRITE commands if AXI burst size is larger than SDRAM burst length.

25.3.1.4.3.3 Pipelined Access

Figure 25-14 shows an SDRAM pipelined read access that is triggered by AXI bus read access.

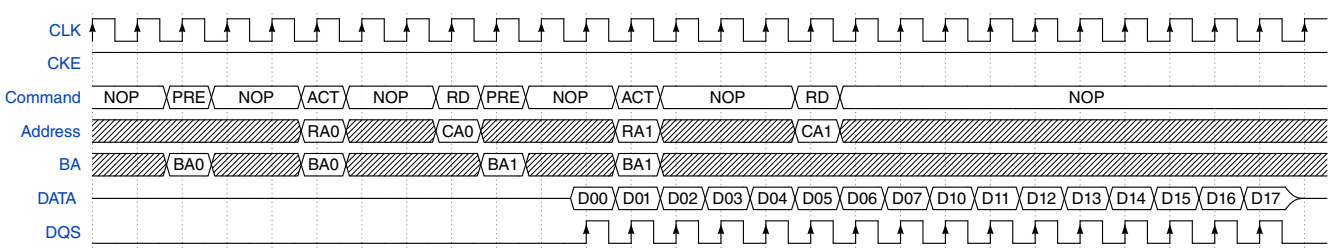


Figure 25-14. AXI Command - Pipelined SDRAM Read

NOTE

- The SEMC opens/closes pages automatically for next AXI command during SDRAM Read burst.
- Data D0x is from CA0, and data D1x is from CA1. This naming rule is also applied to the following SDRAM figures.

Figure 25-15 shows the SDRAM device pipelined write access that triggered by AXI bus write access.

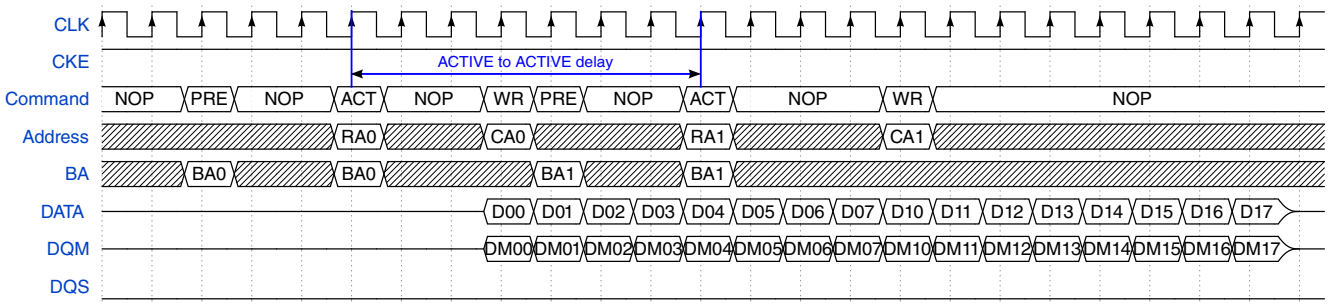


Figure 25-15. AXI Command - Pipelined SDRAM Write

NOTE

- The SEMC opens/closes pages automatically for next AXI command during SDRAM Write burst.

25.3.1.4.3.4 SDRAM Back-to-Back Access

This section describes the SDRAM back-to-back access.

25.3.1.4.3.4.1 Read-to-Read Access

Figure 25-16 shows SDRAM read-to-read access timing when CL=1.

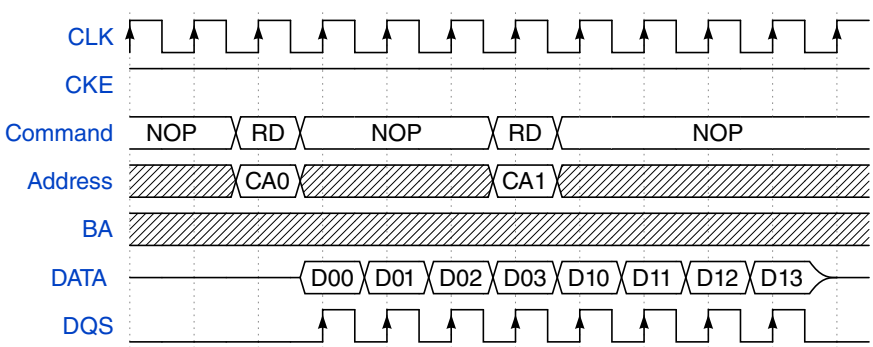


Figure 25-16. SDRAM Read-to-Read Access (CL=1)

Figure 25-17 shows SDRAM read-to-read access timing when CL=2.

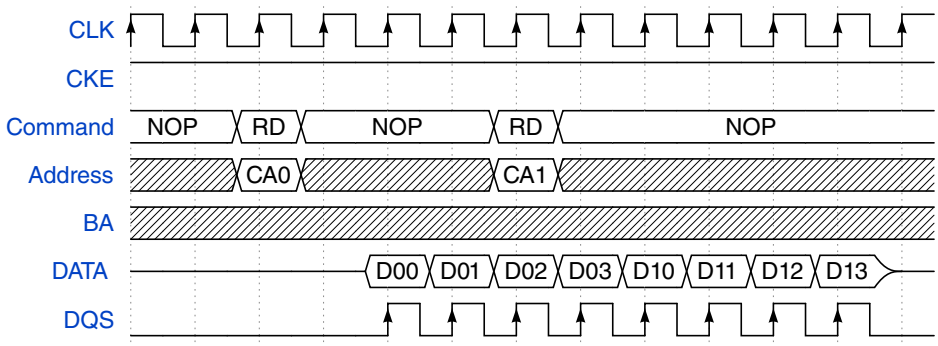


Figure 25-17. SDRAM Read-to-Read Access (CL=2)

Figure 25-18 shows SDRAM read-to-read access timing when CL=3.

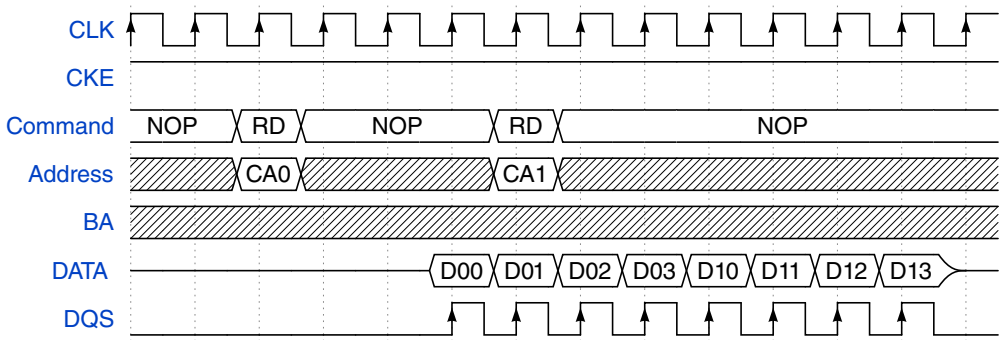


Figure 25-18. SDRAM Read-to-Read Access (CL=3)

25.3.1.4.3.4.2 Read-to-Write Access

Figure 25-19 shows SDRAM read-to-write access timing when CL=1.

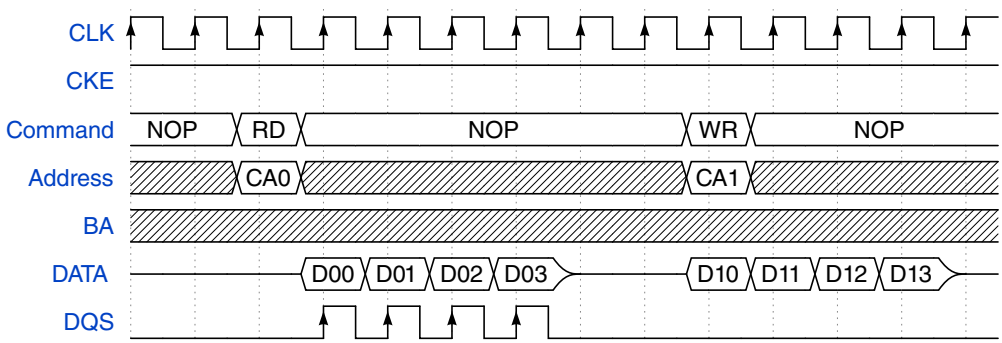


Figure 25-19. SDRAM Read-to-Write Access (CL=1)

Figure 25-20 shows SDRAM read-to-write access timing when CL=2.

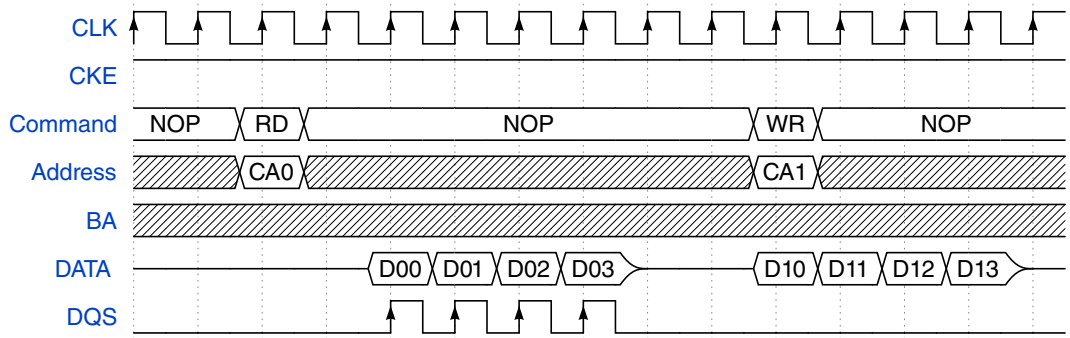


Figure 25-20. SDRAM Read-to-Write Access (CL=2)

Figure 25-21 shows SDRAM read-to-write access timing when CL=3.

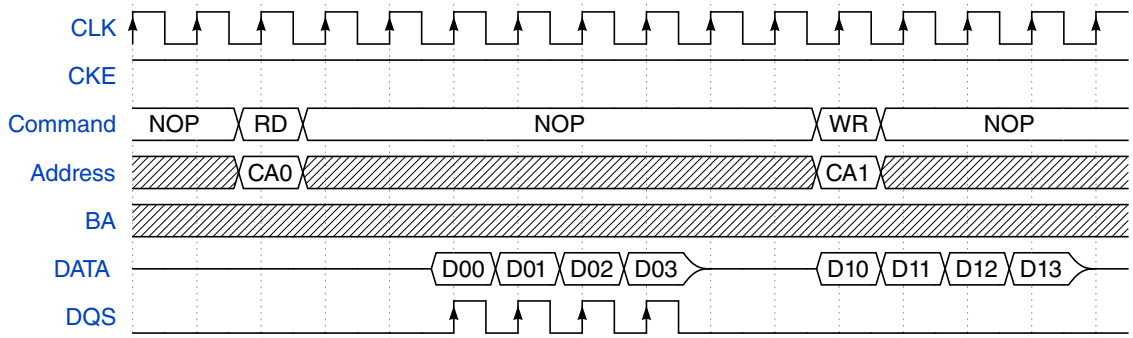


Figure 25-21. SDRAM Read-to-Write Access (CL=3)

25.3.1.4.3.4.3 Write-to-Write Access

Figure 25-22 shows SDRAM write-to-write access timing.

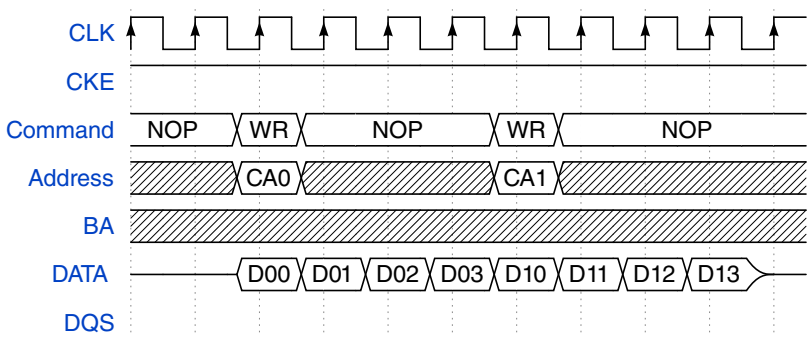


Figure 25-22. SDRAM Write-to-Write Access

25.3.1.4.3.4.4 Write-to-Read Access

Figure 25-23 shows SDRAM write-to-read access timing when CL=1.

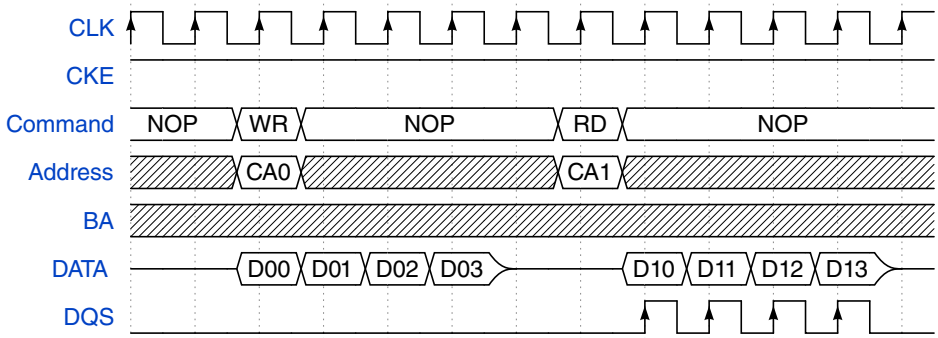


Figure 25-23. SDRAM Write-to-Read Access (CL=1)

Figure 25-24 shows SDRAM write-to-read access timing when CL=2.

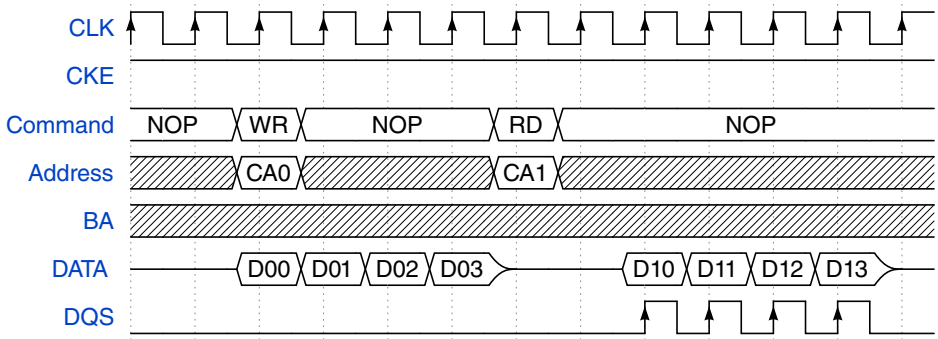


Figure 25-24. SDRAM Write-to-Read Access (CL=2)

Figure 25-25 shows SDRAM write-to-read access timing when CL=3.

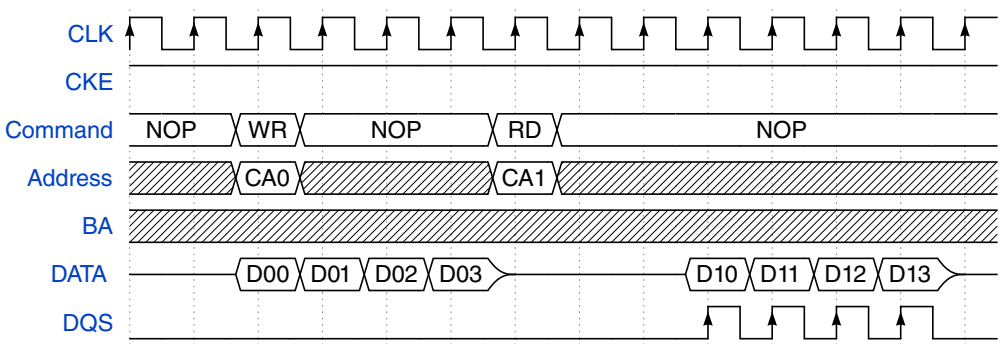


Figure 25-25. SDRAM Write-to-Read Access (CL=3)

25.3.1.4.4 Refresh Command - SDRAM AUTO REFRESH

When SDRAM refresh is enabled (SDRAMCR3[REN]=1), the SEMC will send an AUTO REFRESH command to all SDRAM devices whenever the refresh timer expires. Multiple AUTO REFRESH commands can be sent if refresh burst is not set to one (SDRAMCR3[REBL]!=0).

The refresh timer's count cycle is controlled by UT, RT and PRESCALE bits in SDRAMCR3.

Figure 25-26 shows the sequence of AUTO REFRESH Command.

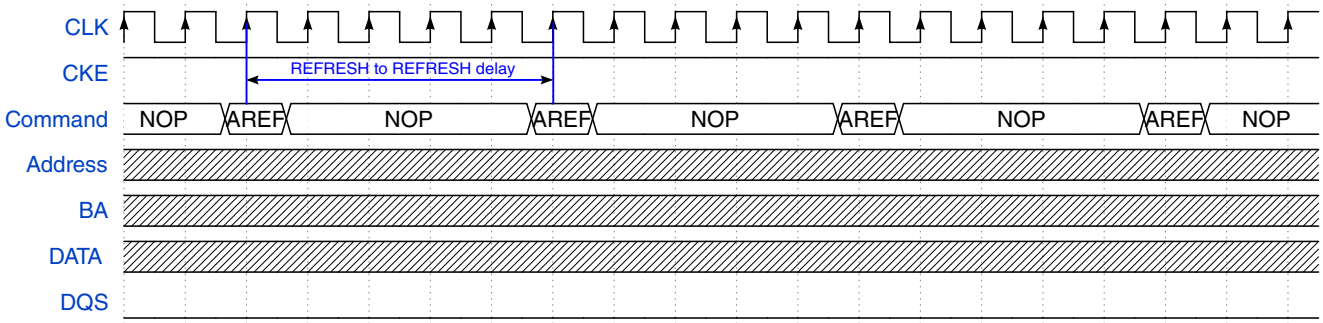


Figure 25-26. Refresh Command - SDRAM AUTO REFRESH

NOTE

- AUTO REFRESH command is sent to all devices.
- For this example:
  - Refresh to refresh delay is 5 clock cycles (SDRAMCR2[REF2REF]=4)
  - Refresh burst length is 4 (SDRAMCR3[REBL]=3)

25.3.1.4.5 SDRAM Low Power Feature

This section describes the low power feature that is implemented for SDRAM.

25.3.1.4.5.1 SDRAM Stop Mode

The SEMC puts SDRAM to self refresh mode automatically when the system enters STOP mode. The SEMC brings SDRAM out of self refresh mode when the system exits STOP mode.

Figure 25-27 shows the sequence of STOP mode entering and exiting.

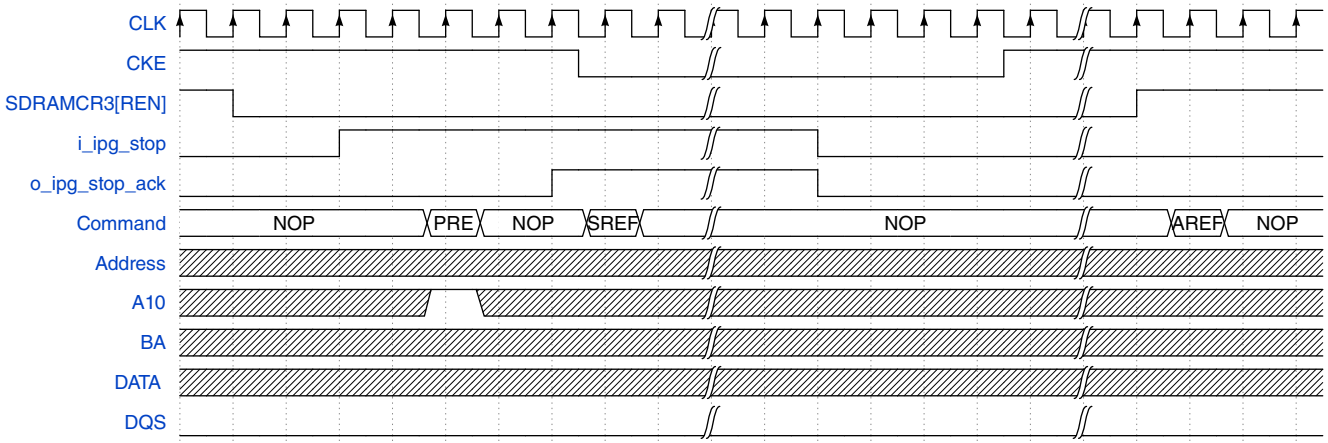


Figure 25-27. SDRAM Stop Mode Enter and Exit

NOTE

- The SEMC sends PRECHARGE ALL command to close any opened page on device before entering self refresh mode. There is no PRECHARGE ALL command if no page is opened at any device.
- SELF REFRESH command is sent to all devices.
- Disable the SEMC auto refresh bit (SDRAMCR3[REN]=0) before entering STOP mode, otherwise the SEMC may bring devices out of self refresh mode.

25.3.1.4.5.2 SDRAM Bus Idle

When there are no pending IP, AXI, or refresh commands, SDRAM interface will in the idle state. When the SDRAM interface bus idle timer expires, the SEMC closes all opened pages on all SDRAM devices. Refer to SDRAMCR2[ITO] for more information.

25.3.1.4.6 SDRAM Page Management

This section describes the SDRAM page management.

The SEMC will save row addresses for up to 8 opened pages: 4 pages for CS0/CS2 (4 banks), 4 pages for CS1/CS3 (4 banks). Device CS0 and CS2 pages are never opened at the same time. Before the SEMC opens pages on CS2, it closes all opened pages on CS0. This is similar for CS1/CS3.

The SEMC closes all pages on all CS by issuing a PRECHARGE ALL command in following cases:

1. SELF REFRESH triggered by IP command
2. SELF REFRESH triggered by stop mode entry

3. The idle time on AXI bus and SDRAM interface is longer than SDRAM idle timeout time (SDRAMCR2[ITO]).

NOTE

The SEMC will not send out PRECHARGE ALL command if there are no pages opened on any bank/CS.

The SEMC will close all pages on one CS in following cases:

1. PRECHARGE ALL command triggered by IP command
2. AUTO REFRESH command triggered by IP command
3. ACTIVE/WRITE/READ command triggered by IP command and CS miss
4. WRITE/READ command triggered by current AXI command and CS miss
5. WRITE/READ command triggered by next AXI command (not accepted yet), CS miss and not same CS as current AXI command

NOTE

One example for CS miss: current IP command access CS2 but there is page opened on CS0 already.

The SEMC will close one page on one CS in following cases:

1. PRECHARGE command triggered by IP command and this bank is opened
2. ACTIVE/READ/WRITE command triggered by IP command, this bank is opened and page missed.
3. WRITE/READ command triggered by current AXI command, this bank is opened and page missed.
4. WRITE/READ command triggered by next AXI command (not accepted yet), this bank is opened, page missed and different bank accessed with current AXI command.

NOTE

One example for page miss: current bank is opened but the opened page row address is different with current SDRAM command access.

The SEMC will open one page on one CS in following cases:

1. ACTIVE/READ/WRITE command triggered by IP command, this bank is not opened.
2. WRITE/READ command triggered by current AXI command, this bank is not opened.
3. WRITE/READ command triggered by next AXI command (not accepted yet), this bank is not opened and different bank accessed with current AXI command.

NOTE

One example for page miss: current bank is opened but the opened page row address is different with current SDRAM command access.

25.3.1.5 NAND Flash Controller Operations

This section describes the operations of NAND Flash controller.

25.3.1.5.1 NAND Flash Address Multiplexing

The NAND Flash controller uses a multiplexed address split into rows and columns. Figure 25-28 shows how the bits of the linear address sent to the SEMC are split up into the row and column addresses sent to the NAND Flash memory device.

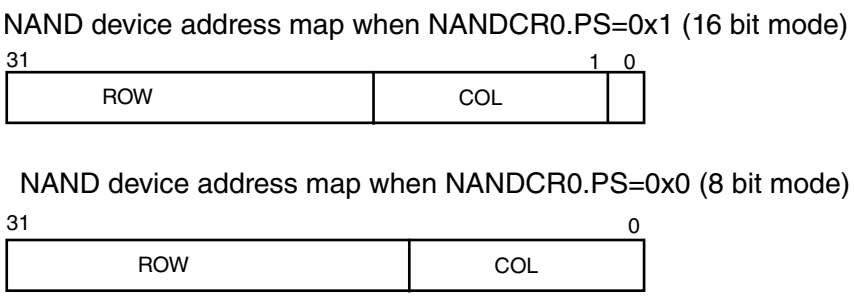


Figure 25-28. NAND Flash Address Multiplexing

NOTE

- NAND address bits consist of row and column address bits.
- Column address bits offset is bit 0 if NANDCR0[PS]=0, the offset is bit 1 if NANDCR0[PS]=1. Column address bit width is determined by NANDCR0[COL].
- Column bit number should cover all page size. For example, column bit number should be 14 if NAND Flash page size is 8192 + 448 (main + spare) bytes.
- Row address bits offset is determined by NANDCR0[PS] and NANDCR[COL].
- NAND device base address on SoC is configured by BR4 and BR8 registers.

25.3.1.5.2 NAND Flash Access Address Type

Table 25-3 lists the NAND address types that are triggered by IP commands.

Table 25-3. NAND Address Types

Address Type	IPCMD[7:4]
Five Bytes Row/Column Address	0x0
One Byte Column Address	0x1
Two Bytes Column Address	0x2
One Byte Row Address	0x3
Two Bytes Row Address	0x4
Three Bytes Row Address	0x5

For IP command, address type is determined by IPCMD register. For AXI command, address type is fixed to "Two Bytes Column Address" because AXI access is limited in opened NAND page.

25.3.1.5.2.1 Five Bytes Row/Column Address

Figure 25-29 shows the Five Bytes Row/Column Address type access timing.

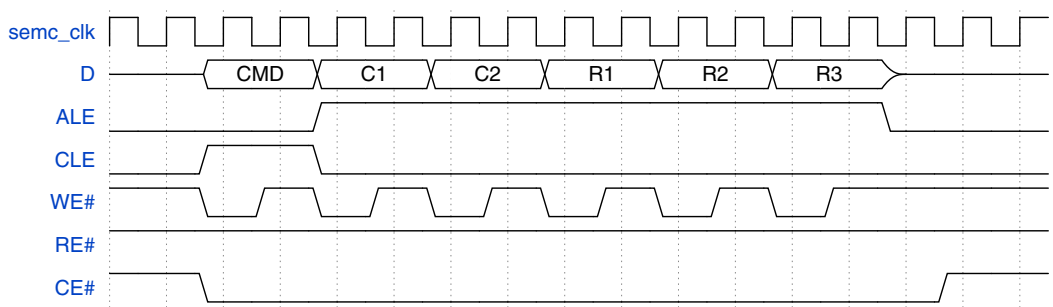


Figure 25-29. NAND Flash Access Address Type - Five Bytes Row/Column Address

There are some option bits to support different NAND address types in the Five Bytes Row/Column Address type. Table 25-4 lists all the combinations of address types.

Table 25-4. Address Bytes Order for five Bytes Row/Column Address Mode

NANDCR3[NDOPT3]	NANDCR3[NDOPT2]	NANDCR3[NDOPT1]	Addres byte order
0	0	0	C1/C2/R1/R2/R3
0	0	1	C1/R1/R2/R3
0	1	0	C1/C2/R1/R2
0	1	1	C1/R1/R2
1	-	0	C1/C2/R1
1	-	1	C1/R1

25.3.1.5.2.2 One Byte Column Address

Figure 25-30 shows the One Byte Column Address type access timing.

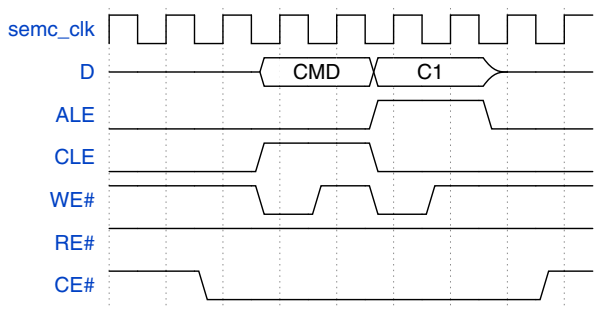


Figure 25-30. NAND Flash Access Address Type - One Byte Column Address

25.3.1.5.2.3 Two Bytes Column Address

Figure 25-31 shows the Two Bytes Column Address type access timing.

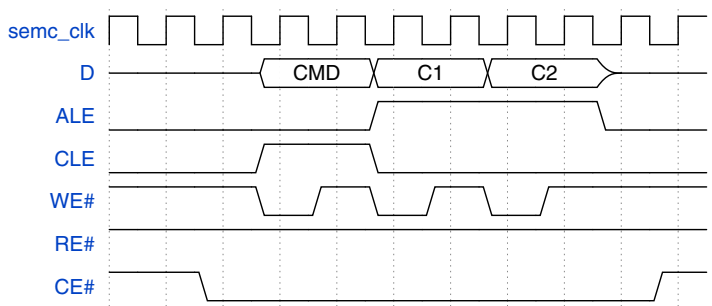


Figure 25-31. NAND Flash Access Address Type - Two Bytes Column Address

25.3.1.5.2.4 One Byte Row Address

Figure 25-32 shows the One Byte Row Address type access timing.

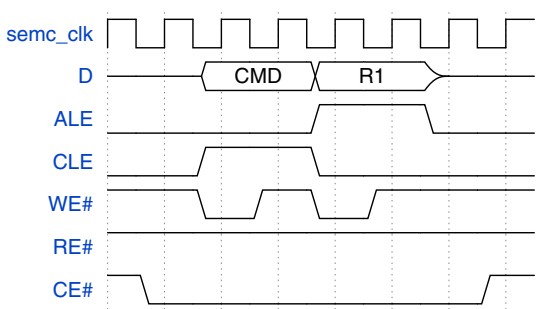


Figure 25-32. NAND Flash Access Address Type - One Byte Row Address

25.3.1.5.2.5 Two Bytes Row Address

Figure 25-33 shows the Two Bytes Row Address type access timing.

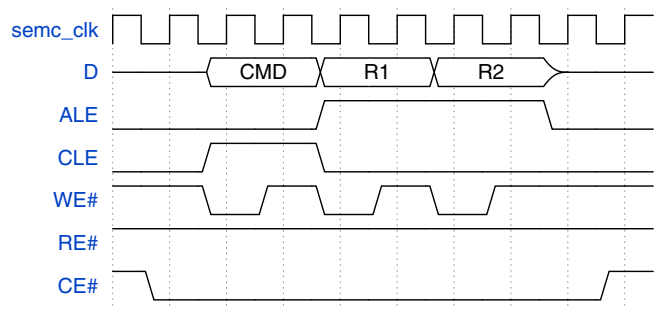


Figure 25-33. NAND Flash Access Address Type - Two Bytes Row Address

25.3.1.5.2.6 Three Bytes Row Address

Figure 25-34 shows the Three Bytes Row Address type access timing.

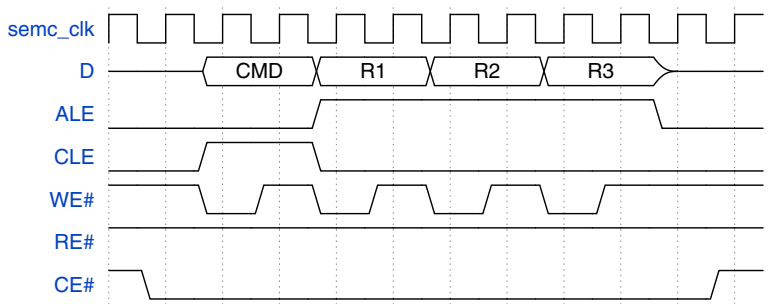


Figure 25-34. NAND Flash Access Address Type - Three Bytes Row Address

25.3.1.5.3 NAND Flash Access Command Type

NAND Flash access support following command types:

1. Command Phase
2. Command + Wait Phase
3. Command + Address Phase
4. Command + Address + Wait Phase
5. Command + Address + Read Phase
6. Command + Address + Write Phase (for IP command)
7. Command + Write Phase
8. Command + Read Phase
9. Write Phase
10. Read Phase
11. Command + Address + Command + Read Phase (for AXI command - Read)
12. Command + Address + Write Phase (for AXI command - Write)

25.3.1.5.3.1 Command Phase

Figure 25-35 shows the NAND Flash Command Phase access timing.

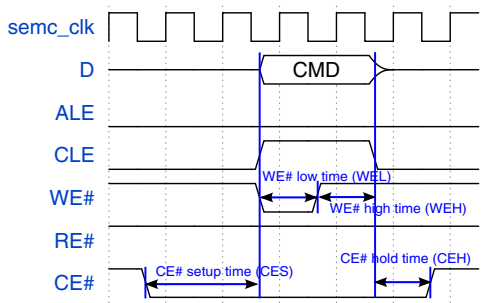


Figure 25-35. NAND Flash Access Command Type - Command Phase

NOTE

- semc_clk is the SEMC functional clock.
- For this example:
  - CE# setup time is 2 clock cycles (NANDCR1[CES]=1)
  - WE# low time is 1 clock cycle (NANDCR1[WEL]=0)
  - WE# high time is 1 clock cycle (NANDCR1[WEH]=0)
  - CE# hold time is 1 clock cycle (NANDCR1[CEH]=0)

25.3.1.5.3.2 Command + Wait Phase

Figure 25-36 shows NAND Flash Command + Wait Phase access timing

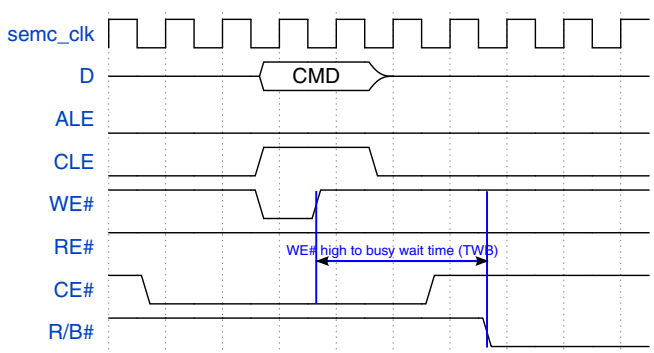


Figure 25-36. NAND Flash Access Command Type - Command + Wait Phase

NOTE

- For this example:
  - WE# high to busy time is 3 clock cycles (NANDCR1[TWB]=2)

25.3.1.5.3.3 Command + Address Phase

Figure 25-37 shows the NAND Flash Command + Address Phase access timing.

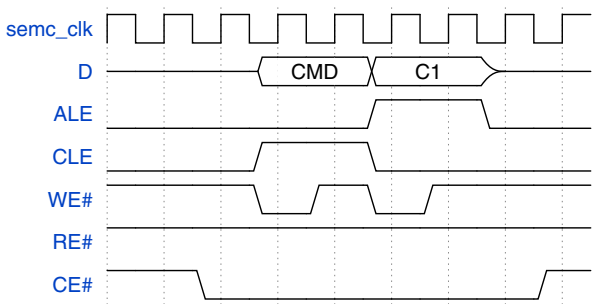


Figure 25-37. NAND Flash Access Command Type - Command + Address Phase

25.3.1.5.3.4 Command + Address + Wait Phase

Figure 25-38 shows NAND Flash Command + Address + Wait Phase access timing.

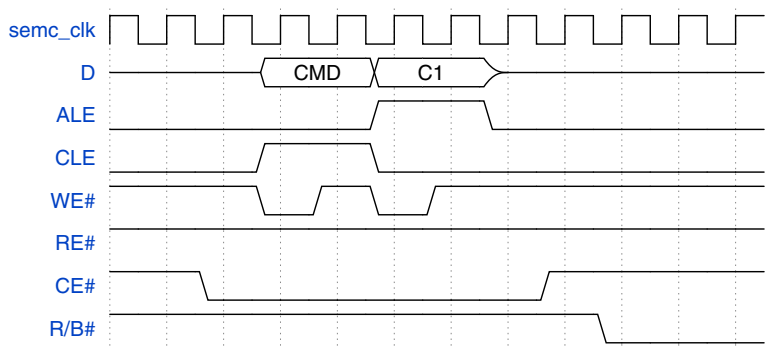


Figure 25-38. NAND Flash Access Command Type - Command + Address + Wait Phase

25.3.1.5.3.5 Command + Address + Read Phase

Figure 25-39 shows NAND Flash Command + Address + Read Phase access timing.

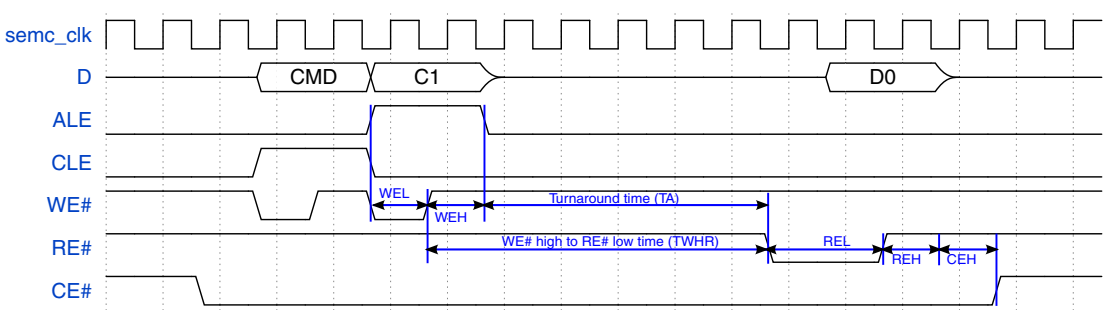


Figure 25-39. NAND Flash Access Command Type - Command + Address + Read Phase

NOTE

- For this example:
  - WE# low time is 1 clock cycle (NANDCR1[WEL]=0)
  - WE# high time is 1 clock cycle (NANDCR1[WEH]=0)
  - RE# low time is 2 clock cycles (NANDCR1[REL]=1)
  - RE# high time is 1 clock cycle (NANDCR1[REH]=0)
  - Turnaround time is 5 clock cycles (NANDCR1[TA]=4)
  - WE# high to RE# low time is 6 clock cycles (NANDCR2[TWHR]=5)
  - CE# hold time is 1 clock cycle (NANDCR1[CEH]=0)

25.3.1.5.3.6 Command + Address + Write Phase (for IP Command)

Figure 25-40 shows NAND Flash Command + Address + Write Phase access timing.

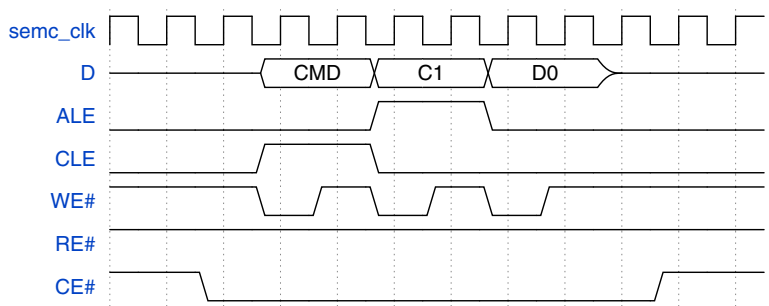


Figure 25-40. NAND Flash Access Command Type - Command + Address + Write Phase

25.3.1.5.3.7 Command + Read Phase

Figure 25-41 shows NAND Flash Command + Read Phase access timing.

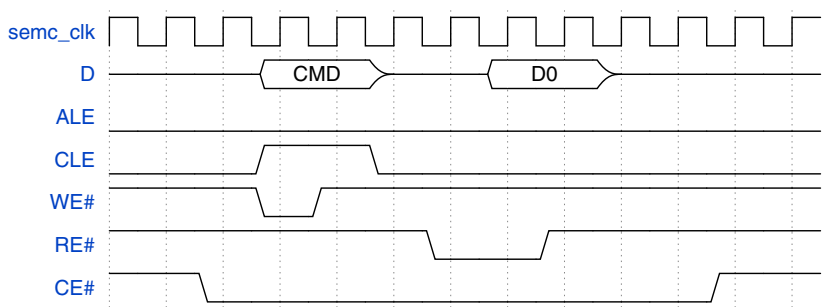


Figure 25-41. NAND Flash Access Command Type - Command + Read Phase

25.3.1.5.3.8 Command + Write Phase

Figure 25-42 shows NAND Flash Command + Write Phase access timing.

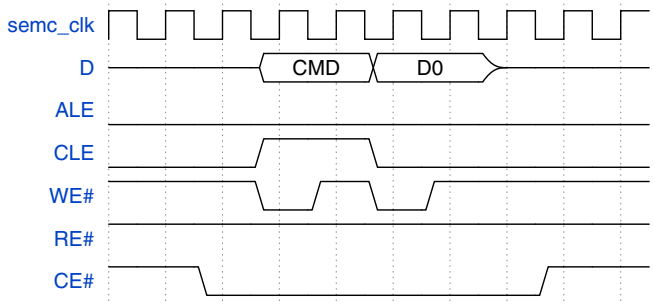


Figure 25-42. NAND Flash Access Command Type - Command + Write Phase

25.3.1.5.3.9 Read Phase

Figure 25-43 shows NAND Flash Read Phase access timing.

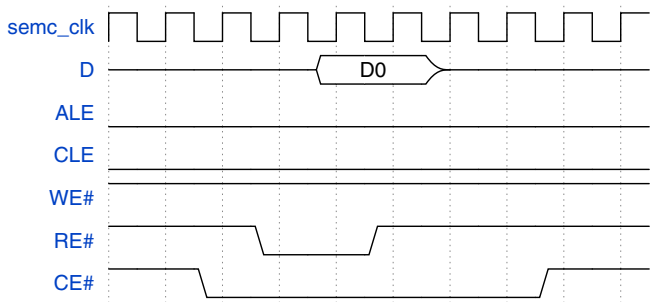


Figure 25-43. NAND Flash Access Command Type - Read Phase

25.3.1.5.3.10 Write Phase

Figure 25-44 shows the timing of NAND Flash Write Phase access.

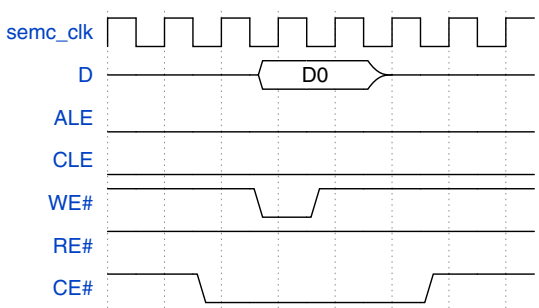


Figure 25-44. NAND Flash Access Command type - Write Phase

25.3.1.5.3.11 Command + Address + Command + Read Phase

Figure 25-45 shows the NAND Flash Command + Address + Command + Read Phase access timing.

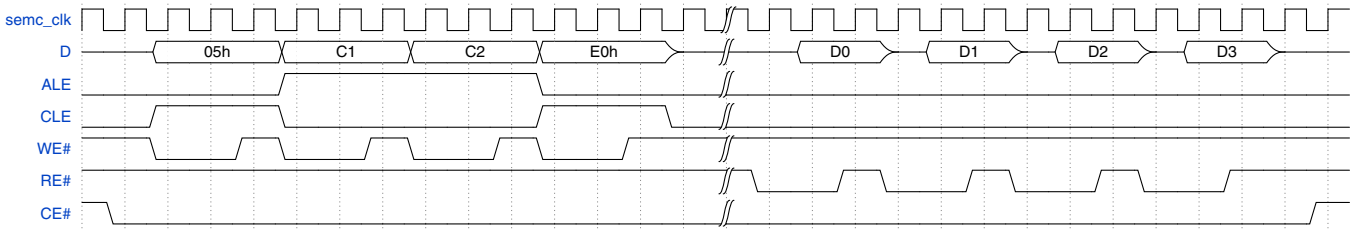


Figure 25-45. NAND Flash Access Command Type - Command + Address + Command + Read Phase

NOTE

- This command type is for AXI command only
- The command code in this command type is fixed value (Change Column Address) to support ONFI NAND device. The first byte command code is 05h, and secend byte command code is E0h.
- The address type in this command type is fixed to two byte column address.

25.3.1.5.3.12 Command + Address + Write Phase (for AXI command)

Figure 25-46 shows the NAND Flash Command + Address + Write Phase access timing.

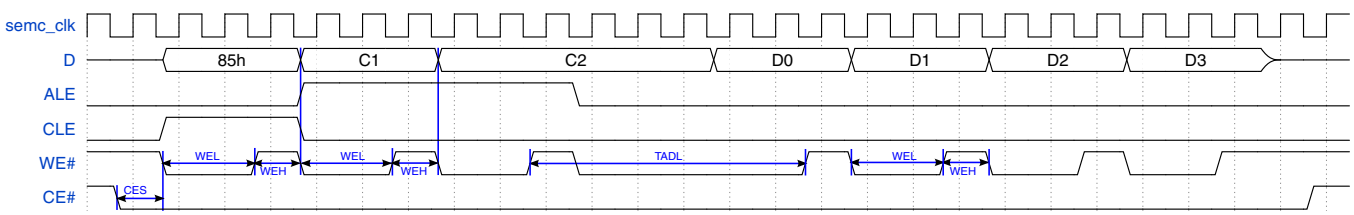


Figure 25-46. NAND Flash Access Command Type - Command + Address + Write Phase

NOTE

- This command type is for AXI command only
- The command code in this command type is fixed value (85h - Change Write Column Address) to support ONFI NAND device.
- The address type in this command type is fixed to two byte column address.
- For this example:
  - WE# low time is 2 clock cycles (NANDCR1[WEL]=1)
  - WE# high time is 1 clock cycle (NANDCR1[WEH]=0)
  - Address cycle to data loading time is 6 clock cycles (NANDCR2[TADL]=5)

25.3.1.5.4 NAND Flash Read Access in EDO and non-EDO Mode

NAND Flash support both EDO and non-EDO modes. Figure 25-47 shows the access timing of EDO mode.

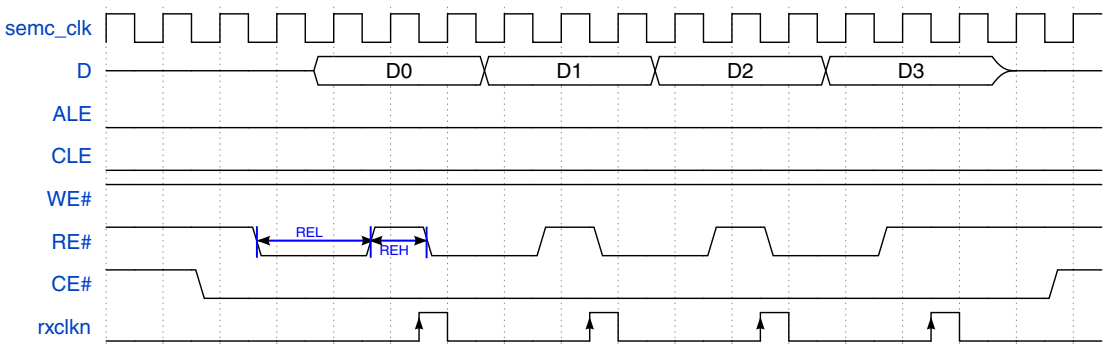


Figure 25-47. NAND Flash Read Acess in EDO mode

NOTE

- rxclkn is the SEMC internal clock to sample read data. It is close to falling edge of RE#.
- For this example:
  - RE# low time is 2 clock cycles (NANDCR1[REL]=1)
  - RE# high time is 1 clock cycle (NANDCR1[REH]=0)

Figure 25-48 shows the access timing of non-EDO mode.

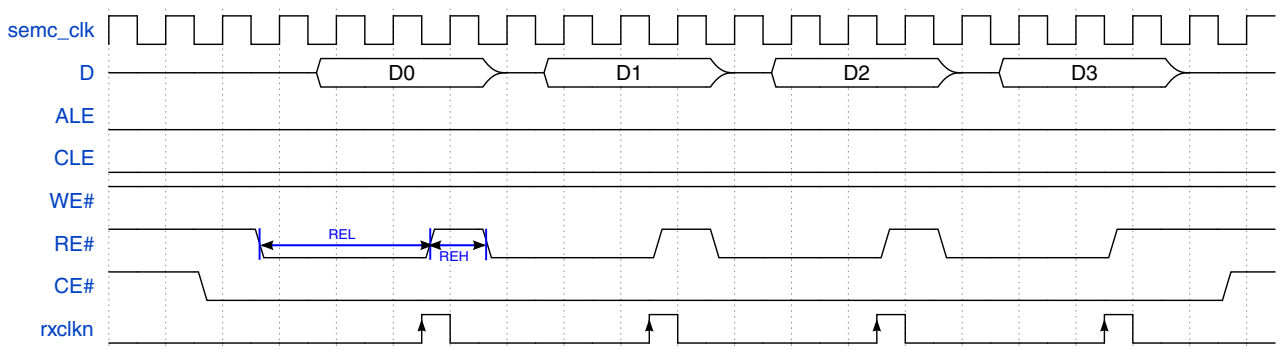


Figure 25-48. NAND Flash Read Acess in non-EDO mode

NOTE

- rxclkn is the SEMC internal clock to sample read data. It is close to rising edge of RE#.
- For this example:
  - RE# low time is 3 clock cycles (NANDCR1[REL]=2)
  - RE# high time is 1 clock cycle (NANDCR1[REH]=0)

25.3.1.5.5 NAND Flash Access in SYNC Mode

NAND Flash supports SYNC mode, the following chapters describes the access timings.

25.3.1.5.5.1 NAND Command + Address + Command + Read Operation in SYNC Mode

Figure 25-49 shows NAND Command + Address + Command + Read operation in SYNC mode.

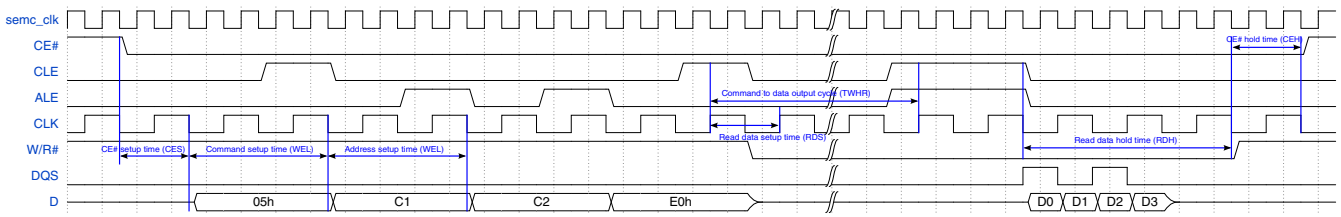


Figure 25-49. NAND Command + Address + Command + Read in SYNC Mode

NOTE

- semc_clk is the SEMC internal functional clock. CLK is the synchronous clock to latch command and address cycles. The frequency of CLK is half of semc_clk. For example, if semc_clk frequency is 200 MHz, the CLK frequency is 100 MHz.
- For this example:
  - CE# setup time is 2 clock cycles (NANDCR1[CES]=1)
  - Command/Address setup time is 4 clock cycles (NANDCR1[WEL]=2)
  - NANDCR3[RDS] and NANDCR2[TWHR] control the timing from command to data read phase. In general, NANDCR3[RDS] controls the minimum delay before data read phase. In this figure, NANDCR3[RDS]=0.
  - Read data hold time is 6 clock cycles (NANDCR3[RDH]=4)
  - CE# hold time is 2 clock cycles (NANDCR1[CEH]=1)

25.3.1.5.5.2 NAND Command + Address + Write Operation in SYNC Mode

Figure 25-50 shows NAND Command + Address + Write operation in SYNC mode.

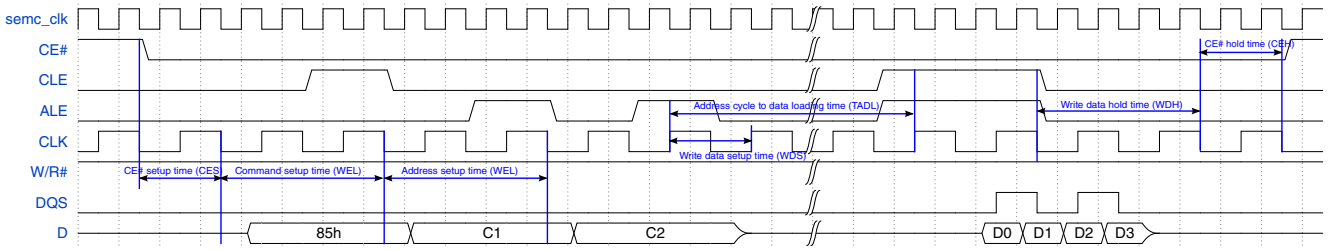


Figure 25-50. NAND Command + Address + Write in SYNC Mode

NOTE

- For this example:
  - CE# setup time is 2 clock cycles (NANDCR1[CES]=1)
  - Command/Address setup time is 4 clock cycles (NANDCR1[WEL]=2)
  - NANDCR3[WDS] and NANDCR2[TADL] control the timing from command to data write phase. In general, NANDCR3[WDS] controls the minimum delay before data write phase. In this figure, NANDCR3[WDS]=0.
  - Write data hold time is 4 clock cycles (NANDCR3[WDH]=2)
  - CE# hold time is 2 clock cycles (NANDCR1[CEH]=1)

25.3.1.6 NOR Flash Controller Operations

This section describes the operations of NOR Flash controller.

25.3.1.6.1 NOR Flash Address Multiplexing

NOR Flash use a multiplexed address split into rows and columns. Figure 25-51 shows how the bits of the linear address sent to the SEMC are split up into the row and column addresses sent to the NOR Flash memory device.

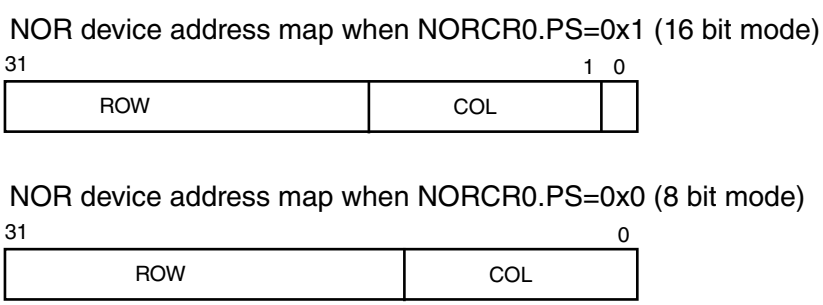


Figure 25-51. NOR Flash Address Multiplexing

NOTE

- NOR address bits consist of row and column address bits.
- Column address bits offset is bit 0 if NORCR0[PS]=0 and bit 1 if NORCR0[PS]=1. Column address bit width is determined by NORCR0[COL].
- Row address bits offset is determined by NORCR0[PS] and NORCR0[COL].
- NOR device base address on SoC is configured by BR5 register.

25.3.1.6.2 NOR Flash Read Operation in ASYNC Mode

NOR Flash controller supports following address modes, which setting by NORCR0[AM].

Figure 25-52 shows the ASYNC NOR Flash read in ADMUX address mode.

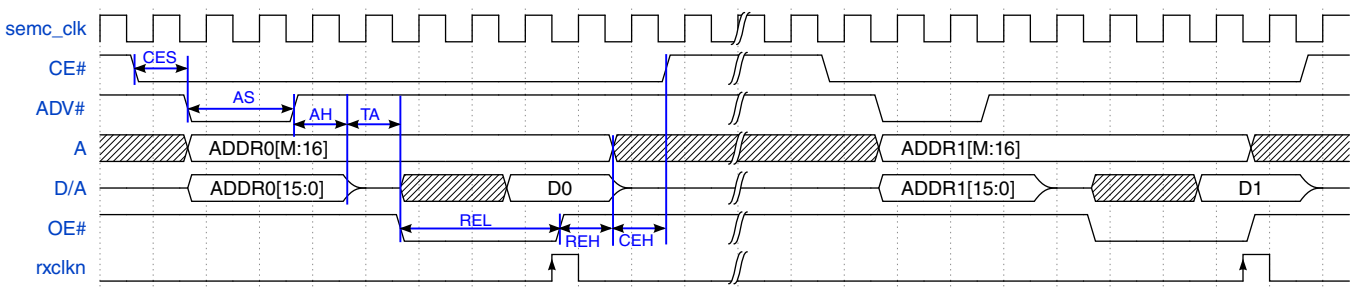


Figure 25-52. ASYNC NOR Flash Read in ADMUX Address Mode

NOTE

- semc_clk and rxclkn are the internal signals. semc_clk is the SEMC functional clock. rxclkn is the clock to sample data from NOR Flash.
- For this example:
  - CE# setup time is 1 clock cycle (NORCR1[CES]=0)
  - Address setup time is 2 clock cycles (NORCR1[AS]=1)
  - Address hold time is 1 clock cycle (NORCR1[AH]=0)
  - Turnaround time is 1 clock cycle (NORCR2[TA]=0)
  - OE# low time is 3 clock cycles (NORCR1[REL]=2)
  - OE# high time is 1 clock cycle (NORCR1[REH]=0)
  - CE# hold time is 1 clock cycle (NORCR1[CEH]=0)

Figure 25-53 shows the ASYNC NOR Flash read in AADM address mode.

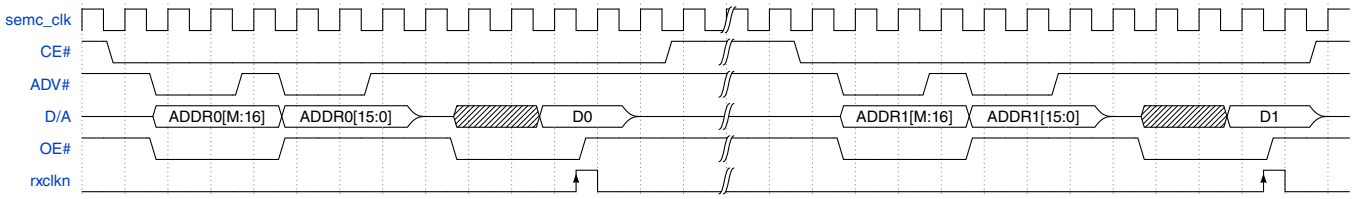


Figure 25-53. ASYNC NOR Flash Read in AADM Address Mode

25.3.1.6.3 NOR Flash Write Operation in ASYNC Mode

Figure 25-54 shows the ASYNC NOR Flash write in ADMUX address mode.

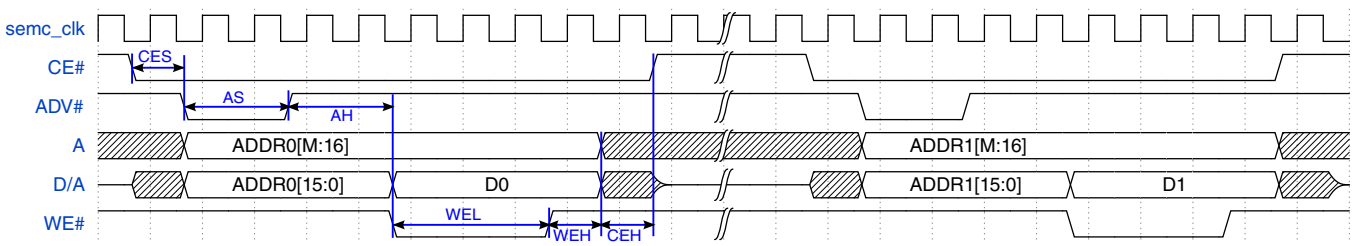


Figure 25-54. ASYNC NOR Flash Write in ADMUX Address Mode

NOTE

- semc_clk is the SEMC functional clock.
- For this example:
  - CE# setup time is 1 clock cycle (NORCR1[CES]=0)
  - Address setup time is 2 clock cycles (NORCR1[AS]=1)
  - Address hold time is 2 clock cycles (NORCR1[AH]=1)
  - WE# low time is 3 clock cycles (NORCR1[WEL]=2)
  - WE# high time is 1 clock cycle (NORCR1[WEH]=0)
  - CE# hold time is 1 clock cycle (NORCR1[CEH]=0)

Figure 25-55 shows the ASYNC NOR Flash write in AADM address mode.

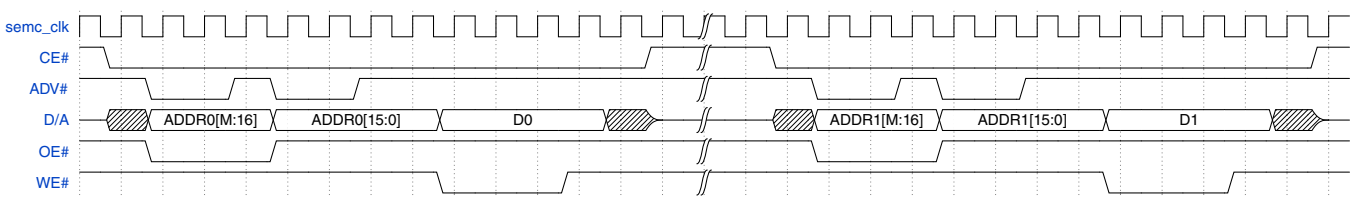


Figure 25-55. ASYNC NOR Flash Write in AADM Address Mode

25.3.1.6.4 NOR Flash Read Operation in SYNC Mode

Figure 25-56 shows the SYNC NOR Flash read in ADMUX address mode.

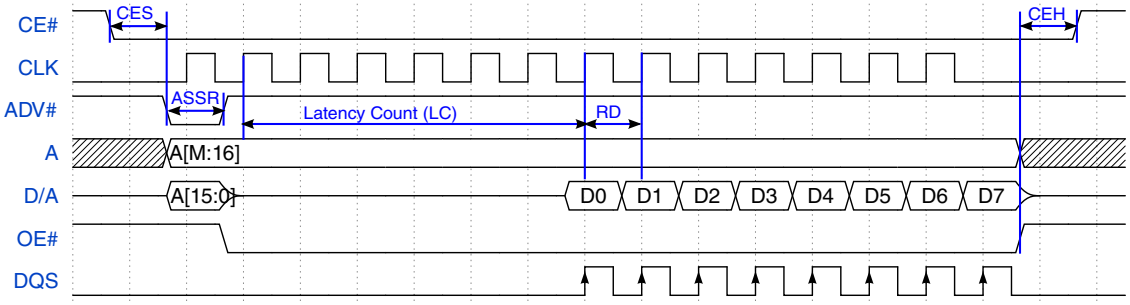


Figure 25-56. SYNC Flash Read in ADMUX Address Mode

NOTE

- DQS is the clock to sample data from NOR Flash. It generally loops back from DQS pad.
- For this example:
  - CE# setup time is 1 clock cycle (NORCR1[CES]=0)
  - Address setup time is 1 clock cycle (NORCR3[ASSR]=0)
  - Latency count is 6 clock cycles (NORCR2[LC]=6)
  - Read time is 1 clock cycle (NORCR2[RD]=0)
  - CE# hold time is 1 clock cycle (NORCR1[CEH]=0)

Figure 25-57 shows the SYNC NOR Flash read in AADM address mode.

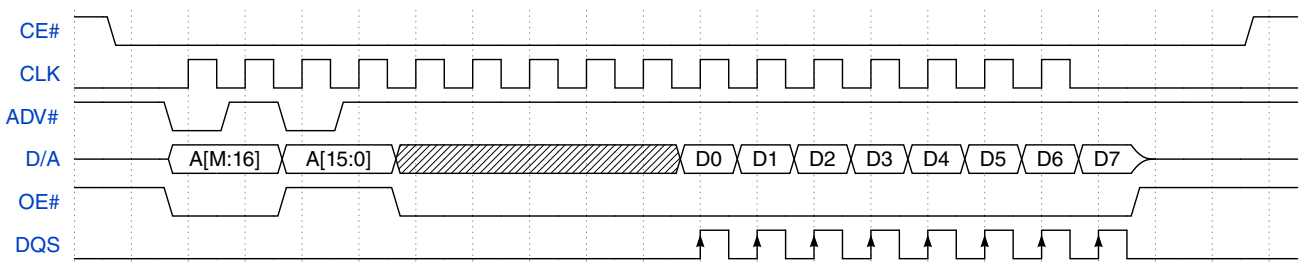


Figure 25-57. SYNC Flash Read in AADM Address Mode

25.3.1.6.5 NOR Flash Write Operation in SYNC Mode

Flash write operation in SYNC mode is same as asynchronous.

25.3.1.7 SRAM Controller Operations

This section describes the operations of SRAM controller.

25.3.1.7.1 SRAM Address Multiplexing

SRAM use a multiplexed address split into rows and columns. Figure 25-58 shows how the bits of the linear address sent to the SEMC are split up into the row and column addresses sent to the SRAM memory device.

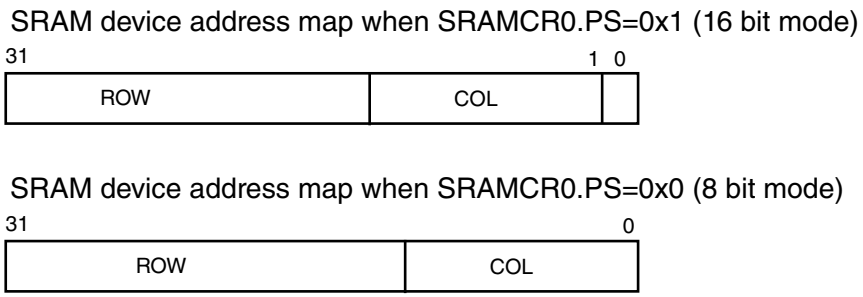


Figure 25-58. SRAM Address Multiplexing

NOTE

- SRAM address bits consist of row and column address bits.
- Column address bits offset is bit 0 if SRAMCR0[PS]=0, offset is bit 1 if SRAMCR0[PS]=1. Column address bit width is determined by SRAMCR0[COL].
- Row address bits offset is determined by SRAMCR0[PS] and SRAMCR0[COL].
- SRAM device base address on SoC is configured by BR6 register.

25.3.1.7.2 SRAM Read Operation in ASYNC Mode

Figure 25-59 shows the ASYNC SRAM read in ADMUX address mode.

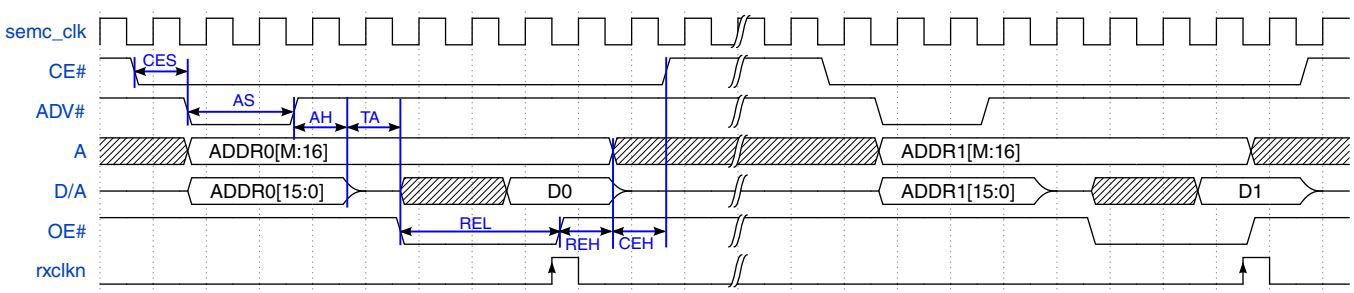


Figure 25-59. ASYNC SRAM Read in ADMUX Address Mode

NOTE

- semc_clk and rxclkn are the internal signals. semc_clk is the SEMC functional clock. rxclkn is the clock to sample data from SRAM.
- For this example:

- CE# setup time is 1 clock cycle (SRAMCR1[CES]=0)
- Address setup time is 2 clock cycles (SRAMCR1[AS]=1)
- Address hold time is 1 clock cycle (SRAMCR1[AH]=0)
- Turnaround time is 1 clock cycle (SRAMCR2[TA]=0)
- OE# low time is 3 clock cycles (SRAMCR1[REL]=2)
- OE# high time is 1 clock cycle (SRAMCR1[REH]=0)
- CE# hold time is 1 clock cycle (SRAMCR1[CEH]=0)

Figure 25-60 shows the ASYNC SRAM read in AADM address mode.

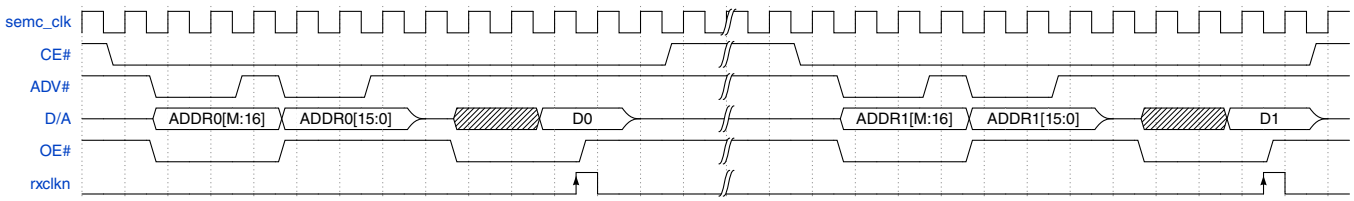


Figure 25-60. ASYNC SRAM Read in AADM Address Mode

25.3.1.7.3 SRAM Write Operation in ASYNC Mode

Figure 25-61 shows the ASYNC SRAM write in ADMUX address mode.

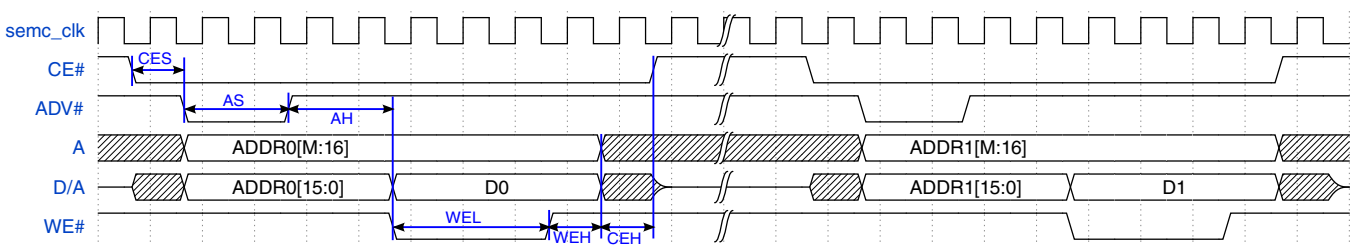


Figure 25-61. ASYNC SRAM Write in ADMUX Address Mode

NOTE

- For this example:
  - CE# setup time is 1 clock cycle (SRAMCR1[CES]=0)
  - Address setup time is 2 clock cycles (SRAMCR1[AS]=1)
  - Address hold time is 2 clock cycles (SRAMCR1[AH]=1)
  - WE# low time is 3 clock cycles (SRAMCR1[WEL]=2)
  - WE# high time is 1 clock cycle (SRAMCR1[WEH]=0)
  - CE# hold time is 1 clock cycle (SRAMCR1[CEH]=0)

Figure 25-62 shows the ASYNC SRAM write in AADM address mode.

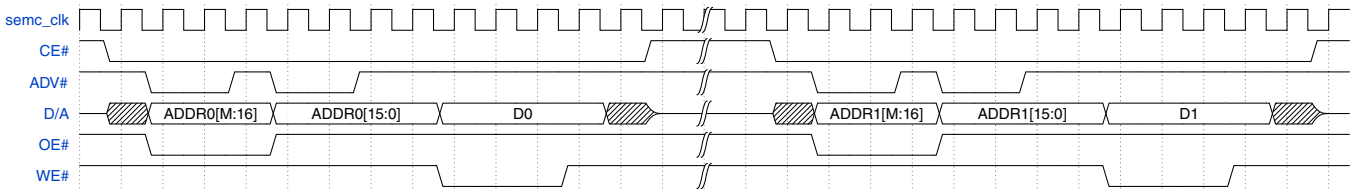


Figure 25-62. ASYNC SRAM Write in AADM Address Mode

25.3.1.7.4 SRAM Register Operation in ASYNC Mode

Figure 25-63 shows the ASYNC SRAM register access in ADMUX address mode.

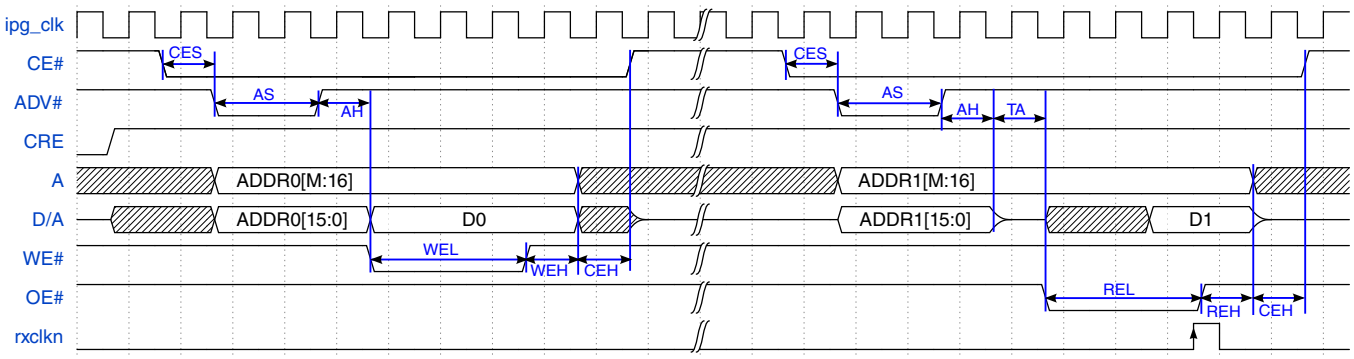


Figure 25-63. ASYNC SRAM Register Access in ADMUX Address Mode

NOTE

- For this example:
  - First operation is SRAM register write access. ADDR0[M:0] contains the value for writing to SRAM register. The value of D0 can be ignored.
  - Second operation is SRAM register read access. ADDR1[M:0] contains the value for choosing of SRAM register. D1 is the value that read from SRAM register.
  - The value of ADDR0/1 comes from IPCR0 register.
    - If BR6[BA]=0x98000 and IPCR0=0x9810883E, ADDR0/1 is 0x8441F when port size is 16bit mode. The address is divided by 2 in 16bit mode.
    - If BR6[BA]=0x98000 and IPCR0=0x9810883E, ADDR0/1 is 0x10883E when port size is 8bit mode.
  - CE# setup time is 1 clock cycle (SRAMCR1[CES]=0)

- Address setup time is 2 clock cycles (SRAMCR1[AS]=1)
- Address hold time is 1 clock cycle (SRAMCR1[AH]=0)
- WE# low time is 3 clock cycles (SRAMCR1[WEL]=2)
- WE# high time is 1 clock cycle (SRAMCR1[WEH]=0)
- CE# hold time is 1 clock cycle (SRAMCR1[CEH]=0)
- Turnaround time is 1 clock cycle (SRAMCR2[TA]=0)
- OE# low time is 3 clock cycles (SRAMCR1[REL]=2)
- OE# high time is 1 clock cycle (SRAMCR1[REH]=0)

25.3.1.7.5 SRAM Read Operation in SYNC Mode

Figure 25-64 shows the SYNC SRAM read in ADMUX address mode.

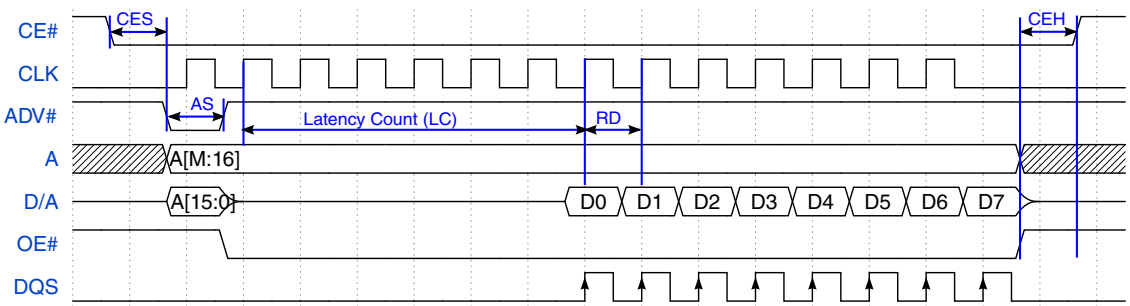


Figure 25-64. SYNC SRAM Read in ADMUX Address Mode

NOTE

- DQS is the clock to sample data from SRAM. It generally loops back from DQS pad.
- For this example:
  - CE# setup time is 1 clock cycle (SRAMCR1[CES]=0)
  - Address setup time is 1 clock cycle (SRAMCR1[AS]=0)
  - Latency count is 6 clock cycles (SRAMCR2[LC]=6)
  - Read time is 1 clock cycle (SRAMCR2[RD]=0)
  - CE# hold time is 1 clock cycle (SRAMCR1[CEH]=0)

25.3.1.7.6 SRAM Write Operation in SYNC Mode

Figure 25-65 shows the SYNC SRAM write in ADMUX address mode.

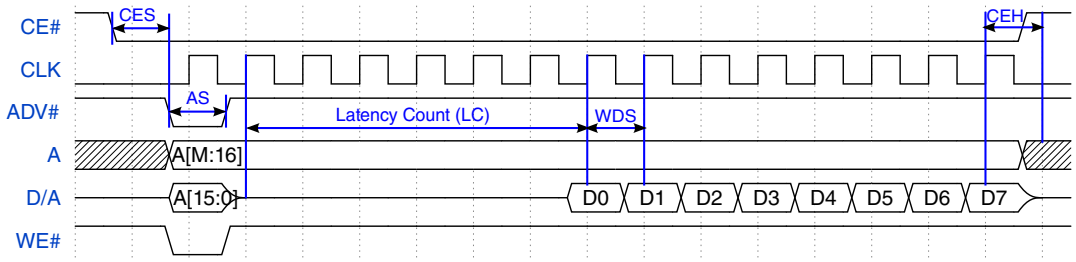


Figure 25-65. SRAM Write in SYNC Mode (ADMUX)

NOTE

- For this example:
  - CE# setup time is 1 clock cycle (SRAMCR1[CES]=0)
  - Address setup time is 1 clock cycle (SRAMCR1[AS]=0)
  - Latency count is 6 clock cylces (SRAMCR2[LC]=6)
  - Write data setup time is 1 clock cycle (SRAMCR2[WDS]=0)
  - CE# hold time is 1 clock cycle (SRAMCR1[CEH]=0)

25.3.1.8 Display Bus Interface Controller Operations

This section describes the operations of DBI controller.

DCX signal is controlled by write address. Write access to DBI is handled as DBI COMMAND WRITE transaction if address is higher than 0x10000 (DCX low). Otherwise, it is handled as DBI DATA WRITE transaction (DCX high).

For example, if BR7[BA]=0x9C000:

- An AXI write to address 0x9C010000 is treated as DBI COMMAND WRITE transaction.
- An AXI write to address 0x9C000000 is treated as DBI DATA WRITE transaction.

Figure 25-66 shows DBI write operation.

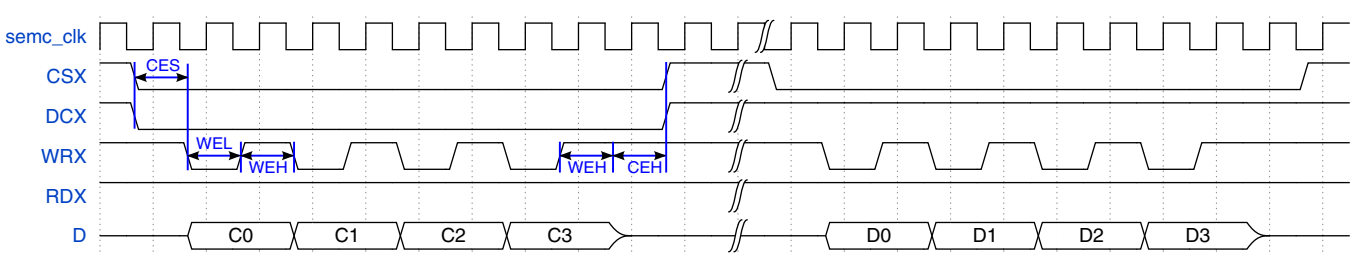


Figure 25-66. DBI Write Operation

NOTE

- DBI address is used to determine DBI COMMAND or DATA WRITE.
- DBI bus never send out address information for COMMAND and DATA WRITE.
- AXI WRAP write access to DBI is supported.
- DBI bus interface does not support WRITE mask feature.
- All write strobe in AXI write access to DBI must be valid. If AXI write strobe is invalid, the SEMC sends write data 0x00 instead of original write data.
- When DBI bus is 16 bit (DBICR0[PS]=1), AXI or IP command access start address to DBI bus must be 16 bit aligned. Otherwise the write data to DBI may be wrong. There is no bus error response or internal error status generated for this case.
- For this example:
  - CSX setup time is 1 clock cycle (DBICR1[CES]=0)
  - WRX low time is 1 clock cycle (DBICR1[WEL]=0)
  - WRX high time is 1 clock cycle (DBICR1[WEH]=0)
  - CSX hold time is 1 clock cycle (DBICR1[CEH]=0)

Figure 25-67 shows DBI read operation.

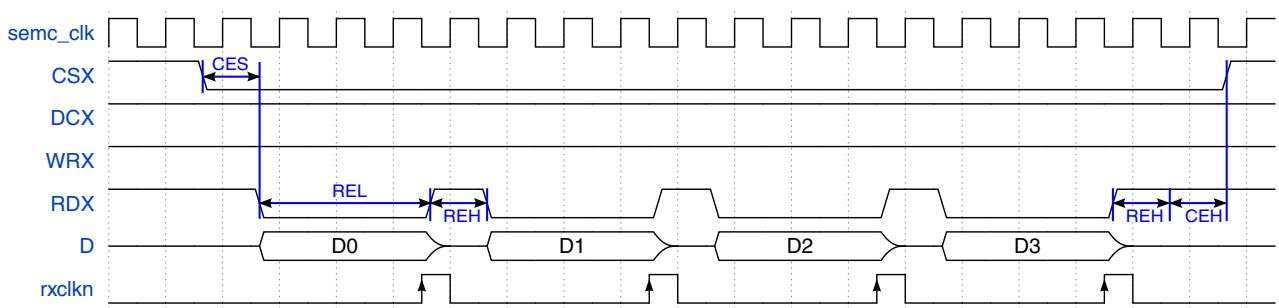


Figure 25-67. DBI Read Operation

NOTE

- Read access to DBI is always handled as DBI DATA READ transaction.
- DBI bus never sent out address information for DATA READ command.
- AXI WRAP read access to DBI is not supported except when the first beat address offset is zero in the burst.
- When DBI bus is 16 bit (DBICR0[PS]=1), AXI command or IP command access start address to DBI bus must be 16 bit aligned. Otherwise the read data is unkonwn. There is

no bus error response or internal error status generated for this case.

- For this example:
  - CSX setup time is 1 clock cycle (DBICR1[CES]=0)
  - RDX low time is 3 clock cycles (DBICR1[REL]=2)
  - RDX high time is 1 clock cycle (DBICR1[REH]=0)
  - CSX hold time is 1 clock cycle (DBICR1[CEH]=0)

25.3.2 Clocks

There is one clock used for the SEMC AXI bus, IP bus, internal logic and external interface. It is shown as semc_clk in SEMC timing figures.

25.3.3 Reset

There is a hardware reset for the SEMC, which reset all the logic inside it. When set MCR[SWRST], there will be a software reset to reset all logic in the SEMC except configuration registers.

25.3.4 Interrupts

There is an interrupt output from the SEMC. It indicates the interrupt source that defined in INTR register. The interrupt source is enabled by INTEN register.

25.4 External Signals

Table 25-5 lists the SEMC module signals.

Table 25-5. SEMC Signals

Signal	Direction	Description
SEMC_DATA[15-00]	I/O	Data/Address Bits The SEMC have several controllers, and they share pins. SEMC_DATA works as data signal in SDRAM/NAND/8080 applications and works as data or address signal in NOR/SRAM applications. For detail pin mux information, refer to Pin Mux.
SEMC_DM[1-0]	O	Write Data Mask Bits

Table continues on the next page...

Table 25-5. SEMC Signals (continued)

Signal	Direction	Description
SEMC_ADDR[12-00]	O	Address Bits SEMC_ADDR08 can work as chip select pin in NAND/NOR/SRAM/ 8080 applications.
SEMC_BA[1-0]	O	Bank Address Bits SEMC_BA0 can work as WAIT in SRAM application. SEMC_BA1 can work as ALE/ADV#/DCX in NAND/NOR/SRAM/8080 applications.
SEMC_CAS	O	SDRAM Column Address Strobe
SEMC_RAS	O	SDRAM Row Address Strobe
SEMC_WE	O	SDRAM Write Enable
SEMC_CKE	O	SDRAM Clock enable
SEMC_CLK	O	SDRAM Clock
SEMC_CS0	O	SDRAM Chip Select 0
SEMC_CSX[3-0]	O	Configurable Chip Select bits
SEMC_RDY	I/O	Ready input pin for NAND It can also work as chip select pin in SDRAM/NOR/SRAM/8080 applications.
SEMC_DQS	I/O	DQS SEMC controller provides internal dummy read strobe for read data from SDRAM/NOR/SRAM. Higher read frequency can be achieved by looping back this dummy read strobe from pad. This pin can be floated or put some cap loads on board level to compensate DATA/CLK pins load.
SEMC_DQS4	I/O	DQS Data read strobe for NAND Flash.
SEMC_CLKX0	O	Configurable clock 0 for NOR/SRAM
SEMC_CLKX1	O	Configurable clock 1 for NOR/SRAM

25.4.1 Pin Mux

Table 25-6 lists the pin mux of the SEMC module.

Table 25-6. SEMC Pin Mux Overview

Signal	SDRAM	NAND	NOR (ADMUX 16bit)	SRAM (ADMUX 16bit)	DBI	Comment
SEMC_DATA00	D0	D0	D0/A0	D0/A0	D0	Hardware MUX The SEMC determines the pin function according to its internal state
SEMC_DATA01	D1	D1	D1/A1	D1/A1	D1	
SEMC_DATA02	D2	D2	D2/A2	D2/A2	D2	
SEMC_DATA03	D3	D3	D3/A3	D3/A3	D3	

Table continues on the next page...

Table 25-6. SEMC Pin Mux Overview (continued)

Signal	SDRAM	NAND	NOR (ADMUX 16bit)	SRAM (ADMUX 16bit)	DBI	Comment
SEMC_DATA04	D4	D4	D4/A4	D4/A4	D4	
SEMC_DATA05	D5	D5	D5/A5	D5/A5	D5	
SEMC_DATA06	D6	D6	D6/A6	D6/A6	D6	
SEMC_DATA07	D7	D7	D7/A7	D7/A7	D7	
SEMC_DATA08	D8	D8	D8/A8	D8/A8	D8	
SEMC_DATA09	D9	D9	D9/A9	D9/A9	D9	
SEMC_DATA10	D10	D10	D10/A10	D10/A10	D10	
SEMC_DATA11	D11	D11	D11/A11	D11/A11	D11	
SEMC_DATA12	D12	D12	D12/A12	D12/A12	D12	
SEMC_DATA13	D13	D13	D13/A13	D13/A13	D13	
SEMC_DATA14	D14	D14	D14/A14	D14/A14	D14	
SEMC_DATA15	D15	D15	D15/A15	D15/A15	D15	
SEMC_DM1	DQM1	-	-	UB#	-	
SEMC_DM0	DQM0	-	-	LB#	-	
SEMC_ADDR00	A0	-	A16	A16	-	
SEMC_ADDR01	A1	-	A17	A17	-	
SEMC_ADDR02	A2	-	A18	A18	-	
SEMC_ADDR03	A3	-	A19	A19	-	
SEMC_ADDR04	A4	-	A20	A20	-	
SEMC_ADDR05	A5	-	A21	A21	-	
SEMC_ADDR06	A6	-	A22	A22	-	
SEMC_ADDR07	A7	-	A23	A23	-	
SEMC_ADDR08	A8	CE#	CE#/A24	CE#/A24	CSX	Configured by IOCR[MUX_A8]
SEMC_ADDR09	A9	CLE	-	CRE	-	Hardware MUX
SEMC_ADDR10	A10	-	-	-	-	The SEMC determines the pin function according to its internal state
SEMC_ADDR11	A11	WE#	WE#	WE#	WRX	SEMC_ADDR11 works as CLK signal in NAND synchronous mode
SEMC_ADDR12	A12	RE#	OE#	OE#	RDX	SEMC_ADDR12 works as W/R# signal in NAND synchronous mode
SEMC_BA0	BA0	-	-	-	-	Hardware MUX
SEMC_BA1	BA1	ALE	ADV#	ADV#	DCX	The SEMC determines the pin function according to its internal state
SEMC_CAS	CAS#	-	-	-	-	
SEMC_RAS	RAS#	-	-	-	-	
SEMC_WE	WE#	-	-	-	-	
SEMC_CKE	CKE	-	-	-	-	
SEMC_CLK	CLK	-	-	-	-	
SEMC_CS0	CS0	-	-	-	-	

Table continues on the next page...

Table 25-6. SEMC Pin Mux Overview (continued)

Signal	SDRAM	NAND	NOR (ADMUX 16bit)	SRAM (ADMUX 16bit)	DBI	Comment
SEMC_CSX0	CS	CE#	CE#	CE#	CSX	Configured by IOCR[MUX_CSX0]
SEMC_CSX1	CS	CE#	CE#	CE#	CSX	Configured by IOCR[MUX_CSX1]
SEMC_CSX2	CS	CE#	CE#	CE#	CSX	Configured by IOCR[MUX_CSX2]
SEMC_CSX3	CS	CE#	CE#	CE#	CSX	Configured by IOCR[MUX_CSX3]
SEMC_RDY	CS	R/B#	CE#	CE#	CSX	Configured by IOCR[MUX_RDY]
SEMC_DQS	DQS	-	DQS	DQS	-	Hardware MUX
SEMC_DQS4	-	DQS	-	-	-	Hardware MUX
SEMC_CLKX0	-	-	CLK	CLK	-	Configured by IOCR[MUX_CLKX0]
SEMC_CLKX1	-	-	CLK	CLK	-	Configured by IOCR[MUX_CLKX1]

NOTE

- IOCR configuration should be static. Dynamic remapping of pin muxing controlled by IOCR is not supported.
- NOR and SRAM pins are configured as 16 bit ADMUX mode in above table. It is the recommended configuration.

25.5 Initialization

To initialize the block:

- Ensure the clock for the SEMC is enabled.
- Configure corresponding pin MUX for the SEMC external signals.
- Configure registers of the SEMC properly, according to the applications.
- Configure the Module Control Register and clear MCR[MDIS] bit to enable the SEMC module.
- Reset SEMC optionally by setting the MCR[SWRST] bit.

25.6 Application Information

This section describes applications supported by the SEMC module.

25.6.1 SDRAM Application

This section provides the example sequences for SDRAM device.

- Configure MCR[DQSMD] bit to select the read clock source. Suggest to set it with 0x1 to reach high clock frequency.
- Configure IOCR register to choose chip select pins.
- Configure BMCR0/1 registers to adjust AXI bus access efficiency scheme.
- Configure Base Register 0/1/2/3 with base address, memory size and valid information.
- Configure INTEN and INTR registers if need to generate interrupt.
- Configure SDRAM Control Registers with valid settings that matchs SDRAM device.
- Initialize SDRAM device by IP command registers (IPCR0/1/2, IPCMD and IPTXDAT).
- Write and read operations can be triggered by AXI or IP command. However, AXI command is suggested.

25.6.2 NAND Application

This section provides the example sequences for NAND device.

- Configure IOCR register to choose chip select pins.
- Configure BMCR0/1 registers to adjust AXI bus access efficiency scheme.
- Configure Base Register 4/8 with base address, memory size and valid information.
- Configure INTEN and INTR registers if need to generate interrupt.
- Configure NAND Control Registers with valid settings that matchs NAND device.
- Configure DLL Control Register for synchronous mode.
- Initialize NAND device by IP command registers (IPCR0/1/2, IPCMD and IPTXDAT).
- For programming operation, need to open page by IP command. Then you can write data to the NAND buffer by IP or AXI bus.
- For read operation, need to open page by IP command. Then you can read data from the NAND buffer by IP or AXI bus.
- For erase operation, you should do it by IP command.

25.6.3 NOR Application

This section provides the example sequences for NOR device.

- Configure MCR[DQSMD] bit to select the read clock source for synchronous mode. Suggest to set it with 0x1 to reach high clock frequency.

- Configure IOCR register to choose chip select pins.
- Configure BMCR0/1 registers to adjust AXI bus access efficiency scheme.
- Configure Base Register 5 with base address, memory size and valid information.
- Configure INTEN and INTR registers if need to generate interrupt.
- Configure NOR Control Registers with valid settings that matchs NOR device.
- Initialize NOR device by IP command registers (IPCR0/1/2, IPCMD and IPTXDAT) if needed.
- Programming operation can be triggered by IP command only.
- Read operation can be triggered by AXI or IP command.

25.6.4 SRAM Application

This section provides the example sequences for SRAM device.

- Configure MCR[DQSMD] bit to select the read clock source for synchronous mode. Suggest to set it with 0x1 to reach high clock frequency.
- Configure IOCR register to choose chip select pins.
- Configure BMCR0/1 registers to adjust AXI bus access efficiency scheme.
- Configure Base Register 6 with base address, memory size and valid information.
- Configure INTEN and INTR registers if need to generate interrupt.
- Configure SRAM Control Registers with valid settings that matchs SRAM device.
- Initialize SRAM device by IP command registers (IPCR0/1/2, IPCMD and IPTXDAT) if needed.
- Write and read operations can can be triggered by AXI or IP command.

25.6.5 DBI Application

This section provides the example sequences for Display Bus Interface.

- Configure IOCR register to choose chip select pins.
- Configure BMCR0/1 registers to adjust AXI bus access efficiency scheme.
- Configure Base Register 7 with base address, memory size and valid information.
- Configure INTEN and INTR registers if need to generate interrupt.
- Configure DBI Control Registers with valid settings that matchs DBI device.
- Write and read operations can be triggered by AXI or IP command.

25.7 Memory Map and Register Definition

This section includes the SEMC module memory map and detailed descriptions of all registers.

25.7.1 SEMC register descriptions

25.7.1.1 SEMC memory map

SEMC base address: 402F_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Module Control Register (MCR)	32	RW	1000_0002h
4h	IO MUX Control Register (IOCR)	32	RW	0000_0000h
8h	Bus (AXI) Master Control Register 0 (BMCR0)	32	RW	0000_0000h
Ch	Bus (AXI) Master Control Register 1 (BMCR1)	32	RW	0000_0000h
10h - 30h	Base Register n (BR0 - BR8)	32	RW	0000_0000h
34h	DLL Control Register (DLLCR)	32	RW	0000_0100h
38h	Interrupt Enable Register (INTEN)	32	RW	0000_0000h
3Ch	Interrupt Register (INTR)	32	RW	0000_0000h
40h	SDRAM Control Register 0 (SDRAMCR0)	32	RW	0000_0C26h
44h	SDRAM Control Register 1 (SDRAMCR1)	32	RW	0099_4934h
48h	SDRAM Control Register 2 (SDRAMCR2)	32	RW	8000_0EEEh
4Ch	SDRAM Control Register 3 (SDRAMCR3)	32	RW	4080_8000h
50h	NAND Control Register 0 (NANDCR0)	32	RW	0000_0000h
54h	NAND Control Register 1 (NANDCR1)	32	RW	0000_0000h
58h	NAND Control Register 2 (NANDCR2)	32	RW	0001_0410h
5Ch	NAND Control Register 3 (NANDCR3)	32	RW	0000_0000h
60h	NOR Control Register 0 (NORCR0)	32	RW	0000_0000h
64h	NOR Control Register 1 (NORCR1)	32	RW	0000_0000h
68h	NOR Control Register 2 (NORCR2)	32	RW	0000_0000h
6Ch	NOR Control Register 3 (NORCR3)	32	RW	0000_0000h
70h	SRAM Control Register 0 (SRAMCR0)	32	RW	0000_0000h
74h	SRAM Control Register 1 (SRAMCR1)	32	RW	0000_0000h
78h	SRAM Control Register 2 (SRAMCR2)	32	RW	0000_0000h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
7Ch	SRAM Control Register 3 (SRAMCR3)	32	RW	0000_0000h
80h	DBI-B Control Register 0 (DBICR0)	32	RW	0000_0000h
84h	DBI-B Control Register 1 (DBICR1)	32	RW	0000_0000h
90h	IP Command Control Register 0 (IPCR0)	32	RW	0000_0000h
94h	IP Command Control Register 1 (IPCR1)	32	RW	0000_0000h
98h	IP Command Control Register 2 (IPCR2)	32	RW	0000_0000h
9Ch	IP Command Register (IPCMD)	32	RW	0000_0000h
A0h	TX DATA Register (IPTXDAT)	32	RW	0000_0000h
B0h	RX DATA Register (IPRXDAT)	32	RO	0000_0000h
C0h	Status Register 0 (STS0)	32	RO	0000_0001h
C4h	Status Register 1 (STS1)	32	RO	0000_0000h
C8h	Status Register 2 (STS2)	32	RO	0000_0000h
CCh	Status Register 3 (STS3)	32	RO	0000_0000h
D0h	Status Register 4 (STS4)	32	RO	0000_0000h
D4h	Status Register 5 (STS5)	32	RO	0000_0000h
D8h	Status Register 6 (STS6)	32	RO	0000_0000h
DCh	Status Register 7 (STS7)	32	RO	0000_0000h
E0h	Status Register 8 (STS8)	32	RO	0000_0000h
E4h	Status Register 9 (STS9)	32	RO	0000_0000h
E8h	Status Register 10 (STS10)	32	RO	0000_0000h
ECh	Status Register 11 (STS11)	32	RO	0000_0000h
F0h	Status Register 12 (STS12)	32	RO	0000_0000h
F4h	Status Register 13 (STS13)	32	RO	0000_0100h
F8h	Status Register 14 (STS14)	32	RO	0000_0000h
FCh	Status Register 15 (STS15)	32	RO	0000_0000h

25.7.1.2 Module Control Register (MCR)

25.7.1.2.1 Offset

Register	Offset
MCR	0h

25.7.1.2.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved			BTO					CTO							
Reset	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								WPOL1	WPOL0	Reserved			DQSMD	MDIS	SWRST
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

25.7.1.2.3 Fields

Field	Function
31-29 —	Reserved
28-24 BTO	Bus timeout cycles The SEMC monitors AXI transactions in the queue. When it exceeds the timeout cycles, an AXIBUSERR interrupt is generated. The AXI Bus timeout is $255*2^{BTO}$ clock cycles. 00000b - 255*1 00001b - 255*2 11111b - $255*2^{31}$
23-16 CTO	Command Execution timeout cycles When Command Execution time exceeds the timeout cycles, an IPCMDERR or AXICMDERR interrupt is generated. When CTO is set to zero, timeout cycles is 256*1024 cycle. Otherwise timeout cycles is CTO*1024 cycle.
15-8 —	Reserved
7 WPOL1	R/B# polarity for NAND device This bit configures the expected polarity of R/B# signal from NAND device. 0b - R/B# polarity is not changed. 1b - R/B# polarity is inverted.
6 WPOL0	WAIT/RDY polarity for SRAM/NOR This bit configures the expected polarity of WAIT/RDY signal from SRAM and NOR devices. 0b - WAIT/RDY polarity is not changed. 1b - WAIT/RDY polarity is inverted.
5-3 —	Reserved
2 DQSMD	DQS (read strobe) mode DQSMD controls the read clock source of SDRAM, NOR and SRAM in synchronous mode. When it set, the read clock is loopbacked from DQS pad, it benefits the read path timing, and can get high read clock frequency.

Table continues on the next page...

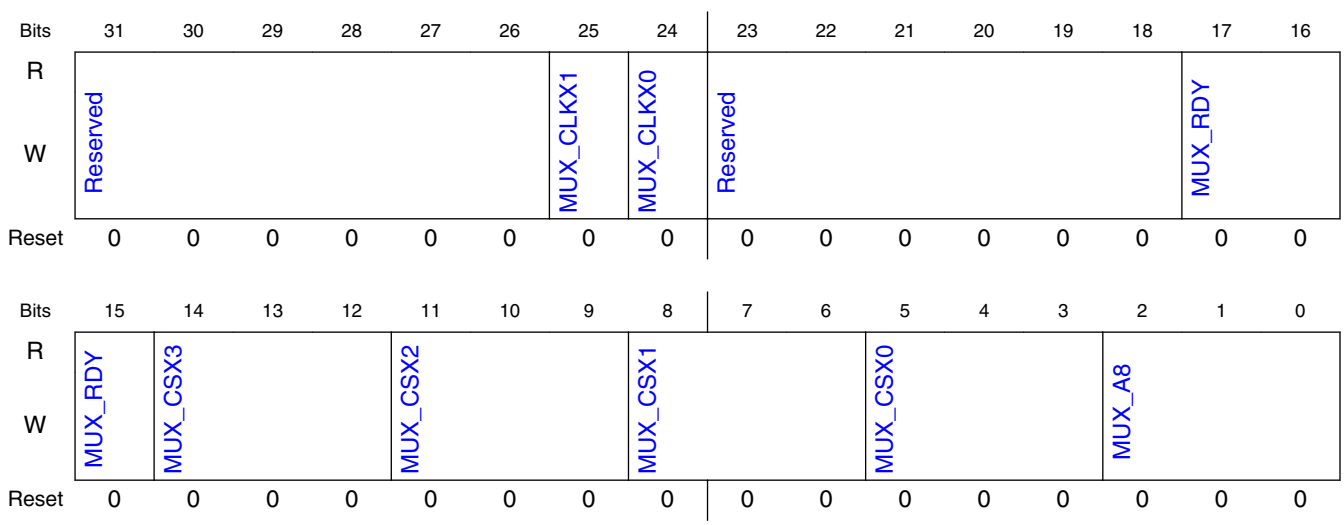
Field	Function
	0b - Dummy read strobe loopbacked internally 1b - Dummy read strobe loopbacked from DQS pad
1 MDIS	Module Disable When this bit set, the SEMC internal clock is stopped, and no function is available except register access. 0b - Module enabled 1b - Module disabled
0 SWRST	Software Reset Reset all internal logic in the SEMC except configuration registers. It is an auto clear bit. 0b - No reset 1b - Reset

25.7.1.3 IO MUX Control Register (IOCR)

25.7.1.3.1 Offset

Register	Offset
IOCR	4h

25.7.1.3.2 Diagram



25.7.1.3.3 Fields

Field	Function
31-26	Reserved

Table continues on the next page...

Field	Function
—	
25 MUX_CLKX1	SEMC_CLKX1 function selection This field controls the clock source of SEMC_CLKX1 signal. 0b - NOR clock 1b - SRAM clock
24 MUX_CLKX0	SEMC_CLKX0 function selection This field controls the clock source of SEMC_CLKX0 signal. 0b - NOR clock 1b - SRAM clock
23-18 —	Reserved
17-15 MUX_RDY	SEMC_RDY function selection This field controls the function of SEMC_RDY signal. 000b - NAND R/B# input 001b - SDRAM CS1 010b - SDRAM CS2 011b - SDRAM CS3 100b - NOR CE# 101b - SRAM CE# 110b - DBI CSX 111b - Reserved
14-12 MUX_CSX3	SEMC_CSX3 output selection This field controls the function of SEMC_CSX3 signal. 000b - Reserved 001b - SDRAM CS1 010b - SDRAM CS2 011b - SDRAM CS3 100b - NAND CE# 101b - NOR CE# 110b - SRAM CE# 111b - DBI CSX
11-9 MUX_CSX2	SEMC_CSX2 output selection This field controls the function of SEMC_CSX2 signal. 000b - Reserved 001b - SDRAM CS1 010b - SDRAM CS2 011b - SDRAM CS3 100b - NAND CE# 101b - NOR CE# 110b - SRAM CE# 111b - DBI CSX
8-6 MUX_CSX1	SEMC_CSX1 output selection This field controls the function of SEMC_CSX1 signal. 000b - Reserved 001b - SDRAM CS1 010b - SDRAM CS2 011b - SDRAM CS3 100b - NAND CE# 101b - NOR CE# 110b - SRAM CE# 111b - DBI CSX

Table continues on the next page...

Field	Function
5-3 MUX_CSX0	SEMC_CSX0 output selection This field controls the function of SEMC_CSX0 signal. 000b - Reserved 001b - SDRAM CS1 010b - SDRAM CS2 011b - SDRAM CS3 100b - NAND CE# 101b - NOR CE# 110b - SRAM CE# 111b - DBI CSX
2-0 MUX_A8	SEMC_ADDR08 output selection This field controls the function of SEMC_ADDR08 signal. 000b - SDRAM Address bit 8 (A8) or NOR/SRAM Address bit 24 (A24) in ADMUX 16bit mode 001b - NAND CE# 010b - NOR CE# 011b - SRAM CE# 100b - DBI CSX 101-111b - SDRAM Address bit 8 (A8) or NOR/SRAM Address bit 24 (A24) in ADMUX 16bit mode

25.7.1.4 Bus (AXI) Master Control Register 0 (BMCR0)

25.7.1.4.1 Offset

Register	Offset
BMCR0	8h

25.7.1.4.2 Function

Queue A weight settings for AXI bus access to SDRAM, NAND, NOR, SRAM and DBI slaves.

NOTE

Please refer to BMCR0 Registers Configuration

25.7.1.4.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved								WRWS							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	WSH								WAGE				WQOS			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

25.7.1.4.4 Fields

Field	Function
31-24 —	Reserved
23-16 WRWS	Weight of slave hit with Read/Write Switch This weight score is valid when the queue command's slave is same as current executing command with read/write operation switch.
15-8 WSH	Weight of Slave Hit without read/write switch This weight score is valid when queue command's slave is the same as current executing command without read/write operation switch.
7-4 WAGE	Weight of AGE Each command in queue has an age score to indicate its wait period. The age score is multiplied by WAGE to get the weight score. Recommend to use non-zero value.
3-0 WQOS	Weight of QOS AXI bus access has AxQOS signal set, which is used as a priority indicator for the associated write or read transaction. A higher value indicates a higher priority transaction. AxQOS is multiplied by WQOS to get the weight score.

25.7.1.5 Bus (AXI) Master Control Register 1 (BMCR1)

25.7.1.5.1 Offset

Register	Offset
BMCR1	Ch

25.7.1.5.2 Function

Queue B weight settings for AXI bus access to SDRAM slave.

NOTE

Please refer to BMCR1 Registers Configuration

25.7.1.5.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	WBR								WRWS							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	WPH								WAGE				WQOS			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

25.7.1.5.4 Fields

Field	Function
31-24 WBR	Weight of Bank Rotation This weight score is valid when queue command's bank is not same as current executing command.
23-16 WRWS	Weight of slave hit without Read/Write Switch This weight score is valid when queue command's read/write operation is same as current executing command.
15-8 WPH	Weight of Page Hit This weight score is valid when queue command's page hits current executing command.
7-4 WAGE	Weight of AGE Each command in queue has an age signal to indicate its wait period. It is multiplied by WAGE to get the weight score. Recommend to use non-zero value.
3-0 WQOS	Weight of QOS AXI bus access has AxQOS signal set, which is used as a priority indicator for the associated write or read transaction. A higher value indicates a higher priority transaction. AxQOS is multiplied by WQOS to get the weight score.

25.7.1.6 Base Register n (BR0 - BR8)

25.7.1.6.1 Offset

For n = 0 to 8:

Register	Offset
BRn	10h + (n × 4h)

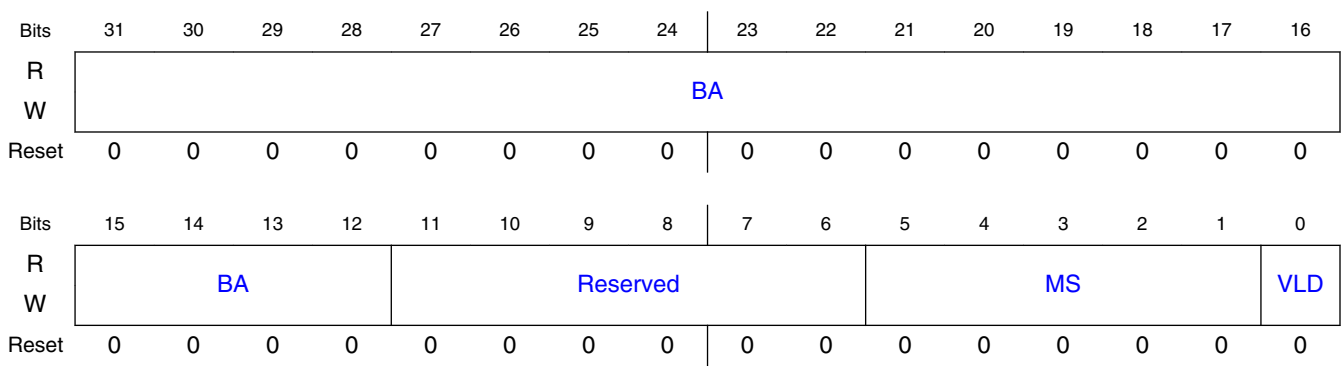
25.7.1.6.2 Function

Table 25-7 shows the detail information about the Base Registers.

Table 25-7. Base Register Information

Register	Reset Value	Purpose	Comments
BR0	0x8000001D	For SDRAM CS0	For AXI and IP commands
BR1	0x8400001C	For SDRAM CS1	For AXI and IP commands
BR2	0x8800001C	For SDRAM CS2	For AXI and IP commands
BR3	0x8C00001C	For SDRAM CS3	For AXI and IP commands
BR4	0x9E00001A	For NAND	For AXI command only
BR5	0x90000018	For NOR	For AXI and IP commands
BR6	0x98000018	For SRAM	For AXI and IP commands
BR7	0x9C00001A	For DBI-B	For AXI and IP commands
BR8	0x00000026	For NAND	For IP command only

25.7.1.6.3 Diagram



25.7.1.6.4 Fields

Field	Function
31-12 BA	Base Address

Table continues on the next page...

Field	Function
	This field determines high position 20 bits of SoC level base address. The low position 12 bits of the address are all zero. The base address must be within the SEMC range defined for the device memory map.
11-6 —	Reserved
5-1 MS	Memory size It defines the size of the memory. Memory size must be aligned with base address. Memory regions should be configured to avoid overlapping areas. Overlapping regions will cause bus errors. 00000b - 4KB 00001b - 8KB 00010b - 16KB 00011b - 32KB 00100b - 64KB 00101b - 128KB 00110b - 256KB 00111b - 512KB 01000b - 1MB 01001b - 2MB 01010b - 4MB 01011b - 8MB 01100b - 16MB 01101b - 32MB 01110b - 64MB 01111b - 128MB 10000b - 256MB 10001b - 512MB 10010b - 1GB 10011b - 2GB 10100-11111b - 4GB
0 VLD	Valid This bit controls whether the memory is valid or not. 0b - The memory is invalid, can not be accessed. 1b - The memory is valid, can be accessed.

25.7.1.7 DLL Control Register (DLLCR)

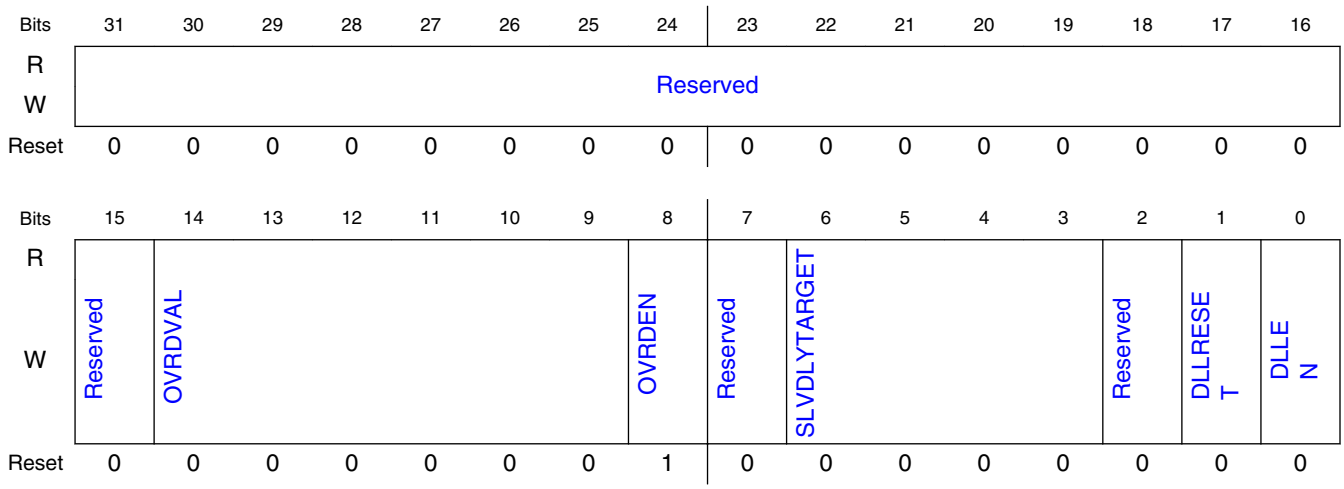
25.7.1.7.1 Offset

Register	Offset
DLLCR	34h

25.7.1.7.2 Function

This register provides the configuration fields for sample clock from DLL. It is for NAND only.

25.7.1.7.3 Diagram



25.7.1.7.4 Fields

Field	Function
31-15 —	Reserved
14-9 OVRDVAL	Override Value When OVRDEN is set, the delay cell number is OVRDVAL+1 for slave clock delay line.
8 OVRDEN	Override Enable Slave clock delay line delay cell number selection override enable. 0b - The delay cell number is not overridden. 1b - The delay cell number is overridden.
7 —	Reserved
6-3 SLVDLYTARGET	Delay Target for Slave The delay target for slave delay line is: ((SLVDLYTARGET+1) * 1/32 * clock cycle of reference clock (ipgclock).
2 —	Reserved
1 DLLRESET	DLL Reset Software can force a reset on DLL by setting this field to 0x1. This causes the DLL to lose lock and re-calibrate to detect an ref_clock half period phase shift. The reset action is edge triggered, so software need to clear this bit after setting this bit (no delay limitation). 0b - DLL is not reset. 1b - DLL is reset.
0 DLLEN	DLL calibration enable

Field	Function
	When this bit is cleared, DLL calibration is disabled and the delay cell number in slave delay line is always 1. Please note that SLV delay line is overridden when OVRDEN bit is set and this bit field setting is ignored. 0b - DLL calibration is disabled. 1b - DLL calibration is enabled.

25.7.1.8 Interrupt Enable Register (INTEN)

25.7.1.8.1 Offset

Register	Offset
INTEN	38h

25.7.1.8.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved										NDNOPENDEN	NDPAGEENDEN	AXIBUSERRE N	AXICMDERREN	IPCMDERREN	IPCMDDONEEN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

25.7.1.8.3 Fields

Field	Function
31-6 —	Reserved
5 NDNOPENDEN	NAND no pending AXI access interrupt enable This bit enable/disable the NDNOPEND interrupt generation. 0b - Interrupt is disabled 1b - Interrupt is enabled

Table continues on the next page...

Field	Function
4 NDPAGEENDEN	NAND page end interrupt enable This bit enable/disable the NDPAGEEND interrupt generation. 0b - Interrupt is disabled 1b - Interrupt is enabled
3 AXIBUSERREN	AXI bus error interrupt enable 0b - Interrupt is disabled 1b - Interrupt is enabled
2 AXICMDERREN	AXI command error interrupt enable 0b - Interrupt is disabled 1b - Interrupt is enabled
1 IPCMDERREN	IP command error interrupt enable 0b - Interrupt is disabled 1b - Interrupt is enabled
0 IPCMDDONEEN	IP command done interrupt enable 0b - Interrupt is disabled 1b - Interrupt is enabled

25.7.1.9 Interrupt Register (INTR)

25.7.1.9.1 Offset

Register	Offset
INTR	3Ch

25.7.1.9.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved										NDNOPEND	NDPAGEEND	AXIBUSERR	AXICMDERR	IPCMDERR	IPCMDDONE
W											W1C	W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

25.7.1.9.3 Fields

Field	Function
31-6 —	Reserved
5 NDNOPEND	NAND no pending AXI write transaction interrupt This interrupt is generated when all pending AXI write transactions to NAND are finished. 0b - At least one NAND AXI write transaction is pending or no NAND write transaction is sent to the queue. 1b - All NAND AXI write pending transactions are finished.
4 NDPAGEEND	NAND page end interrupt This interrupt is generated when the last address of one page in NAND device is written by AXI command. 0b - The last address of main space in the NAND is not written by AXI command. 1b - The last address of main space in the NAND is written by AXI command.
3 AXIBUSERR	AXI bus error interrupt AXI Bus error interrupt is generated in following cases: • AXI address is invalid • AXI write to NOR Flash • AXI 8-bit write to 16-bit NAND Flash • AXI 8-bit or 16-bit WRAP write/read 0b - No AXI bus error. 1b - AXI bus error occurs.
2 AXICMDERR	AXI command error interrupt AXI command error interrupt is generated when AXI command execution times out. 0b - No AXI command error. 1b - AXI command error occurs.
1 IPCMDERR	IP command error done interrupt IP command error interrupt is generated in following cases: • IP Command Address target invalid device space • IP Command Code unsupported • IP Command triggered when previous command not finished yet • IP Command Execution timeout 0b - No IP command error. 1b - IP command error occurs.
0 IPCMDDONE	IP command normal done interrupt It monitors whether IP command is done or not. 0b - IP command is not done. 1b - IP command is done.

25.7.1.10 SDRAM Control Register 0 (SDRAMCR0)

25.7.1.10.1 Offset

Register	Offset
SDRAMCR0	40h

25.7.1.10.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved	BANK2	Reserved		CL		COL		COL8	BL			Reserved			PS
Reset	0	0	0	0	1	1	0	0	0	0	1	0	0	1	1	0

25.7.1.10.3 Fields

Field	Function
31-15 —	Reserved
14 BANK2	2 Bank selection bit This bit configures the bank numbers. 0b - SDRAM device has 4 banks. 1b - SDRAM device has 2 banks.
13-12 —	Reserved
11-10 CL	CAS Latency The CAS latency (CL) is the delay, in clock cycles, between the registration of a READ command and the availability of the output data. 00b - 1 01b - 1 10b - 2 11b - 3
9-8 COL	Column address bit number 00b - 12 01b - 11 10b - 10 11b - 9
7	Column 8 selection

Table continues on the next page...

Field	Function
COL8	This bit works with COL field to determine the column address bit number. 0b - Column address bit number is decided by COL field. 1b - Column address bit number is 8. COL field is ignored.
6-4 BL	Burst Length The burst length determines the number of column locations that can be accessed for a given READ or WRITE command. 000b - 1 001b - 2 010b - 4 011b - 8 100b - 8 101b - 8 110b - 8 111b - 8
3-1 —	Reserved
0 PS	Port Size The port size must be aligned with SDRAM data width. 0b - 8bit 1b - 16bit

25.7.1.11 SDRAM Control Register 1 (SDRAMCR1)

25.7.1.11.1 Offset

Register	Offset
SDRAMCR1	44h

25.7.1.11.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved								ACT2PRE				CKEOFF			
Reset	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0	1

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	WRC			RFRC					ACT2RW				PRE2ACT			
Reset	0	1	0	0	1	0	0	1	0	0	1	1	0	1	0	0

25.7.1.11.3 Fields

Field	Function
31-24 —	Reserved
23-20 ACT2PRE	ACTIVE to PRECHARGE minimum time The SEMC promises ACT2PRE+1 clock cycles delay between ACTIVE command to PRECHARGE/ PRECHARGE ALL command.
19-16 CKEOFF	CKE off minimum time The SEMC promises at least CKEOFF+1 clock cycles in self refresh mode. It helps to meet the minimum period request to remain in self refresh mode.
15-13 WRC	WRITE recovery time The SEMC promises WRC+1 clock cycles delay between WRITE command to PRECHARGE/ PRECHARGE ALL command. It helps to meet tWR timing requirement by SDRAM device.
12-8 RFRC	REFRESH recovery time The SEMC promises RFRC+1 clock cycles delay between AUTO REFRESH command to ACTIVE command. It helps to meet tRFC timing requirement by SDRAM device.
7-4 ACT2RW	ACTIVE to READ/WRITE delay The SEMC promises ACT2RW+1 clock cycles delay between ACTIVE command to READ/WRITE command. It helps to meet tRCD timing requirement by SDRAM device.
3-0 PRE2ACT	PRECHARGE to ACTIVE/REFRESH command wait time The SEMC promises PRE2ACT+1 clock cycles delay between PRECHARGE/PRECHARGE ALL command to ACTIVE/REFRESH command. It helps to meet tRP timing requirement by SDRAM device.

25.7.1.12 SDRAM Control Register 2 (SDRAMCR2)

25.7.1.12.1 Offset

Register	Offset
SDRAMCR2	48h

25.7.1.12.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ITO								ACT2ACT							
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	REF2REF								SRRC							
Reset	0	0	0	0	1	1	1	0	1	1	1	0	1	1	1	0

25.7.1.12.3 Fields

Field	Function
31-24 ITO	SDRAM idle timeout The SEMC closes all opened pages if the SDRAM idle time lasts more than idle timeout period. SDRAM is considered idle when there is no AXI Bus transfer and no SDRAM command pending. 00000000b - IDLE timeout period is 256*Prescale period. 00000001-11111111b - IDLE timeout period is ITO*Prescale period.
23-16 ACT2ACT	ACTIVE to ACTIVE delay The SEMC promises ACT2ACT+1 clock cycles delay between ACTIVE command to ACTIVE command. It helps to meet tRRD timing requirement by SDRAM device.
15-8 REF2REF	REFRESH to REFRESH delay The SEMC promises REF2REF+1 clock cycles delay between REFRESH command to REFRESH command. It helps to meet tRFC timing requirement by SDRAM device.
7-0 SRRC	SELF REFRESH recovery time The SEMC promises SRRC+1 clock cycles delay between SELF REFRESH command to any subsequent command. SDRAM device may have limitation on self refresh recovery time. It help to meet the wait time from exiting self refresh mode to sending new command to device.

25.7.1.13 SDRAM Control Register 3 (SDRAMCR3)

25.7.1.13.1 Offset

Register	Offset
SDRAMCR3	4Ch

25.7.1.13.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	UT								RT							
Reset	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PRESCALE								Reserved				REBL			REN
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

25.7.1.13.3 Fields

Field	Function
31-24 UT	Urgent refresh threshold Urgent refresh threshold is used to handle urgent refresh request. Urgent refresh timer starts to count when normal refresh request is asserted but not handled by the controller yet. When the urgent refresh timer count up to the threshold, urgent refresh request is sent to the controller. The priority of urgent refresh is higher than any pending AXI or IP command to SDRAM, so it can be handled by the controller at the first time. **NOTE:** Refresh request is considered as urgent refresh request when urgent threshold is no less than refresh period. 00000000b - 256*(Prescaler period) 00000001-11111111b - UT*(Prescaler period)
23-16 RT	Refresh timer period Refresh timer period is the threshold for SDRAM normal refresh request. The normal refresh request is sent to the controller when the refresh timer count up to it. The priority of normal refresh request is lower than any pending AXI or IP command to SDRAM. 00000000b - (256+1)*(Prescaler period) 00000001-11111111b - (RT+1)*(Prescaler period)
15-8 PRESCALE	Prescaler period Prescaler period is used as a clock cycle count unit for refresh and bus idle timer period. 00000000b - (256*16+1) clock cycles 00000001-11111111b - (PRESCALE*16+1) clock cycles
7-4 —	Reserved
3-1 REBL	Refresh burst length The SEMC can send multiple AUTO REFRESH commands in one burst when REBL is set to non-zero. The number of AUTO REFRESH command in one burst is listed below. 000b - 1 001b - 2 010b - 3 011b - 4 100b - 5 101b - 6 110b - 7

Table continues on the next page...

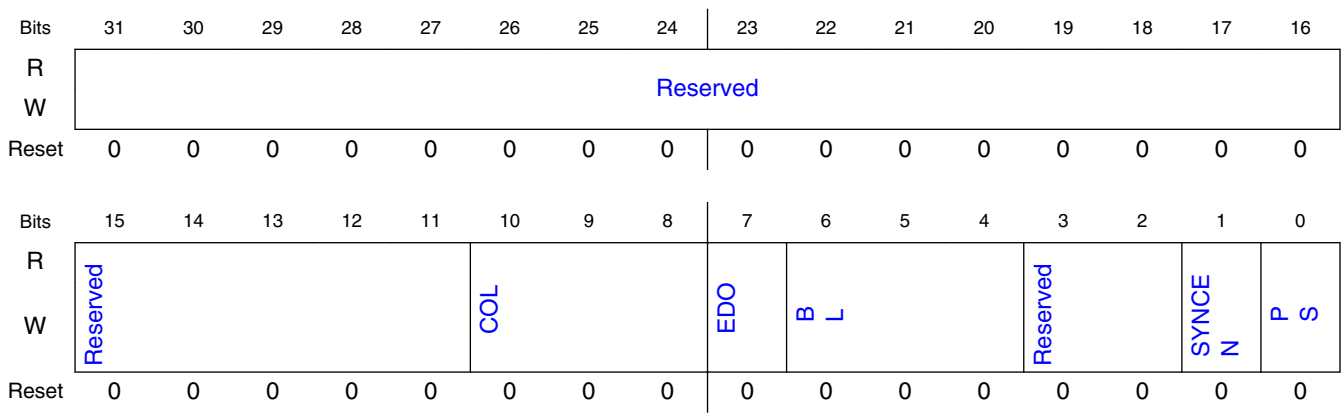
Field	Function
	111b - 8
0 REN	Refresh enable The SEMC sends AUTO REFRESH command automatically when REN bit is set. The AUTO REFRESH command interval is decided by UT, RT and PRESCALE bits. 0b - The SEMC does not send AUTO REFRESH command automatically 1b - The SEMC sends AUTO REFRESH command automatically

25.7.1.14 NAND Control Register 0 (NANDCR0)

25.7.1.14.1 Offset

Register	Offset
NANDCR0	50h

25.7.1.14.2 Diagram



25.7.1.14.3 Fields

Field	Function
31-11 —	Reserved
10-8 COL	Column address bit number 000b - 16 001b - 15 010b - 14 011b - 13

Table continues on the next page...

Field	Function
	100b - 12 101b - 11 110b - 10 111b - 9
7 EDO	EDO mode enabled It defines the read access timing in asynchronous mode. 0b - EDO mode disabled 1b - EDO mode enabled
6-4 BL	Burst Length The burst length determines the number of column locations that can be accessed for a given READ or WRITE command. 000b - 1 001b - 2 010b - 4 011b - 8 100b - 16 101b - 32 110b - 64 111b - 64
3-2 —	Reserved
1 SYNCEN	Synchronous Mode Enable It defines the access mode. 0b - Asynchronous mode is enabled. 1b - Synchronous mode is enabled.
0 PS	Port Size The port size must be aligned with NAND Flash data width. 0b - 8bit 1b - 16bit

25.7.1.15 NAND Control Register 1 (NANDCR1)

25.7.1.15.1 Offset

Register	Offset
NANDCR1	54h

25.7.1.15.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CEITV				TA				REH				REL			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	WEH				WEL				CEH				CES			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

25.7.1.15.3 Fields

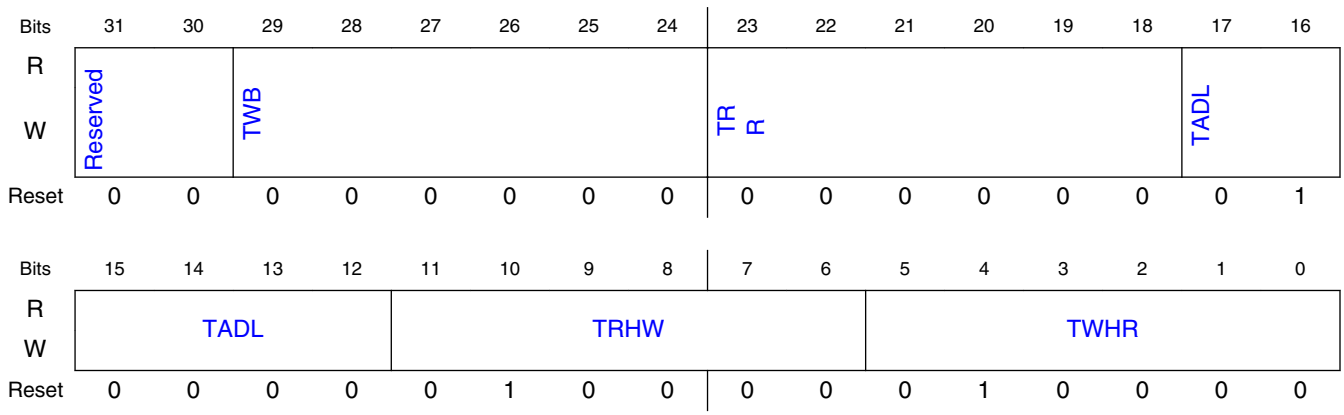
Field	Function
31-28 CEITV	CE# interval time CE# interval minimum time is CEITV+1 clock cycles.
27-24 TA	Turnaround time Turnaround time is TA+1 clock cycles. Both the SEMC and Device do not drive Data Lines to avoid bus confliction. This field applied to ASYNC mode only.
23-20 REH	RE# high time RE# high time is REH+1 clock cycles. This field applied to ASYNC mode only.
19-16 REL	RE# low time RE# low time is REL+1 clock cycles. This field applied to ASYNC mode only.
15-12 WEH	WE# high time WE# high time is WEH+1 clock cycles in ASYNC mode. It is used as command and address hold time in SYNC mode. WEH must be even number in SYNC mode, and command/address hold time is WEH clock cycles.
11-8 WEL	WE# low time WE# low time is WEL+1 clock cycles in ASYNC mode. It is used as command and address setup time in SYNC mode. WEL must be even number in SYNC mode, and command/address setup time is WEL+2 clock cycles.
7-4 CEH	CE# hold time CE# hold minimum time is CEH+1 clock cycles. CEH must be odd number in SYNC mode. CE# hold time may be larger than this time to wait all read data sampled.
3-0 CES	CE# setup time CE# setup time is CES+1 clock cycles. CES must be odd number in SYNC mode.

25.7.1.16 NAND Control Register 2 (NANDCR2)

25.7.1.16.1 Offset

Register	Offset
NANDCR2	58h

25.7.1.16.2 Diagram



25.7.1.16.3 Fields

Field	Function
31-30 —	Reserved
29-24 TWB	WE# high to busy time WE# high to busy time is TWB+1 clock cycles
23-18 TRR	Ready to RE# low time Ready to RE# low minimum time is TRR+1 clock cycles. Actual time may be 1 or 2 clock cycles more than this because of synchronizor logic. This field applied to ASYNC mode only.
17-12 TADL	Address cycle to data loading time Address to data loading time is TADL+1 clock cycles. Timing for TADL begins in the address cycle on the final rising edge of WE#, and ends with the first rising edge of WE# for data input.
11-6 TRHW	RE# high to WE# low time RE# high to WE# low time is TRHW+1 clock cycles
5-0 TWHR	WE# high to RE# low time WE# high to RE# low time is TWHR+1 clock cycles in ASYNC mode. TWHR represents command, address or data input cycle to data output clock cycles in SYNC mode. The delay is TWHR+2 clock cycles and TWHR must be even.

25.7.1.17 NAND Control Register 3 (NANDCR3)

25.7.1.17.1 Offset

Register	Offset
NANDCR3	5Ch

25.7.1.17.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	WDH				WDS				RDH				RDS			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved												Reserved	NDOPT3	NDOPT2	NDOPT1
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

25.7.1.17.3 Fields

Field	Function
31-28 WDH	Write Data Hold time Write data hold time is WDH+2 clock cycles, and WDH must be even number. This field applied to SYNC mode only.
27-24 WDS	Write Data Setup time Write data setup time is WDS+2 clock cycles, and WDS must be even number. This field applied to SYNC mode only.
23-20 RDH	Read Data Hold time Read data hold time is RDH+2 clock cycles, and RDH must be even number. This field applied to SYNC mode only.
19-16 RDS	Read Data Setup time Read data setup time is RDS+2 clock cycles, and RDS must be even number. It provides the minimum timing control before data output phase. This field applied to SYNC mode only.
15-4 —	Reserved
3 —	Reserved
2	NAND option bit 3

Table continues on the next page...

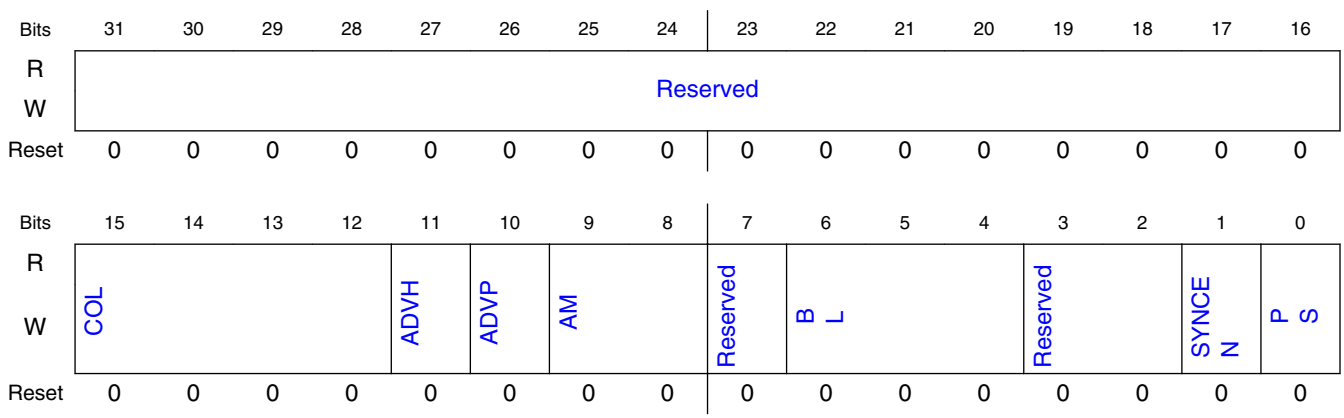
Field	Function
NDOPT3	This bit is intended to bypass row addres byte #1 for address mode 0x0 - Column and Row address(5 Byte-CA0/CA1/RA0/RA1/RA2).
1 NDOPT2	NAND option bit 2 This bit is intended to bypass row addres byte #2 for address mode 0x0 - Column and Row address(5 Byte-CA0/CA1/RA0/RA1/RA2).
0 NDOPT1	NAND option bit 1 This bit is intended to bypass column addres byte #1 for address mode 0x0 - Column and Row address(5 Byte-CA0/CA1/RA0/RA1/RA2). **NOTE:** Please refer to NAND Flash Access Address Type

25.7.1.18 NOR Control Register 0 (NORCR0)

25.7.1.18.1 Offset

Register	Offset
NORCR0	60h

25.7.1.18.2 Diagram



25.7.1.18.3 Fields

Field	Function
31-16 —	Reserved
15-12	Column Address bit width

Table continues on the next page...

Field	Function
COL	0000b - 12 Bits 0001b - 11 Bits 0010b - 10 Bits 0011b - 9 Bits 0100b - 8 Bits 0101b - 7 Bits 0110b - 6 Bits 0111b - 5 Bits 1000b - 4 Bits 1001b - 3 Bits 1010b - 2 Bits 1011b - 12 Bits 1100b - 12 Bits 1101b - 12 Bits 1110b - 12 Bits 1111b - 12 Bits
11 ADVH	ADV# level control during address hold state 0b - ADV# is high during address hold state. 1b - ADV# is low during address hold state.
10 ADVP	ADV# Polarity 0b - ADV# is active low. 1b - ADV# is active high.
9-8 AM	Address Mode It defines the address mode. 00b - Address/Data MUX mode (ADMUX) 01b - Advanced Address/Data MUX mode (AADM) 10b - Reserved 11b - Reserved
7 —	Reserved
6-4 BL	Burst Length The burst length determines the number of column locations that can be accessed for a given READ or WRITE command. 000b - 1 001b - 2 010b - 4 011b - 8 100b - 16 101b - 32 110b - 64 111b - 64
3-2 —	Reserved
1 SYNCEN	Synchronous Mode Enable It defines the access mode. 0b - Asynchronous mode is enabled. 1b - Synchronous mode is enabled. Only fixed latency mode is supported.
0 PS	Port Size The port size must be aligned with NOR Flash data width. 0b - 8bit 1b - 16bit

25.7.1.19 NOR Control Register 1 (NORCR1)

25.7.1.19.1 Offset

Register	Offset
NORCR1	64h

25.7.1.19.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	REH				REL				WEH				WEL			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	AH				AS				CEH				CES			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

25.7.1.19.3 Fields

Field	Function
31-28	RE high time
REH	RE high time is REH+1 clock cycles for ASYNC mode.
27-24	RE low time
REL	RE low time is REL+1 clock cycles for ASYNC mode.
23-20	WE high time
WEH	WE high time is WEH+1 clock cycles for ASYNC mode.
19-16	WE low time
WEL	WE low time is WEL+1 clock cycles for ASYNC mode.
15-12	Address hold time
AH	Address hold time is AH+1 clock cycles for ASYNC mode.
11-8	Address setup time
AS	Address setup time is AS+1 clock cycles
7-4	CE hold time
CEH	CE Hold time is CEH+1 clock cycles

Table continues on the next page...

Field	Function
	This field determines the minimum hold time. Actual hold time may be longer in order to wait all read data sampled.
3-0 CES	CE setup time CE setup time is CES+1 clock cycles.

25.7.1.20 NOR Control Register 2 (NORCR2)

25.7.1.20.1 Offset

Register	Offset
NORCR2	68h

25.7.1.20.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	RDH				CEITV				RD				LC			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	AWDH				TA				Reserved							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

25.7.1.20.3 Fields

Field	Function
31-28 RDH	Read hold time Read hold time is RDH clock cycles. This field applied to SYNC mode only.
27-24 CEITV	CE# interval time CE# interval minimum time is CEITV+1 clock cycles.
23-20 RD	Read time Read time is RD+1 clock cycles. This field applied to SYNC mode only.
19-16 LC	Latency count Latency count is LC clock cycles. This field applied to SYNC mode only.

Table continues on the next page...

Field	Function
15-12 AWDH	Address to write data hold time Address to write data hold time is AWDH clock cycles. This field applied to ASYNC mode only.
11-8 TA	Turnaround time Turnaround time is TA+1 clock cycles. Both the SEMC and Device do not drive Data Lines to avoid bus confliction. This field applied to ASYNC mode only.
7-0 —	Reserved

25.7.1.21 NOR Control Register 3 (NORCR3)

25.7.1.21.1 Offset

Register	Offset
NORCR3	6Ch

25.7.1.21.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								AHSR				ASSR			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

25.7.1.21.3 Fields

Field	Function
31-8 —	Reserved
7-4 AHSR	Address hold time for SYNC read Address hold time is AHSR clock cycles for SYNC read.
3-0 ASSR	Address setup time for SYNC read Address setup time ASSR+1 clock cycles for SYNC read.

25.7.1.22 SRAM Control Register 0 (SRAMCR0)

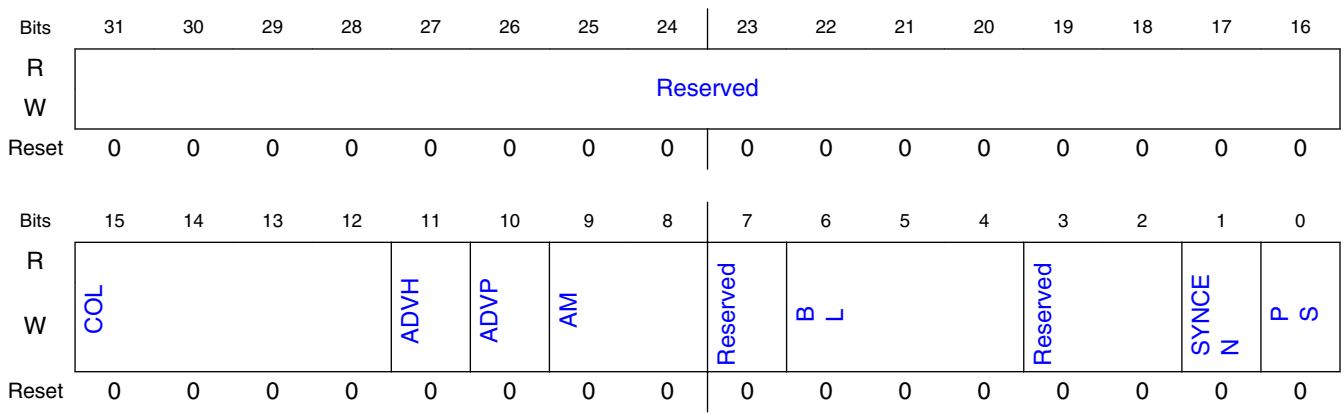
25.7.1.22.1 Offset

Register	Offset
SRAMCR0	70h

25.7.1.22.2 Function

This register configures SRAM device 0

25.7.1.22.3 Diagram



25.7.1.22.4 Fields

Field	Function
31-16 —	Reserved
15-12 COL	Column Address bit width 0000b - 12 Bits 0001b - 11 Bits 0010b - 10 Bits 0011b - 9 Bits 0100b - 8 Bits 0101b - 7 Bits 0110b - 6 Bits 0111b - 5 Bits 1000b - 4 Bits

Table continues on the next page...

Field	Function
	1001b - 3 Bits 1010b - 2 Bits 1011b - 12 Bits 1100b - 12 Bits 1101b - 12 Bits 1110b - 12 Bits 1111b - 12 Bits
11 ADVH	ADV# level control during address hold state 0b - ADV# is high during address hold state. 1b - ADV# is low during address hold state.
10 ADVP	ADV# polarity 0b - ADV# is active low. 1b - ADV# is active high.
9-8 AM	Address Mode It defines the address mode. 00b - Address/Data MUX mode (ADMUX) 01b - Advanced Address/Data MUX mode (AADM) 10b - Reserved 11b - Reserved
7 —	Reserved
6-4 BL	Burst Length The burst length determines the number of column locations that can be accessed for a given READ or WRITE command. 000b - 1 001b - 2 010b - 4 011b - 8 100b - 16 101b - 32 110b - 64 111b - 64
3-2 —	Reserved
1 SYNCEN	Synchronous Mode Enable It defines the access mode. 0b - Asynchronous mode is enabled. 1b - Synchronous mode is enabled. Only fixed latency mode is supported.
0 PS	Port Size The port size must be aligned with SRAM data width. 0b - 8bit 1b - 16bit

25.7.1.23 SRAM Control Register 1 (SRAMCR1)

25.7.1.23.1 Offset

Register	Offset
SRAMCR1	74h

25.7.1.23.2 Function

This register configures SRAM device 0

25.7.1.23.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	REH				REL				WEH				WEL			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	AH				AS				CEH				CES			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

25.7.1.23.4 Fields

Field	Function
31-28	RE high time
REH	RE high time is REH+1 clock cycles. This field applied to ASYNC mode only.
27-24	RE low time
REL	RE low time is REL+1 clock cycles. This field applied to ASYNC mode only.
23-20	WE high time
WEH	WE high time is WEH+1 clock cycles. This field applied to ASYNC mode only.
19-16	WE low time
WEL	WE low time is WEL+1 clock cycles. This field applied to ASYNC mode only.
15-12	Address hold time
AH	Address hold time is AH+1 clock cycles for ASYNC mode. Address hold time is AH clock cycles for SYNC mode.
11-8	Address setup time
AS	Address setup time is AS+1 clock cycles
7-4	CE hold time
CEH	CE Hold time is CEH+1 clock cycles

Table continues on the next page...

Field	Function
	This field determines the minimum hold time. Actual hold time may be larger in order to wait all read data sampled.
3-0 CES	CE setup time CE setup time is CES+1 clock cycles.

25.7.1.24 SRAM Control Register 2 (SRAMCR2)

25.7.1.24.1 Offset

Register	Offset
SRAMCR2	78h

25.7.1.24.2 Function

This register configures SRAM device 0

25.7.1.24.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	RDH				CEITV				RD				LC			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	AWDH				TA				WDH				WDS			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

25.7.1.24.4 Fields

Field	Function
31-28 RDH	Read hold time Read hold time is RDH clock cycles. This field applied to SYNC mode only.
27-24 CEITV	CE# interval time CE# interval minimum time is CEITV+1 clock cycles.
23-20	Read time

Table continues on the next page...

Field	Function
RD	Read time is RD+1 clock cycles. This field applied to SYNC mode only.
19-16 LC	Latency count Latency count is LC clock cycles. This field applied to SYNC mode only.
15-12 AWDH	Address to write data hold time Address to write data hold time is AWDH clock cycles. This field applied to ASYNC mode only.
11-8 TA	Turnaround time Turnaround time is TA+1 clock cycles. Both the SEMC and Device do not drive Data Lines to avoid bus confliction. This field applied to ASYNC mode only.
7-4 WDH	Write Data hold time Write data hold time is WDH clock cycles. This field applied to SYNC mode only.
3-0 WDS	Write Data setup time Write data setup time is WDS+1 clock cycles. This field applied to SYNC mode only.

25.7.1.25 SRAM Control Register 3 (SRAMCR3)

25.7.1.25.1 Offset

Register	Offset
SRAMCR3	7Ch

25.7.1.25.2 Function

This register configures SRAM device 0

25.7.1.25.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

25.7.1.25.4 Fields

Field	Function
31-0 —	Reserved

25.7.1.26 DBI-B Control Register 0 (DBICR0)

25.7.1.26.1 Offset

Register	Offset
DBICR0	80h

25.7.1.26.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	COL				Reserved					BL			Reserved			PS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

25.7.1.26.3 Fields

Field	Function
31-16 —	Reserved
15-12 COL	Column Address bit width 0000b - 12 Bits 0001b - 11 Bits 0010b - 10 Bits 0011b - 9 Bits 0100b - 8 Bits 0101b - 7 Bits 0110b - 6 Bits 0111b - 5 Bits

Table continues on the next page...

Field	Function
	1000b - 4 Bits 1001b - 3 Bits 1010b - 2 Bits 1011b - 12 Bits 1100b - 12 Bits 1101b - 12 Bits 1110b - 12 Bits 1111b - 12 Bits
11-7 —	Reserved
6-4 BL	Burst Length The burst length determines the number of column locations that can be accessed for a given READ or WRITE command. 000b - 1 001b - 2 010b - 4 011b - 8 100b - 16 101b - 32 110b - 64 111b - 64
3-1 —	Reserved
0 PS	Port Size The port size must be aligned with DBI data width. 0b - 8bit 1b - 16bit

25.7.1.27 DBI-B Control Register 1 (DBICR1)

25.7.1.27.1 Offset

Register	Offset
DBICR1	84h

25.7.1.27.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	REH2		REL2		CEITV				REH				REL			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	WEH				WEL				CEH				CES			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

25.7.1.27.3 Fields

Field	Function
31-30 REH2	RDX High Time bit [5:4] RDX High time is REH+1 clock cycles
29-28 REL2	RDX Low Time bit [5:4] RDX Low time is REL+1 clock cycles
27-24 CEITV	CSX interval time CSX interval minimum time is CEITV+1 clock cycles.
23-20 REH	RDX High Time bit [3:0] RDX High time is REH+1 clock cycles
19-16 REL	RDX Low Time bit [3:0] RDX Low time is REL+1 clock cycles
15-12 WEH	WRX High Time WRX High time is WEH+1 clock cycles
11-8 WEL	WRX Low Time WRX Low time is WEL+1 clock cycles
7-4 CEH	CSX Hold Time CSX Hold minimum time is CEH+1 clock cycles. This field determines the minimum hold time. Actual hold time may be larger in order to wait all read data sampled.
3-0 CES	CSX Setup Time CSX Setup time is CES+1 clock cycles

25.7.1.28 IP Command Control Register 0 (IPCR0)

25.7.1.28.1 Offset

Register	Offset
IPCR0	90h

25.7.1.28.2 Function

Slave address

25.7.1.28.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

25.7.1.28.4 Fields

Field	Function
31-0 SA	Slave address The slave address is used for any command from IP bus.

25.7.1.29 IP Command Control Register 1 (IPCR1)

25.7.1.29.1 Offset

Register	Offset
IPCR1	94h

25.7.1.29.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	NAND_EXT_ADDR								Reserved					DATSZ		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

25.7.1.29.3 Fields

Field	Function
31-16 —	Reserved
15-8 NAND_EXT_ADDR	NAND Extended Address NAND IP command address is extended to 40-bit when IPCMD's KEY is 0x5AA5. High 8-bit is from NAND_EXT_ADDR, and low 32-bit is from IPCR0.
7-3 —	Reserved
2-0 DATSZ	Data Size in Byte It controls the data size of any write/read command. When IP command is not a write/read operation, DATSZ field can be ignored. 000b - 4 001b - 1 010b - 2 011b - 3 100b - 4 101b - 4 110b - 4 111b - 4

25.7.1.30 IP Command Control Register 2 (IPCR2)

25.7.1.30.1 Offset

Register	Offset
IPCR2	98h

25.7.1.30.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved												BM3	BM2	BM1	BM0
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

25.7.1.30.3 Fields

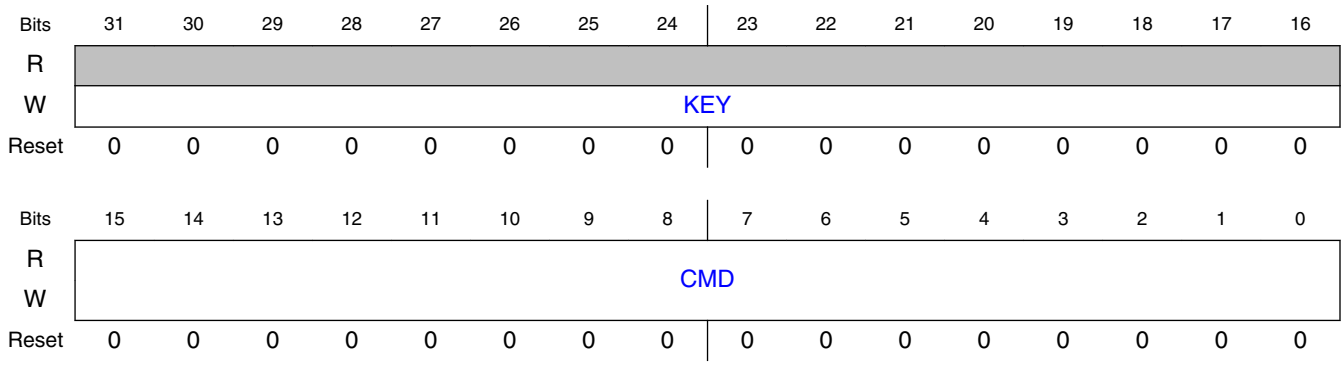
Field	Function
31-4 —	Reserved
3 BM3	Byte Mask for Byte 3 (IPTXDAT bit 31:24) 0b - Byte is unmasked 1b - Byte is masked
2 BM2	Byte Mask for Byte 2 (IPTXDAT bit 23:16) 0b - Byte is unmasked 1b - Byte is masked
1 BM1	Byte Mask for Byte 1 (IPTXDAT bit 15:8) 0b - Byte is unmasked 1b - Byte is masked
0 BM0	Byte Mask for Byte 0 (IPTXDAT bit 7:0) 0b - Byte is unmasked 1b - Byte is masked

25.7.1.31 IP Command Register (IPCMD)

25.7.1.31.1 Offset

Register	Offset
IPCMD	9Ch

25.7.1.31.2 Diagram



25.7.1.31.3 Fields

Field	Function
31-16 KEY	This field should be written with 0xA55A when trigging an IP command for all device types. The memory device is selected by the settings of IPCR0 and Base Registers. This field should be written with 0x5AA5 when trigging an IP command for NAND only. The address is extended to 40-bit with the use of NAND_EXT_ADDR. NAND IP base address is from 0x0 and memory size is not limited by BR8[MS].
15-0 CMD	SDRAM Commands: • 0x8: Read • 0x9: Write • 0xA: Mode Register Set • 0xB: Active • 0xC: Auto Refresh • 0xD: Self Refresh • 0xE: Precharge • 0xF: Precharge All • Others: Reserved **NOTE:** Self Refresh is sent to all SDRAM devices because they share the same SEMC_CLK pin. NAND Commands: Bit 15-8 (Command Code) Bit 7-4 (Address mode): • 0x0: Colum and Row address(5 Byte-CA0/CA1/RA0/RA1/RA2) • 0x1: Column address only (1 Byte-CA0) • 0x2: Column address only (2 Byte-CA0/CA1) • 0x3: Row address only (1 Byte-RA0) • 0x4: Row address only (2 Byte-RA0/RA1) • 0x5: Row address only (3 Byte-RA0/RA1/RA2) • Others: Reserved Bit 3-0 (Command mode): • 0x0: Command(0x05)-Address-Command-Read (Reserved for AXI Read) • 0x1: Command(0x85)-Address-Write (Reserved for AXI Write) • 0x2: Command • 0x3: Command-Hold • 0x4: Command-Address

Field	Function
	• 0x5: Command-Address-Hold • 0x6: Command-Address-Read • 0x7: Command-Address-Write • 0x8: Command-Read • 0x9: Command-Write • 0xA: Read • 0xB: Write • Others: Reserved NOR Commands: • 0x2: Read • 0x3: Write • Others: Reserved SRAM Commands: • 0x2: Memory Array Read • 0x3: Memory Array Write • 0x4: Memory Register Read • 0x5: Memory Register Write • Others: Reserved DBI_B Commands: • 0x2: Read • 0x3: Write • Others: Reserved

25.7.1.32 TX DATA Register (IPTXDAT)

25.7.1.32.1 Offset

Register	Offset
IPTXDAT	A0h

25.7.1.32.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DAT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DAT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

25.7.1.32.3 Fields

Field	Function
31-0 DAT	Data value to use for an IP write command. Bytes to use are determined based on IPCR2 configuration.

25.7.1.33 RX DATA Register (IPRXDAT)

25.7.1.33.1 Offset

Register	Offset
IPRXDAT	B0h

25.7.1.33.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DAT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DAT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

25.7.1.33.3 Fields

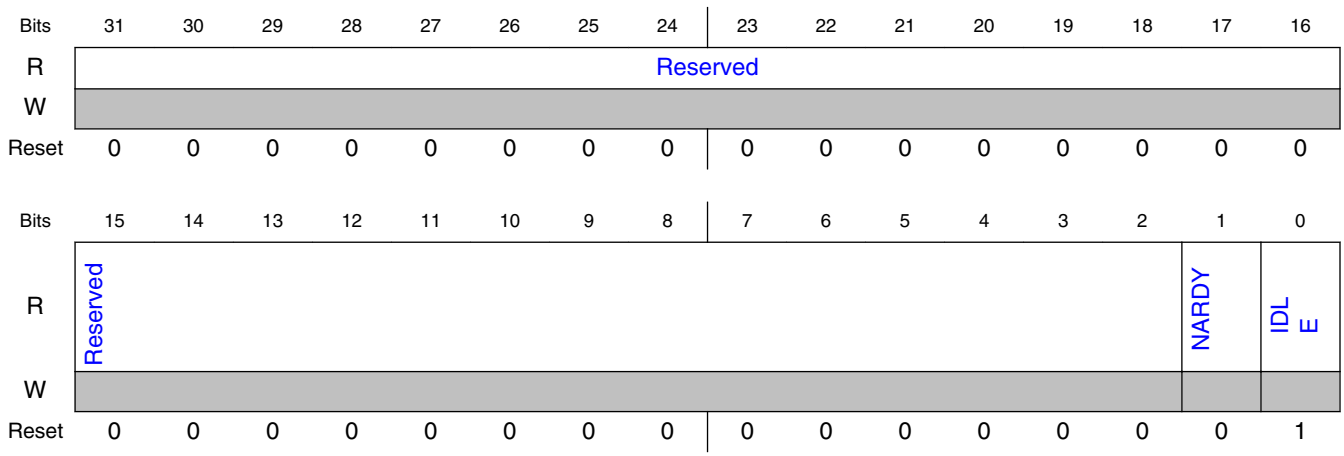
Field	Function
31-0 DAT	Data returned by device for an IP read command.

25.7.1.34 Status Register 0 (STS0)

25.7.1.34.1 Offset

Register	Offset
STS0	C0h

25.7.1.34.2 Diagram



25.7.1.34.3 Fields

Field	Function
31-2 —	Reserved
1 NARDY	Indicating NAND device Ready/WAIT# pin level. 0b - NAND device is not ready 1b - NAND device is ready
0 IDLE	Indicating whether the SEMC is in idle state. When IDLE=1, the SEMC is in idle state. There is no pending AXI command in internal queue and no pending device access.

25.7.1.35 Status Register 1 (STS1)

25.7.1.35.1 Offset

Register	Offset
STS1	C4h

25.7.1.35.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

25.7.1.35.3 Fields

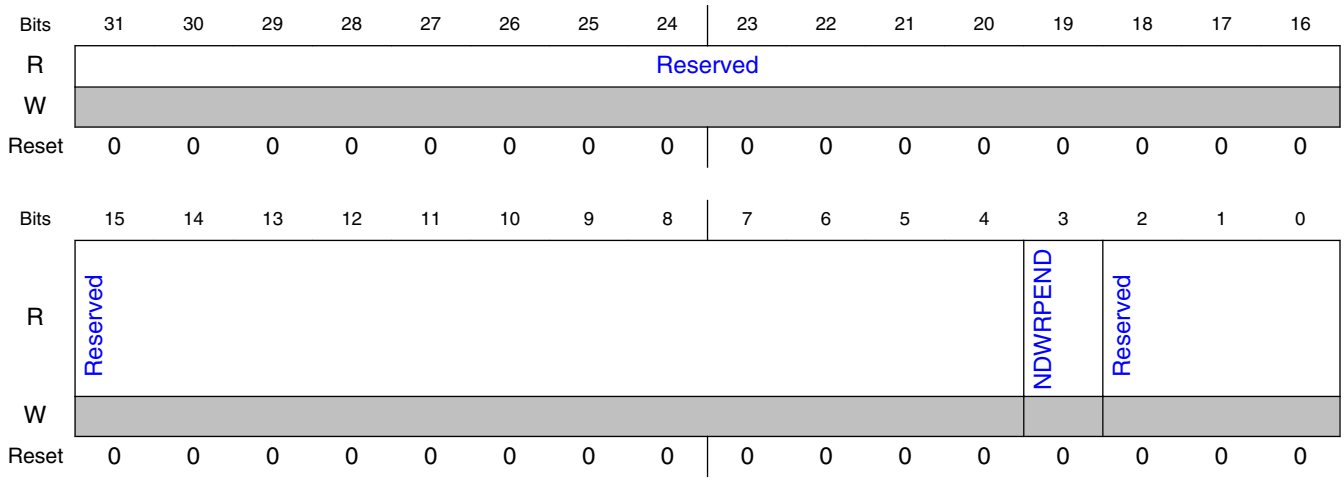
Field	Function
31-0 —	Reserved

25.7.1.36 Status Register 2 (STS2)

25.7.1.36.1 Offset

Register	Offset
STS2	C8h

25.7.1.36.2 Diagram



25.7.1.36.3 Fields

Field	Function
31-4 —	Reserved
3 NDWRPEND	This field indicating whether there is pending AXI command (write) to NAND device. 0b - No pending 1b - Pending
2-0 —	Reserved

25.7.1.37 Status Register 3 (STS3)

25.7.1.37.1 Offset

Register	Offset
STS3	CCh

25.7.1.37.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

25.7.1.37.3 Fields

Field	Function
31-0 —	Reserved

25.7.1.38 Status Register 4 (STS4)

25.7.1.38.1 Offset

Register	Offset
STS4	D0h

25.7.1.38.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

25.7.1.38.3 Fields

Field	Function
31-0 —	Reserved

25.7.1.39 Status Register 5 (STS5)

25.7.1.39.1 Offset

Register	Offset
STS5	D4h

25.7.1.39.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

25.7.1.39.3 Fields

Field	Function
31-0 —	Reserved

25.7.1.40 Status Register 6 (STS6)

25.7.1.40.1 Offset

Register	Offset
STS6	D8h

25.7.1.40.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

25.7.1.40.3 Fields

Field	Function
31-0 —	Reserved

25.7.1.41 Status Register 7 (STS7)

25.7.1.41.1 Offset

Register	Offset
STS7	DCh

25.7.1.41.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

25.7.1.41.3 Fields

Field	Function
31-0 —	Reserved

25.7.1.42 Status Register 8 (STS8)

25.7.1.42.1 Offset

Register	Offset
STS8	E0h

25.7.1.42.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

25.7.1.42.3 Fields

Field	Function
31-0 —	Reserved

25.7.1.43 Status Register 9 (STS9)

25.7.1.43.1 Offset

Register	Offset
STS9	E4h

25.7.1.43.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

25.7.1.43.3 Fields

Field	Function
31-0 —	Reserved

25.7.1.44 Status Register 10 (STS10)

25.7.1.44.1 Offset

Register	Offset
STS10	E8h

25.7.1.44.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

25.7.1.44.3 Fields

Field	Function
31-0 —	Reserved

25.7.1.45 Status Register 11 (STS11)

25.7.1.45.1 Offset

Register	Offset
STS11	ECh

25.7.1.45.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

25.7.1.45.3 Fields

Field	Function
31-0 —	Reserved

25.7.1.46 Status Register 12 (STS12)

25.7.1.46.1 Offset

Register	Offset
STS12	F0h

25.7.1.46.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	NDADDR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	NDADDR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

25.7.1.46.3 Fields

Field	Function
31-0 NDADDR	This field indicating the last write address (AXI command) to NAND device (without base address in SEMC_BR4).

25.7.1.47 Status Register 13 (STS13)

25.7.1.47.1 Offset

Register	Offset
STS13	F4h

25.7.1.47.2 Function

This register indicates the status of sample clock DLLs for synchronous NAND read access.

25.7.1.47.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved		REFSEL						SLVSEL						REFLOCK	SLVLOCK
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

25.7.1.47.4 Fields

Field	Function
31-14	Reserved

Table continues on the next page...

Field	Function
—	
13-8 REFSEL	Sample clock reference delay line delay cell number selection.
7-2 SLVSEL	Sample clock slave delay line delay cell number selection.
1 REFLOCK	Sample clock reference delay line locked. 0b - Reference delay line is not locked. 1b - Reference delay line is locked.
0 SLVLOCK	Sample clock slave delay line locked. 0b - Slave delay line is not locked. 1b - Slave delay line is locked.

25.7.1.48 Status Register 14 (STS14)

25.7.1.48.1 Offset

Register	Offset
STS14	F8h

25.7.1.48.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

25.7.1.48.3 Fields

Field	Function
31-0 —	Reserved

25.7.1.49 Status Register 15 (STS15)

25.7.1.49.1 Offset

Register	Offset
STS15	FCh

25.7.1.49.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

25.7.1.49.3 Fields

Field	Function
31-0 —	Reserved

Chapter 26
Ultra Secured Digital Host Controller (uSDHC)

26.1 Chip-specific uSDHC information

Table 26-1. Reference links to related information

Topic	Related module(s)	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
Interrupts, DMA Events and XBAR Assignments	-	Interrupts, DMA Events and XBAR Assignments

On this device, for data transfer modes, uSDHC1 only supports 1-bit and 4-bit modes, while uSDHC2 can support 1-bit, 4-bit and 8-bit modes.

NOTE

HS400 mode is not supported on this device.

26.2 Overview

The Ultra Secured Digital Host Controller (uSDHC) provides the interface between the host system and the SD/SDIO/MMC cards, as shown in Figure 26-2. The module acts as a bridge, passing host bus transactions to the SD/SDIO/MMC cards by sending commands and performing data accesses to/from the cards. It handles the SD/SDIO/MMC protocols at the transmission level.

26.2.1 Block Diagram

The following figure illustrates the block diagram for uSDHC.

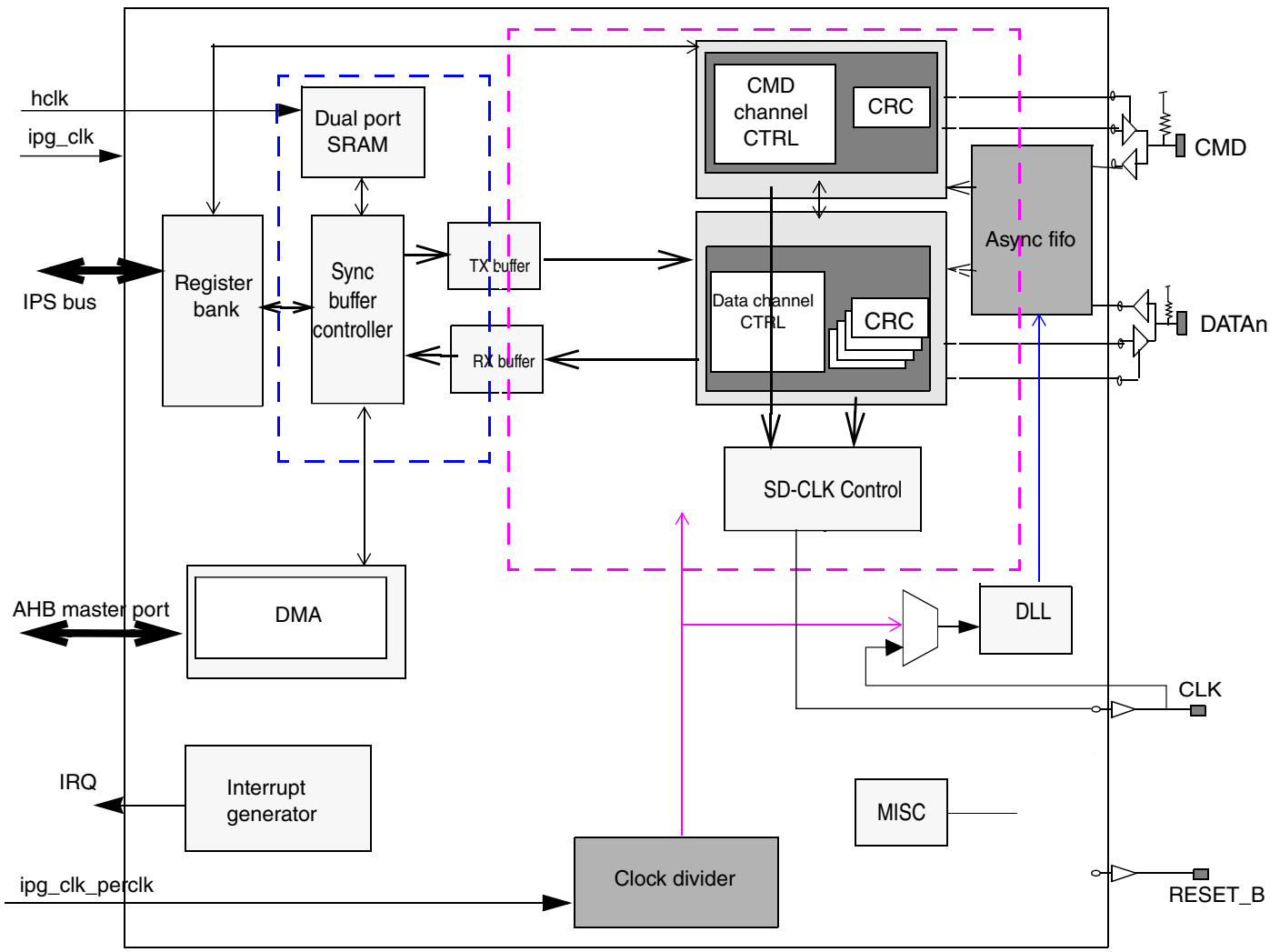


Figure 26-1. uSDHC block diagram

The figure below shows the System connection of uSDHC:

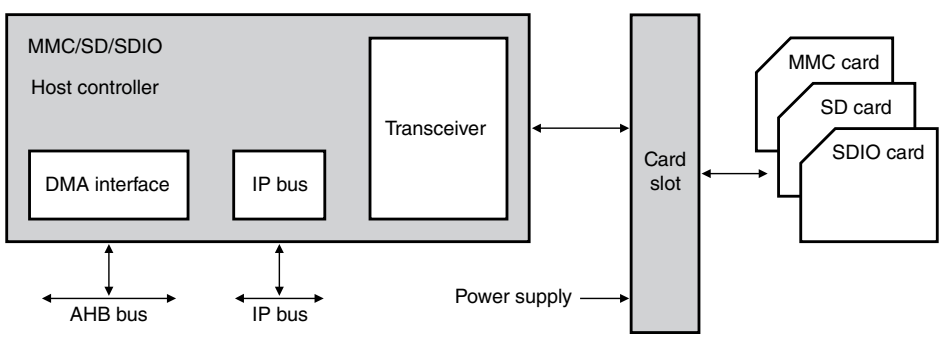


Figure 26-2. System connection of uSDHC

The following are brief descriptions of the cards supported by uSDHC:

- The Multi Media Card (MMC) is a universal low-cost data storage and communication media designed to cover a wide array of applications including mobile video and gaming. Previous MMC cards were based on a 7-pin serial bus with a single data pin, while the new high speed MMC communication is based on an advanced 11-pin serial bus designed to operate in the low-voltage range.
- The Secure Digital Card (SD) is an evolution of the old MMC technology. It is specifically designed to meet the security, capacity, performance, and environmental requirements inherent in newly emerging audio and video consumer electronic devices. The physical form factor, pin assignment, and data transfer protocol are backward compatible with the old MMC (with some additions).
- Under the SD protocol, system connection can be categorized into memory card, I/O card, and combo card, which have both memory and I/O functions. The memory card invokes a copyright protection mechanism that complies with the security of the SDMI standard. The I/O card, which is also known as SDIO card, provides high-speed data I/O with low-power consumption for mobile electronic devices. For the sake of simplicity, the next figure does not show cards with reduced size or mini cards.

26.2.2 Features

The list given below shows the features of the uSDHC module:

- Conforms to the SD Host Controller Standard Specification version 2.0/3.0
- Compatible with the MMC System Specification version 4.2/4.3/4.4/4.41/4.5
- Compatible with the SD Memory Card Specification version 3.0 and supports the Extended Capacity SD Memory Card
- Compatible with the SDIO Card Specification version 2.0/3.0
- Designed to work with SD Memory, miniSD Memory, SDIO, miniSDIO, SD Combo, MMC, MMC plus, and MMC RS cards
- Card bus clock frequency up to 208 MHz
- Supports 1-bit/4-bit SD and SDIO modes, and 1-bit/4-bit/8-bit MMC modes
  - Up to 832 Mbps of data transfer for SDIO cards using four parallel data lines in the Single Data Rate (SDR) mode
  - Up to 400 Mbps of data transfer for SDIO card using four parallel data lines in the Dual Data Rate (DDR) mode
  - Up to 832 Mbps of data transfer for SDXC cards using four parallel data lines in the Single Data Rate (SDR) mode
  - Up to 400 Mbps of data transfer for SDXC card using four parallel data lines in the Dual Data Rate (DDR) mode

  - Up to 1600 Mbps of data transfer for MMC cards using eight parallel data lines in the Single Data Rate (SDR) mode
  - Up to 832 Mbps of data transfer for MMC cards using eight parallel data lines in the Dual Data Rate (DDR) mode
- Supports single block/multi-block read and write
- Supports block sizes of 1 ~ 4096 bytes
- Supports both synchronous and asynchronous abort
- Supports pause during the data transfer at block gap
- Supports SDIO Read Wait and Suspend Resume operations
- Supports Auto CMD12 for multi-block transfer
- Host can initiate non-data transfer command while data transfer is in progress
- Allows cards to interrupt the host in 1-bit and 4-bit SDIO modes; also supports interrupt period
- Embodies a fully configurable 128x32-bit FIFO for read/write data
- Supports internal DMA capabilities
- Supports voltage selection by configuring vendor-specific register bit
- Supports Advanced DMA to perform linked memory access

NOTE

This block can support all the above-listed speed mode and maximum data throughput. However, these may be specific to the device. See the corresponding chip-specific information or the device data sheet for accurate details.

26.3 Functional description

The following sections provide a brief functional description of the major system blocks, including the data buffer, DMA AHB interface, register bank as well as IP Bus interface, dual-port memory wrapper, data/command controller, clock and reset manager, and clock generator.

26.3.1 Modes and operations

26.3.1.1 Data transfer modes

The uSDHC module can select the following modes for data transfer:

- SD 1-bit
- SD 4-bit

- MMC 1-bit
- MMC 4-bit
- MMC 8-bit
- Identification mode (up to 400 kHz)
- MMC full-speed mode (up to 26 MHz)
- MMC high-speed mode (up to 52 MHz)
- MMC HS200 mode(up to 200 MHz)
- MMC DDR mode (52 MHz both edges)
- SD/SDIO full-speed mode (up to 25 MHz)
- SD/SDIO high-speed mode (up to 50 MHz)
- SD/SDIO UHS-I mode(up to 208 MHz in SDR mode, up to 50 MHz in the DDR mode)

NOTE

This block can support all the above listed speed mode and maximum clock frequency. However, these may be specific to the device. See the corresponding chip-specific information or the device data sheet for accurate details.

26.3.2 Data buffer

The uSDHC module uses one configurable data buffer to transfer data between the system bus (IP bus or advanced high-performance bus (AHB) bus) and the SD card in an optimized manner, maximizing throughput between the two clock domains (IP peripheral clock and the master clock).

The buffer is used as a temporary storage for transferring data between the host system and the card. The watermark levels for read and write are both configurable and can range between 1 to 128 words. The burst lengths for read and write are also configurable and can range between 1 to 31 words. The next figure provides the uSDHC buffer scheme.

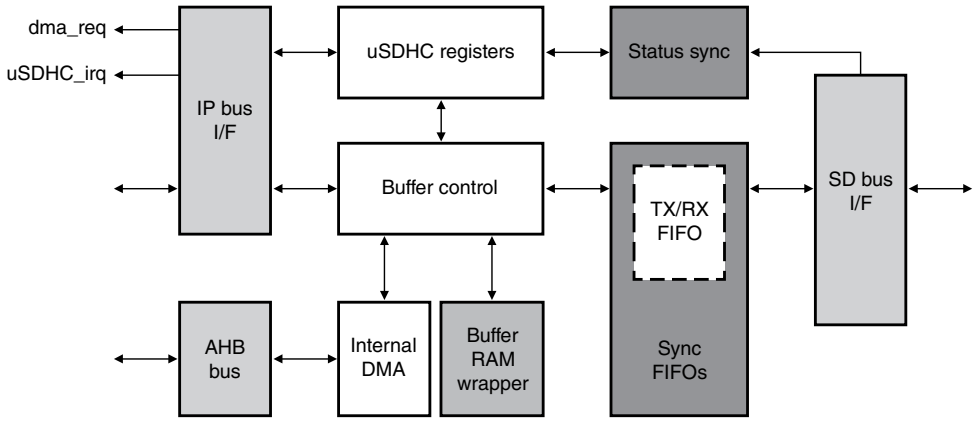


Figure 26-3. uSDHC buffer scheme

Here are 2 transfer modes to access the data buffer:

- CPU polling mode:
  - For a host-read operation, when the number of words received in the buffer meets or exceeds the RD_WML watermark value, by polling the BRR bit, the host driver can read the Buffer Data Port register to fetch the amount of words set in the RD_WML register from the buffer. The write operation is similar. For more information on the process of writing operation, see Write operation sequence.
- Internal DMA mode (includes simple and advanced DMA accesses):
  - The internal DMA access, either by simple or advanced DMA, is over the AHB bus.

For a read operation, when there are more words in the buffer than the amount set in the RD_WML register, the internal DMA starts fetching data over the AHB bus. Except for INCR4 and INCR8, the burst type is always the INCR mode and the burst length depends on the shortest of the following factors:

- Burst length configured in the burst length field of the Watermark Level register
- Watermark level boundary
- Block size boundary
- Data boundary configured in the current descriptor (if the ADMA is active)
- 1 KB address boundary defined in the AHB protocol

The Write operation functions in a similar manner—sequential and contiguous access is necessary to ensure that the pointer address value is correct. Random or skipped access is not possible. The byte order, by reset, is little endian mode. The actual byte order is swapped inside the buffer, according to the endian mode configured by software (see the following figures). For a host write operation, the byte order is swapped after the data is fetched from the buffer and ready to send to the SD Bus. For a host read operation, the byte order is swapped before the data is stored in the buffer.

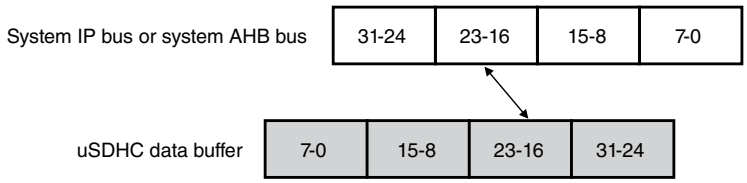


Figure 26-4. Data swap between system bus and uSDHC data buffer in the byte little endian mode

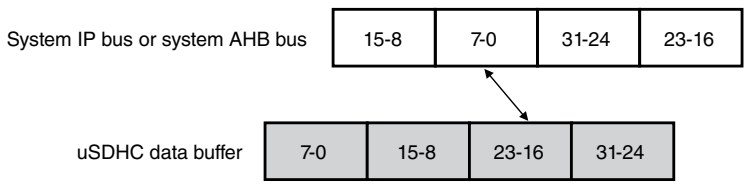


Figure 26-5. Data swap between system bus and uSDHC data buffer in half word big endian mode

26.3.2.1 Write operation sequence

There are 2 ways to write data into the buffer when the user transfers data to the card:

- Processor core polling through the BWR bit in the Interrupt Status register (interrupt or polling)
- Internal DMA

When the internal DMA is not used, the DMAEN bit in the Transfer Type register is not set when the command is sent, uSDHC asserts a DMA request when the amount of buffer space exceeds the value set in the WR_WML register and is ready for receiving new data. At the same time, uSDHC sets the BWR bit. The buffer write ready interrupt is generated if it is enabled by software.

When internal DMA is used, uSDHC does not inform the system before all the required number of bytes are transferred (if no error is encountered). When an error occurs during the data transfer, uSDHC aborts the data transfer and abandons the current block. The host driver should read the contents of the DMA System Address register to obtain the starting address of the abandoned data block. If the current data transfer is in multi-block mode, uSDHC does not automatically send CMD12, even though the AC12EN bit in the Transfer Type register is set. The host driver sends CMD12 in this scenario and restarts the write operation from that address. It is recommended that a software reset for Data be applied before the transfer is restarted.

The uSDHC module does not start data transmission until the buffer has been filled with the number of words set in the WR_WML register. If the buffer is empty and the host system does not write data in time, uSDHC stops the CLK to avoid the data buffer underrun situation.

26.3.2.2 Read operation sequence

There are 2 ways to read data from the buffer when the data is received from the card:

- Processor core polling through the BRR bit in the Interrupt Status register (interrupt or polling)
- Internal DMA

When internal DMA is not used (DMAEN bit in Transfer Type register is not set when the command is sent), uSDHC asserts a DMA request when the amount of data exceeds the value set in the RD_WML register, which is available and ready for system fetching data. At the same time, uSDHC sets the BRR bit. The buffer read ready interrupt is generated if it is enabled by the software.

When internal DMA is used, uSDHC does not inform the system before all the required number of bytes are transferred (if no error is encountered). When an error occurs during the data transfer, uSDHC aborts the data transfer and abandons the current block. The host driver should read the content of the DMA System Address register to get the starting address of the abandoned data block. If the current data transfer is in multi-block mode, uSDHC does not automatically send CMD12, even though the AC12EN bit in the Transfer Type register is set. The host driver sends CMD12 in this scenario and restarts the read operation from that address. It is recommended that a software reset (in register RSTD) for data be applied before the transfer is restarted.

For any write transfer mode, uSDHC does not start data transmission until the number of words set in the RD_WML register are in buffer. If the buffer is full and the host system does not read data in time, uSDHC stops the CLK to avoid the data buffer overrun situation.

26.3.2.3 Data buffer and block size

In the uSDHC module, the data buffer can hold up to 128 words (32-bit) and the watermark levels for write and read can be configured accordingly. The user needs to know the buffer size for the buffer operation during a data transfer to utilize it in the most optimized way.

For both read and write, the watermark level can range between 1 to 128 words. For both read and write, the burst length can range between from 1 to 31 words. The host driver may configure the value according to the system and requirement.

During a multi-block data transfer, the block length can be set to any value between 1 and 4096 bytes, satisfying the requirements of the external card. The only restriction is from the external card, which can be limited in size or support of a partial block access (which is not the integer times of 512 bytes).

As uSDHC treats each block individually, for block sizes which are not multiples of four (not word-aligned), stuffed bytes are required at the end of each block. For example, if the block size is 7 bytes and there are 12 blocks to write, the software side must write twice for each block. For each block, the ending byte is abandoned by uSDHC because it only sends 7 bytes to the card and picks data from the following system write, resulting in 24 beats of write access in total.

26.3.2.4 Dividing large data transfer

This SDIO command CMD53 definition limits the maximum data size of data transfers according to the following formula:

Max data size = Block size x Block count

The length of a multiple block transfer needs to be in multiples of block size. If the total data length cannot be divided evenly into a multiple of the block size, then there are two options to transfer the data. These two options depend on the function and the card design. Option 1 is for the host driver to split the transaction. The remainder of the block size data is then transferred by using a single block command at the end. Option 2 is to add dummy data in the last block to fill the block size. For option 2, the card must manage the removal of the dummy data.

See the figure below for an example showing the division of large data transfers, assuming a kind of WLAN SDIO card that only supports a block size of up to 64 bytes. Although uSDHC supports a block size of up to 4096 bytes, the SDIO can only accept a block size of less than 64 bytes, so the data must be divided (see the example below).

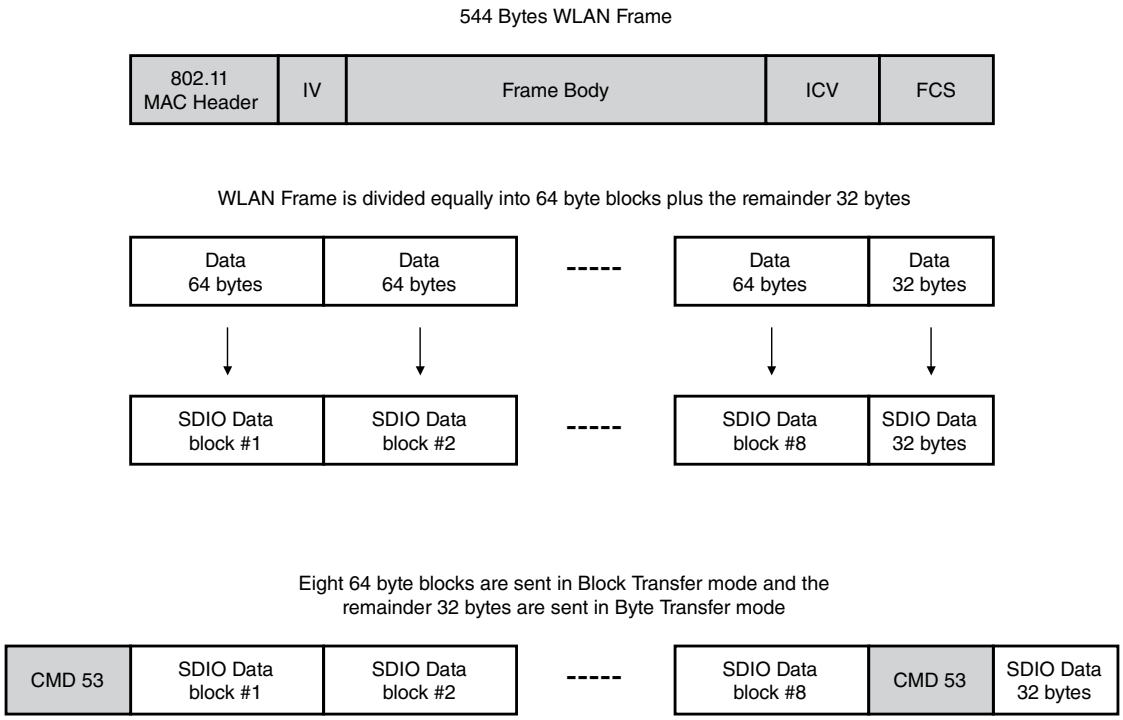


Figure 26-6. Example for dividing large data transfers

26.3.3 DMA AHB interface

The internal DMA implements a DMA engine and the AHB master. When the internal DMA is enabled, but the BWR and BRR bits are set if the BWRSEN and BRRSEN bits have been set in the Interrupt Status Enable register.

See the figure below for an illustration of the DMA AHB interface block.

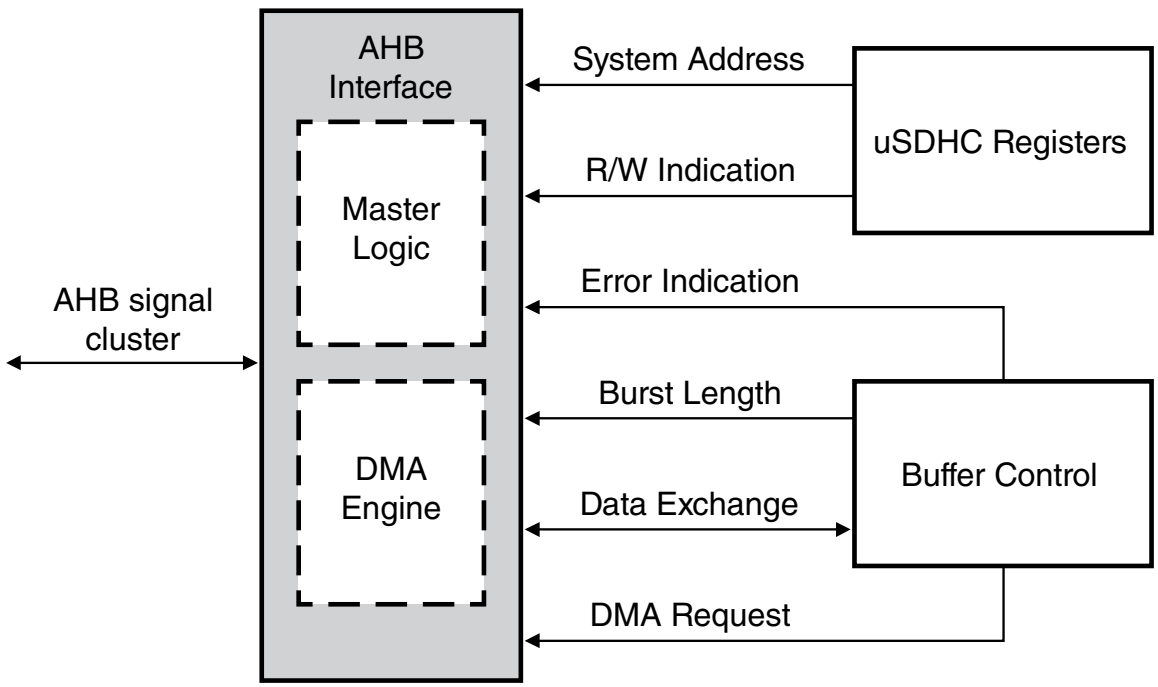


Figure 26-7. DMA AHB interface block

26.3.3.1 Internal DMA request

If the watermark level requirement is met in data transfer or if the last data of current block is ready in the data buffer, and the Internal DMA is enabled, the data buffer block sends a DMA request to AHB interface.

The delay in response from the internal DMA engine depends on the system AHB bus loading and the priority assigned to uSDHC. The DMA engine does not respond to the request during its burst transfer, but is ready to serve as soon as the burst is over. The data buffer de-asserts the request if the data buffer space (for write) or bytes in data buffer is smaller than the watermark level. Upon access to the buffer by internal DMA, the data buffer updates its internal buffer pointer, and when the watermark level is satisfied or the last data of the current block is ready in the data buffer, another DMA request is sent.

The data transfer is in the block unit, and the subsequent watermark level is always automatically set as the remaining number of words. For instance, for a multi block data read with each block size of 31 bytes, and the burst length set to six words (24 bytes). After the first burst transfer, if there are greater than or equal to two words in the buffer, which might contain some data of the next block), another DMA request is sent. This is because the remaining number of words to transfer for the current block is ceiling((31 - 6 * 4) / 4) = 2. The uSDHC module reads two words out of the buffer, with seven valid bytes and one stuffed byte.

26.3.3.2 DMA burst length

Just like a CPU polling access, the DMA burst length for the internal DMA engine can range between 1 to 16 words. The actual burst length for the DMA depends on the lesser of the configured burst length or the remaining words of the current block. See the example in Internal DMA request. After six words are read, the burst length is two words, then the next burst length is six words. This is because the next block starts, which is 31 bytes, more than six words. The host driver may take this variable burst length into account. It is also acceptable to configure the burst length as the divisor of the block size, so that each time the burst length is the same.

26.3.3.3 AHB master interface

It is possible that the internal AHB DMA engine could fail during the data transfer. Upon detection of an AHB bus error during DMA transfer, the DMA engine stops the transfer and goes to the idle state. At that point, the internal data buffer stops receiving incoming data and sending out data. The DMAE bit in the Interrupt Status register is generated to host CPU to report a bus error condition.

After the DMAE interrupt is received, the software sends a CMD12 to abort the current transfer and read the DS_ADDR bits of the DMA System Address register to get the starting address of the corrupted block. After the DMA error is fixed, the software should apply a data reset and restart the transfer from this address to recover the corrupted block.

26.3.3.4 ADMA engine

In the SD host controller standard, a new DMA transfer algorithm called the Advanced DMA (ADMA) is defined. For simple DMA, after the page boundary is reached, a DMA interrupt is generated and the new system address is programmed by the host driver.

The ADMA defines the programmable descriptor table in the system memory. The host driver can calculate the system address at the page boundary and program the descriptor table before executing ADMA. It reduces the frequency of interrupts to the host system. Therefore, higher speed DMA transfers could be realized because the host MCU intervention is not be needed during long DMA-based data transfers.

There are two types of ADMA in host controller: ADMA1 and ADMA2. ADMA1 can support data transfer of 4KB aligned data in system memory, and ADMA2 eliminates the restriction so that the data of any location and any size can be transferred in system memory. Their formats of Descriptor Table are different.

ADMA can recognize all kinds of descriptors defined in SD host controller Standard, and if the "End" flag is detected in the descriptor, ADMA stops after this descriptor is processed.

26.3.3.4.1 ADMA concept and descriptor format

ADMA1 includes the following descriptors:

- Valid/invalid descriptor
- Nop descriptor
- Set data length descriptor
- Set data address descriptor
- Link descriptor
- Interrupt flag and end flag in descriptor

ADMA2 includes the following descriptors:

- Valid/invalid descriptor
- Nop descriptor
- Rsv descriptor (new in ADMA2)
- Set data length and address descriptor
- Link descriptor
- Interrupt flag and end flag in descriptor

ADMA starts read/write operation after it reaches the tran state, using the data length and data address analyzed from most recent descriptor(s).

For ADMA1, the valid data length descriptor is the last set type descriptor before the tran type descriptor. Every tran type descriptor triggers a transfer, and the transfer data length is extracted from the most recent set type descriptor. If there is no set type descriptor after the previous Trans descriptor, the data length is the value for previous transfer, or 0 if no set descriptor is ever met.

For ADMA2, the tran type descriptor contains both data length and transfer data address, so the tran type descriptor itself can start a data transfer.

See the figure below for the format of the descriptor table for ADMA1.

Address/ Page Field		Address/ Page Field		Attribute Field					
31	12	11	6	5	4	3	2	1	0
Address or Data Length		000000		Act 2	Act 1	0	Int	End	Valid

Act 2	Act1	Symbol	Comment	31- 28	27- 12
0	0	Nop	No Operation	Don't Care	
0	1	Set	Set Data Length	0000	Data Length
1	0	Tran	Transfer Data	Data Address	
1	1	Link	Link Descriptor	Descriptor Address	

Valid	Valid = 1 indicates this line of descriptor is effective. If Valid = 0 generate ADMA Error Interrupt and stop ADMA.
End	End = 1 indicates current descriptor is the ending one.
Int	Int = 1 generates DMA Interrupt when this descriptor is processed.

Figure 26-8. Format of the ADMA1 descriptor table

See the figure below for the concept and access method of the descriptor table for ADMA1.

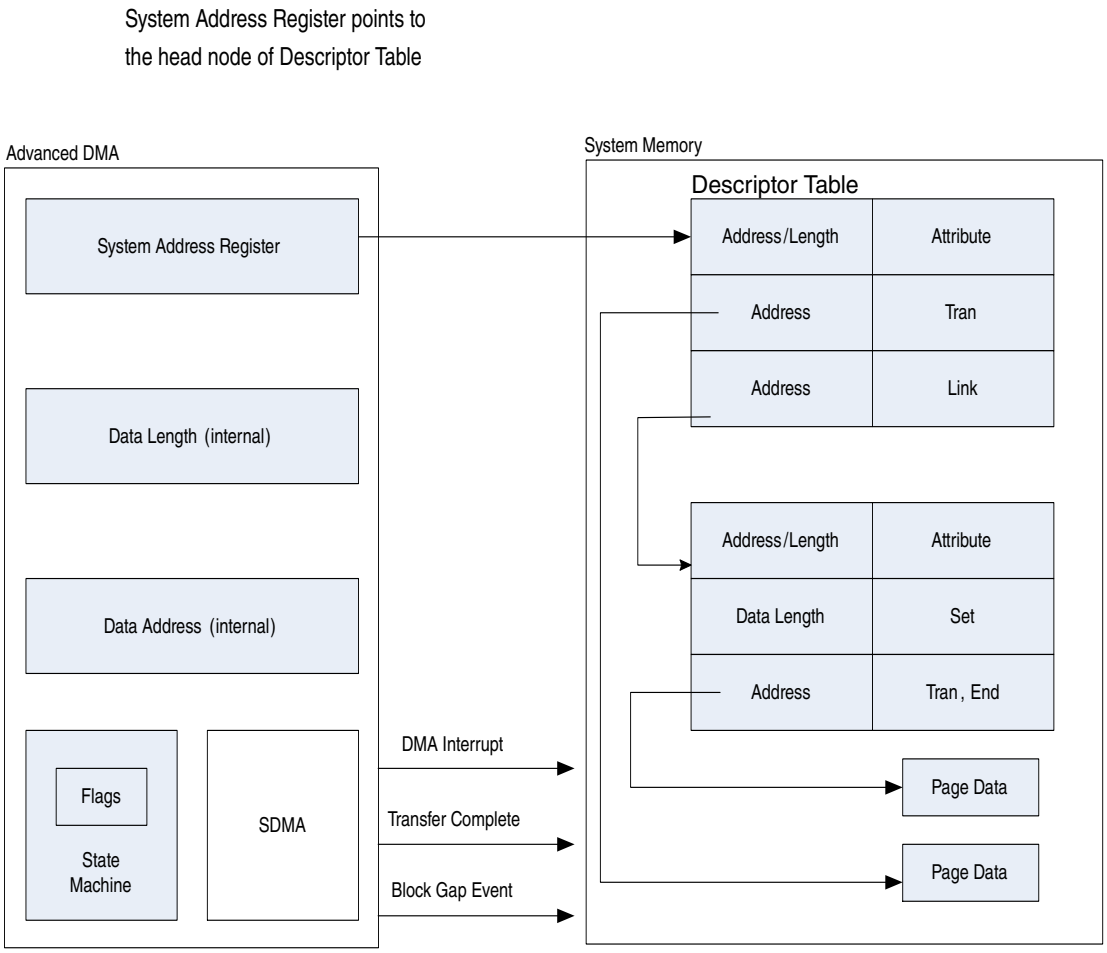


Figure 26-9. Concept and access method of the ADMA1 descriptor table

The figure below explains the ADMA2 format. ADMA2 deals with the lower 32-bit first, and then the higher 32-bit. If the 'Valid' flag of descriptor is 0, it ignores the higher 32-bit. The Address field should be set to word aligned (lower 2-bit is always set to 0). Data length is in byte unit.

Address Field		Length		Reserved		Attribute Field					
63	32	31	16	15	06	05	04	03	02	01	00
32-bit Address		16-bit length		0000000000		Act 2	Act 1	0	Int	End	Valid

Act 2	Act1	Symbol	Comment	Operation
0	0	Nop	No Operation	Don't Care
0	1	Rsv	Reserved	Same as Nop. Read this line and go to next one
1	0	Tran	Transfer Data	Transfer data with address and length set in this descriptor line
1	1	Link	Link Descriptor	Link to another descriptor

Valid	Valid = 1 indicates this line of descriptor is effective. If Valid = 0 generate ADMA Error Interrupt and stop ADMA.
End	End = 1 indicates current descriptor is the ending one.
Int	Int = 1 generates DMA Interrupt when this descriptor is processed.

Figure 26-10. Format of the ADMA2 descriptor table

See the figure below for the concept and access method of the descriptor table for ADMA2.

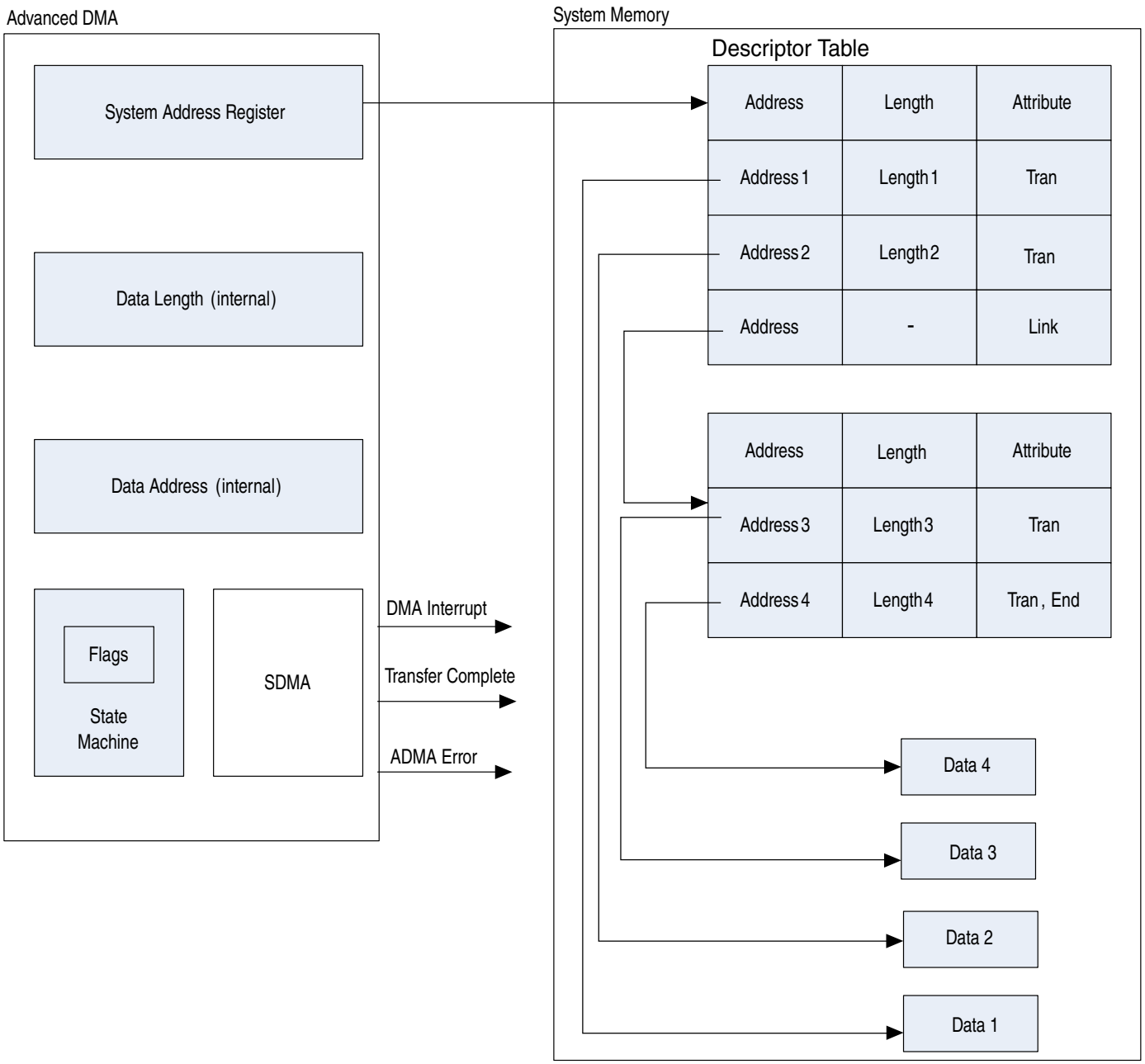


Figure 26-11. Concept and access method of the ADMA2 descriptor table

26.3.3.4.2 ADMA interrupt

If the interrupt flag descriptor is set, ADMA generates an interrupt according to various types of descriptors.

For ADMA1:

- Set type of descriptor: interrupt is generated when data length is set.
- Tran type descriptor: interrupt is generated when this transfer is complete.
- Link type of descriptor: interrupt is generated when new descriptor address is set.
- Nop type of descriptor: interrupt is generated just after this descriptor is fetched.

For ADMA2:

- Tran type of descriptor: interrupt is generated when this transfer is complete.
- Link type of descriptor: interrupt is generated when new descriptor address is set.
- Nop/Rsv type of descriptor: interrupt is generated just after this descriptor is fetched.

26.3.3.4.3 ADMA error

The ADMA stops whenever an error is encountered. These errors include:

- Fetching descriptor error
- AHB response error
- Data length mismatch error

An ADMA descriptor error is generated when it fails to detect a "valid" flag in the descriptor. If an ADMA descriptor error occurs, the interrupt is not generated even if the "Interrupt" flag of this descriptor is set.

When BLKCNTEN bit is set, data length set in register BLK_ATT must be equal to the whole data length set in descriptor, otherwise data length mismatch error is generated.

When BLKCNTEN bit is not set, then the whole data length set in descriptor is a multiple of block lengths; otherwise, when data set in the descriptor nodes are not performed at block boundaries, then data mismatch errors occur.

26.3.4 Register bank with IP bus interface

Register accesses through the IP bus interface are on the register bank.

See the figure below for the block diagram.

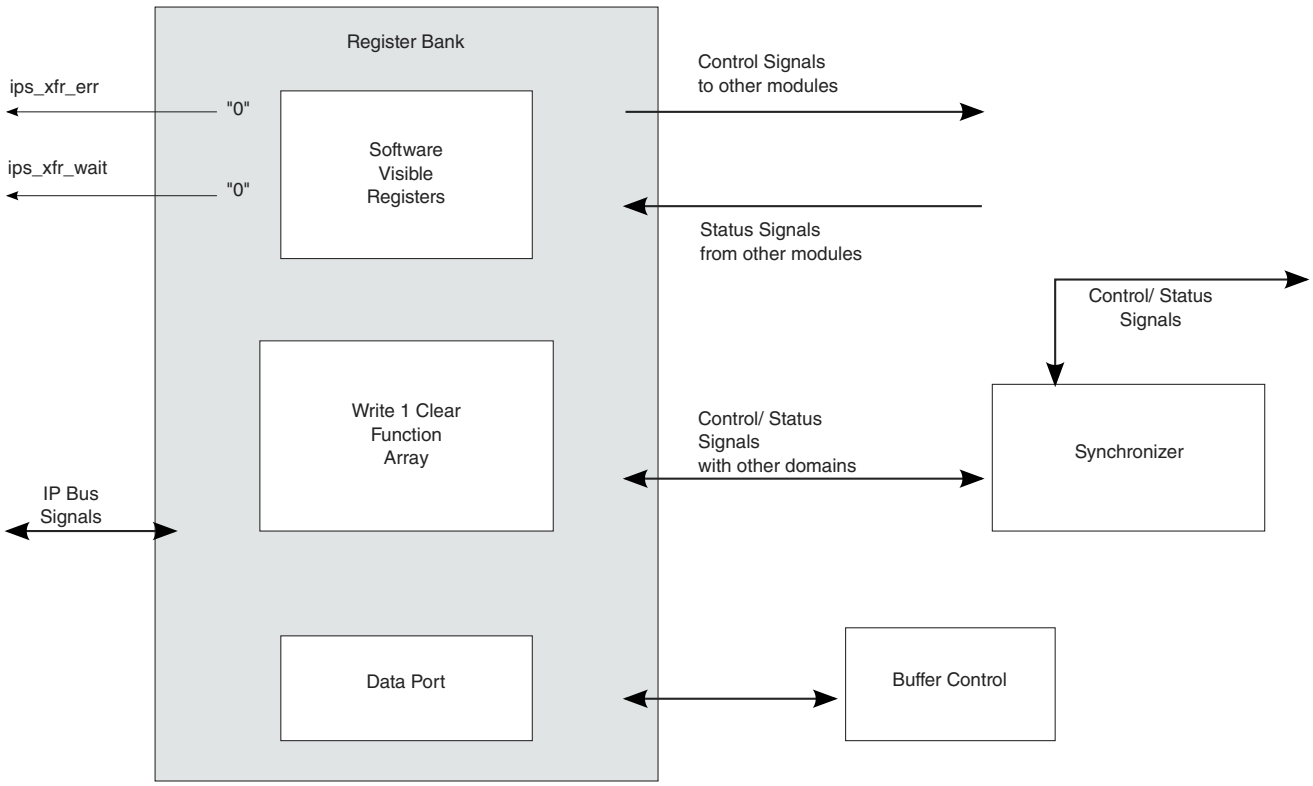


Figure 26-12. Register bank diagram

Only a 32-bit access is allowed, and no partial read/write is supported; therefore, all accesses are word aligned.

Access to an unimplemented address within the register memory map space does not generate a transfer error.

26.3.4.1 SD protocol unit

The SD protocol unit deals with all SD protocol affairs.

The SD protocol unit performs the following functions:

- Acts as the bridge between the internal buffer and the SD bus
- Sends the command data as well as its argument serially
- Stores the serial response bit stream into corresponding registers
- Detects the bus state on the CMD/DAT lines
- Monitors the interrupt from the SDIO card
- Asserts the read wait signal
- Gates off the SD clock when buffer is announcing danger status

The SD protocol unit consists of four submodules:

- SD control misc
- Command control
- Data control
- Clock control

26.3.4.2 SD control misc

In the SD control misc unit:

- The card detection (including DATA3 for card detection) and card interrupt are implemented.
- The module monitors the signal level on all the eight data lines and the command lines. It directly routes the level values into the register bank.

26.3.4.3 SD clock control

If the internal data buffer is near full (for read) or near empty (for write), the SD clock must be gated off to avoid buffer over/under-run. This module asserts the gate of the output SD clock to shut the clock off. After the buffer has space (for read) or has data (for write), the clock gate of this module opens, and the SD clock is active again.

26.3.4.4 Command control

The Command Control module deals with the transactions on the CMD line.

See the figure below for an illustration of the structure for the Command CRC shift register.

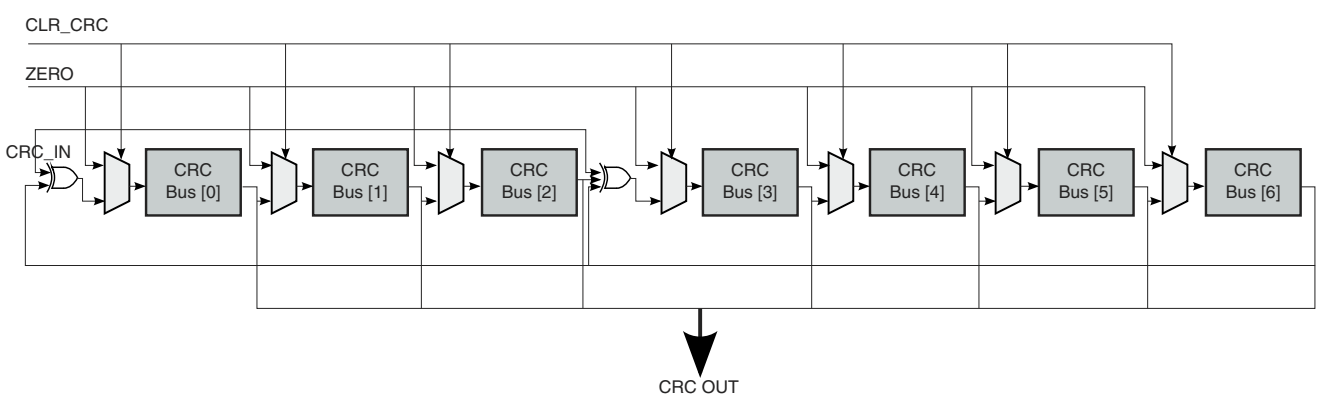


Figure 26-13. Command CRC shift register

The CRC polynomials for the CMD are as follows:

Generator polynomial: $G(x) = x^7 + x^3 + 1$
$M(x) = (\text{first bit}) * x^n + (\text{second bit}) * x^{n-1} + ... + (\text{last bit}) * x^0$
$CRC[6:0] = \text{Remainder } [(M(x) * x^7) / G(x)]$

26.3.4.5 Data control

The data agent deals with the transactions on the eight data lines. Moreover, this module also detects the busy state on the DATA0 line and generates the read wait state by the request from the transceiver.

The CRC polynomials for the data are as follows:

Generator polynomial: $G(x) = x^{16} + x^{12} + x^5 + 1$

$M(x) = (\text{first bit}) * x^n + (\text{second bit}) * x^{n-1} + ... + (\text{last bit}) * x^0$

$CRC[15:0] = \text{Remainder } [(M(x) * x^{16}) / G(x)]$

26.3.5 Card insertion and removal detection

The uSDHC module uses either the DATA3 pin or the to detect card insertion or removal. When there is no card on the MMC/SD bus, the DATA3 is pulled to a low voltage level by default.

When any card is inserted to or removed from the socket, uSDHC detects the logic value changes on the DATA3 pin and generates an interrupt.

26.3.6 Power management and wakeup events

When there is no operation between uSDHC and the card through the SD bus, the user can completely disable the peripheral clock and base clock in the chip-level clock control module to save power. When the user needs to use uSDHC to communicate with the card, it can enable the clock and start the operation.

In some circumstances, when the clocks to uSDHC are disabled, for instance, when the system is in low-power mode, there are some events for which the user needs to enable the clock and handle the event. These events are called wakeup interrupts. The uSDHC module can generate these interrupts even when there are no clocks enabled. The three interrupts that can be used as wakeup events are these:

- Card removal interrupt
- Card insertion interrupt
- Interrupt from SDIO card (not available in multiple block data transfers)

These three wakeup events (or wakeup interrupts) can also be used to wakeup the system from low-power modes.

NOTE

To make the interrupt a wakeup event, when all the clocks to uSDHC are disabled or when the entire system is in low power mode, the corresponding wakeup enabled bit needs to be set. Refer to Protocol Control (PROT_CTRL) for more information on the uSDHC Protocol Control register.

26.3.6.1 Setting wakeup events

For uSDHC to respond to a wakeup event, the software must set the respective wakeup enable bit before the CPU enters the sleep mode.

Before the software disables the host clock, it should ensure that all the following conditions have been met:

- No read or write transfer is active
- Data and command lines are not active
- No interrupts are pending
- Internal data buffer is empty

26.3.7 MMC fast boot

The Embedded MultiMediaCard (eMMC4.3) specification adds a fast boot feature that requires hardware support. There are two types of fast boot modes in the eMMC4.3 specification: boot operation and alternative boot operation. Each type also has with-acknowledge and without-acknowledge modes.

In the boot operation mode, the master (MultiMediaCard host) can read boot data from the slave (MMC device) by keeping CMD line low after power-on, or sending CMD0 with argument + 0xFFFFFFFA (optional for slave), before issuing CMD1.

NOTE

For the eMMC4.3 card setting, please see the eMMC4.3 specification.

26.3.7.1 Boot operation

If the CMD line is held low for 74 clock cycles and more after power-up before the first command is issued, the slave recognizes that boot mode is being initiated and starts preparing boot data internally.

Within one second after the CMD line goes low, the slave starts to send the first boot data to the master on the DATA line(s). The master must keep the CMD line low to read all of the boot data.

NOTE

For the purposes of this documentation, fast boot is called "normal fast boot mode".

If boot acknowledge is enabled, the slave has to send acknowledge pattern '010' to the master within 50ms after the CMD line goes low. If boot acknowledge is disabled, the slave does not send out acknowledge pattern '010'.

The master can terminate the boot mode with the CMD line high.

The boot operation is terminated when all the contents of the enabled boot data are sent to the master. After the boot operation is executed, the slave gets ready for the CMD1 operation and the master needs to start a normal MMC initialization sequence by sending CMD1.

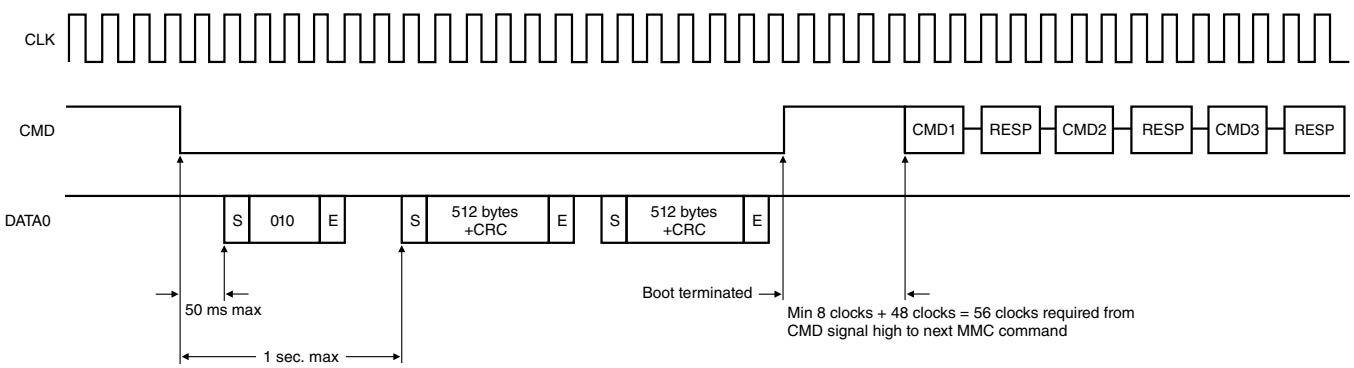


Figure 26-14. MultiMediaCard state diagram (normal boot mode)

26.3.7.2 Alternative boot operation

This boot function is optional for the device. If bit 0 in the extended CSD byte[228] is set to '1', the device supports the alternative boot operation.

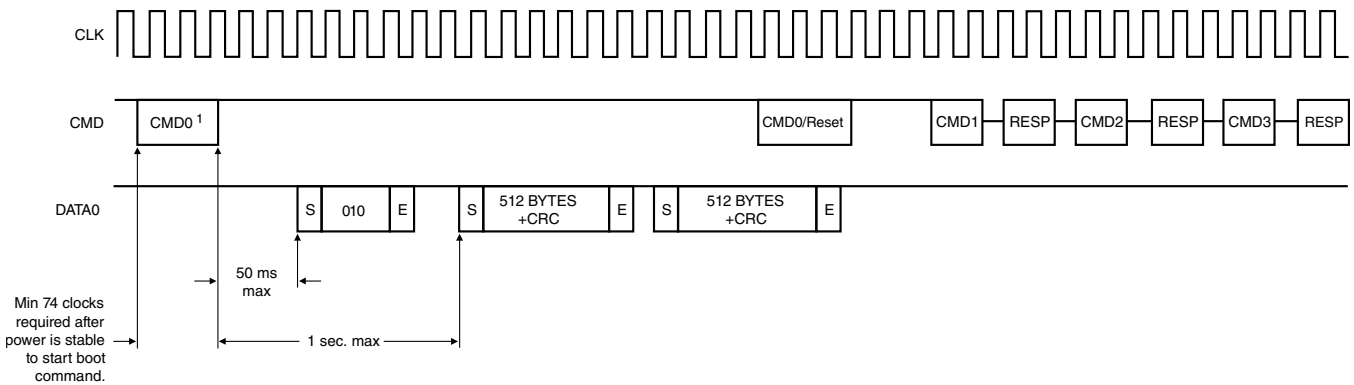
After power-up, if the host issues CMD0 with the argument of 0xFFFFFFFA after 74 clock cycles, before CMD1 is issued or the CMD line goes low, the slave recognizes that boot mode is being initiated and starts preparing boot data internally.

Within one second after CMD0 with the argument of 0xFFFFFFFA is issued, the slave starts to send the first boot data to the master on the DATA line(s).

If boot acknowledge is enabled, the slave must send the acknowledge pattern '010' to the master within 50ms after the CMD0 with the argument of 0xFFFFFFFA is received. If boot acknowledge is disabled, the slave does not send out acknowledge pattern '010'.

The master can terminate the boot mode by issuing CMD0 (Reset).

Boot operation is terminated when all the contents of the enabled boot data are sent to the master. After the boot operation is executed, the slave gets ready for the CMD1 operation and the master needs to start a normal MMC initialization sequence by sending CMD1.



NOTE 1: CMD0 with argument 0xFFFFFFFA

Figure 26-15. MultiMediaCard state diagram (alternative boot mode)

26.3.8 Commands for MMC/SD/SDIO

A table containing the list of commands for the MMC/SD/SDIO cards is provided here.

Refer to the corresponding specifications for more details about the command information.

There are four kinds of commands defined to control the MultiMediaCard:

- Broadcast commands (bc), no response
- Broadcast commands with response (bcr), response from all cards simultaneously
- Addressed (point-to-point) commands (ac), no data transfer on the DATA
- Addressed (point-to-point) data transfer commands (adtc)

Response: A response is a token that is sent from the card to the host as an answer to a previously received command. A response is transferred serially on the CMD line.

Table 26-2. Commands for MMC/SD/SDIO cards

CMD INDEX	Type	Argument	Response type	Abbreviation	Description
CMD0	bc	[31:0] stuff bits	-	GO_IDLE_STATE	Resets all MMC and SD memory cards to idle state.
CMD1	bcr	[31:0] OCR without busy	R3	SEND_OP_COND	Asks all MMC and SD Memory cards in idle state to send their operation conditions register contents in the response on the CMD line.
CMD2	bcr	[31:0] stuff bits	R2	ALL_SEND_CID	Asks all cards to send their CID numbers on the CMD line.
CMD3[1]	ac	[31:6] RCA [15:0] stuff bits	R1 R6 (SDIO)	SET/ SEND_RELATIVE_ADDR	Assigns relative address to the card.
CMD4	bc	[31:0] DSR [15:0] stuff bits	-	SET_DSR	Programs the DSR of all cards.
CMD5	bc	[31:0] OCR without busy	R4	IO_SEND_OP_COND	Asks all SDIO cards in idle state to send them operation conditions register contents in the response on the CMD line.
CMD6[2]	adtc	[31] Mode 0: Check function 1: Switch function [30:8] Reserved for function groups 6 ~ 3 (All 0 or 0xFFFF) [7:4] Function group1 for command system [3:0] Function group2 for access mode	R1	SWITCH_FUNC	Checks switch ability (mode 0) and switch card function (mode 1). Refer to "SD Physical Specification V1.1" for more details.
CMD6[3]	ac	[31:26] Set to 0 [25:24] Access [23:16] Index [15:8] Value [7:3] Set to 0 [2:0] Cmd Set	R1b	SWITCH	Switches the mode of operation of the selected card or modifies the EXT_CSD registers. Refer to "The MultiMediaCard System Specification Version 4.0 Final draft 2" for more details.
CMD7	ac	[31:6] RCA [15:0] stuff bits	R1b	SELECT/ DESELECT_CARD	Toggles a card between the stand-by and transfer states or between the programming and disconnect states. In both cases, the card is selected by its own relative address and gets deselected by any other address. Address 0 deselects all.
CMD8	adtc	[31:0] stuff bits	R1	SEND_EXT_CSD	The card sends its EXT_CSD register as a block of data, with a block size of 512 bytes.

Table continues on the next page...

Table 26-2. Commands for MMC/SD/SDIO cards (continued)

CMD INDEX	Type	Argument	Response type	Abbreviation	Description
CMD9	ac	[31:6] RCA [15:0] stuff bits	R2	SEND_CSD	Addressed card sends its card-specific data (CSD) on the CMD line.
CMD10	ac	[31:6] RCA [15:0] stuff bits	R2	SEND_CID	Addressed card sends its card-identification (CID) on the CMD line.
CMD11	adtc	[31:0] data address	R1	READ_DAT_UNTIL_STOP	Reads data stream from the card, starting at the given address, until a STOP_TRANSMISSION follows.
CMD12	ac	[31:0] stuff bits	R1b	STOP_TRANSMISSION	Forces the card to stop transmission.
CMD13	ac	[31:6] RCA [15:0] stuff bits	R1	SEND_STATUS	Addressed card sends its status register.
CMD14	Reserved				
CMD15	ac	[31:6] RCA [15:0] stuff bits	-	GO_INACTIVE_STATE	Sets the card to inactive state in order to protect the card stack against communication breakdowns.
CMD16	ac	[31:0] block length	R1	SET_BLOCKLEN	Sets the block length (in bytes) for all following block commands (read and write). Default block length is specified in the CSD.
CMD17	adtc	[31:0] data address	R1	READ_SINGLE_BLOCK	Reads a block of the size selected by the SET_BLOCKLEN command.
CMD18	adtc	[31:0] data address	R1	READ_MULTIPLE_BLOCK	Continuously transfers data blocks from card to host until interrupted by a stop command.
CMD19	adtc	[31:0] reserved bits(all 0)	R1	SEND_TUNING_BLOCK	64 bytes tuning pattern is sent for SDR50 and SDR104.
CMD20	adtc	[31:0] data address	R1	WRITE_DAT_UNTIL_STOP	Writes data stream from the host, starting at the given address, until a STOP_TRANSMISION follows.
CMD21	adtc	[31:0] stuff bits	R1	SEND_TUNING_BLOCK	128 clocks of tuning pattern (64 byte in 4 bit mode or 128 byte in 8 bit mode) is sent for HS200 optimal sampling point detection.
CMD22-23	Reserved				
CMD24	adtc	[31:0] data address	R1	WRITE_BLOCK	Writes a block of the size selected by the SET_BLOCKLEN command.
CMD25	adtc	[31:0] data address	R1	WRITE_MULTIPLE_BLOCK	Continuously writes blocks of data until a STOP_TRANSMISSION follows.
CMD26	adtc	[31:0] stuff bits	R1	PROGRAM_CID	Programming of the card identification register. This command is issued only once per card. The card contains hardware to prevent this operation after the

Table continues on the next page...

Table 26-2. Commands for MMC/SD/SDIO cards (continued)

CMD INDEX	Type	Argument	Response type	Abbreviation	Description
					first programming. Normally this command is reserved for the manufacturer.
CMD27	adtc	[31:0] stuff bits	R1	PROGRAM_CSD	Programming of the programmable bits of the CSD.
CMD28	ac	[31:0] data address	R1b	SET_WRITE_PROT	If the card has write protection features, this command sets the write protection bit of the addressed group. The properties of write protection are coded in the card specific data (WP_GRP_SIZE).
CMD29	ac	[31:0] data address	R1b	CLR_WRITE_PROT	If the card provides write protection features, this command clears the write protection bit of the addressed group.
CMD30	adtc	[31:0] write protect data address	R1	SEND_WRITE_PROT	If the card provides write protection features, this command asks the card to send the status of the write protection bits.
CMD31	Reserved				
CMD32	ac	[31:0] data address	R1	TAG_SECTOR_START	Sets the address of the first sector of the erase group.
CMD33	ac	[31:0] data address	R1	TAG_SECTOR_END	Sets the address of the last sector in a continuous range within the selection of a single sector to be selected for erase.
CMD34	ac	[31:0] data address	R1	UNTAG_SECTOR	Removes one previously selected sector from the erase selection.
CMD35	ac	[31:0] data address	R1	TAG_ERASE_GROUP_START	Sets the address of the first erase group within a range to be selected for erase.
CMD36	ac	[31:0] data address	R1	TAG_ERASE_GROUP_END	Sets the address of the last erase group within a continuous range to be selected for erase.
CMD37	ac	[31:0] data address	R1	UNTAG_ERASE_GROUP	Removes one previously selected erase group from the erase selection.
CMD38	ac	[31:0] stuff bits	R1b	ERASE	Erase all previously selected sectors.
CMD39	ac	[31:0] RCA [15] register write flag [14:8] register address [7:0] register data	R4	FAST_IO	Used to write and read 8-bit (register) data fields. The command addresses a card, and a register, and provides the data for writing if the write flag is set. The R4 response contains data read from the address register. This

Table continues on the next page...

Table 26-2. Commands for MMC/SD/SDIO cards (continued)

CMD INDEX	Type	Argument	Response type	Abbreviation	Description
					command accesses application dependent registers which are not defined in the MMC standard.
CMD40	bcr	[31:0] stuff bits	R5	GO_IRQ_STATE	Sets the system into interrupt mode.
CMD41	Reserved				
CDM42	adtc	[31:0] stuff bits	R1b	LOCK_UNLOCK	Used to set/reset the password or lock/unlock the card. The size of the data block is set by the SET_BLOCK_LEN command.
CMD43~51	Reserved				
CMD52	ac	[31:0] stuff bits	R5	IO_RW_DIRECT	Access a single register within the total 128k of register space in any I/O function.
CMD53	ac	[31:0] stuff bits	R5	IO_RW_EXTENDED	Accesses a multiple I/O register with a single command. Allows the reading or writing of a large number of I/O registers.
CMD54	Reserved				
CMD55	ac	[31:16] RCA [15:0] stuff bits	R1	APP_CMD	Indicates to the card that the next command is an application specific command rather than a standard command.
CMD56	adtc	[31:1] stuff bits [0]: RD/WR	R1b	GEN_CMD	Used either to transfer a data block to the card or to get a data block from the card for general purpose / application specific commands. The size of the data block is set by the SET_BLOCK_LEN command.
CMD57-59	Reserved				
CMD60	adtc	[31] WR [30:24] stuff bits [23:16] address [15:8] stuff bits [7:0] byte count	R1b	RW_MULTIPLE_REGISTER	These registers are used to control the behavior of the device and to retrieve status information regarding the operation of the device. All Status and Control registers are WORD (32-bit) in size and are WORD aligned. CMD60 is used to read and write these registers.
CMD61	adtc	[31] WR [30:16] stuff bits [15:0] data unit count	R1b	RW_MULTIPLE_BLOCK	The host issues a RW_MULTIPLE_BLOCK (CMD61) to begin the data transfer.
CMD62-63	Reserved				
ACMD6[4]	ac	[31:2] stuff bits [1:0] bus width	R1	SET_BUS_WIDTH	Defines the data bus width ('00'=1bit or '10'=4bit bus) to be used for data transfer. The allowed data bus widths are given in SCR register.

Table continues on the next page...

Table 26-2. Commands for MMC/SD/SDIO cards (continued)

CMD INDEX	Type	Argument	Response type	Abbreviation	Description
ACMD13[4]	adtc	[31:0] stuff bits	R1	SD_STATUS	Send the SD Memory Card status.
ACMD22[4]	adtc	[31:0] stuff bits	R1	SEND_NUM_WR_SECTORS	Send the number of the written sectors (without errors). Responds with 32-bit plus the CRC data block.
ACMD23[4]	ac	[31:23] stuff bits [22:0] Number of blocks	R1	SET_WR_BLK_ERASE_COUNT	Set the number of write blocks to be pre-erased before writing (to be used for fast Multiple Block WR command). "1"=default(one write block).
ACMD41[4]	bcr	[31:0] OCR	R3	SD_APP_OP_COND	Asks the accessed card to send its operating condition register (OCR) contents in the response on the CMD line.
ACMD42[4]	ac	[31:1] stuff bits [0] set_cd	R1	SET_CLR_CARD_DETECT	Connect(1)/Disconnect(0) the 50KOhm pull-up resistor on DATA3 of the card.
ACMD51[4]	adtc	[31:0] stuff bits	R1	SEND_SCR	Reads the SD Configuration Register (SCR).

1. CMD3 differs for MMC and SD cards. For MMC cards, it is referred to as SET_RELATIVE_ADDR, with a response type of R1. For SD cards, it is referred to as SEND_RELATIVE_ADDR, with a response type of R6 (with RCA inside).
2. CMD6 differs completely between high speed MMC cards and high-speed SD cards. Command SWITCH_FUNC is for high-speed SD cards.
3. Command SWITCH is for high speed MMC cards. The Index field can contain any value from 0-255, but only values 0-191 are valid. If the Index value is in the 192-255 range the card does not perform any modification and the SWITCH_ERROR status bit in the EXT_CSD register is set. The Access Bits are shown in Table 2.
4. ACMDs is preceded with the APP_CMD command. Commands listed are used for SD only, other SD commands not listed are not supported on this module.

The access bits for the EXT_CSD access modes are listed in the following table.

Table 26-3. EXT_CSD access modes

Bits	Access name	Operation
00	Command set	The command set is changed according to the Cmd Set field of the argument.
01	Set bits	The bits in the pointed byte are set, according to the bits set to 1 in the Value field.
10	Clear bits	The bits in the pointed byte are cleared, according to the bits set to 1 in the Value field.
11	Write byte	The Value field is written into the pointed byte.

26.3.9 Clocks

The table found here describes the clock sources for uSDHC. For more information on clocking, see the Clocking chapter.

Table 26-4. Clocks

Clock name	Description
hclk	Bus clock
ipg_clk	Peripheral clock
ipg_clk_s	Peripheral access clock for register access
ipg_clk_perclk	Base clock

26.3.10 Clock and reset manager

This module controls all four kinds reset signals within uSDHC:

- Hardware reset
- Software reset for all logic
- Software reset for the data logic
- Software reset for the command logic

All these signals are fed into this module and stable signals are generated inside the module to reset all other modules. The module also gates off all the inside signals.

26.3.11 Clock generator

The clock generator generates the card CLK by peripheral source clock in two stages. The clock divisor can be configured through register SYS_CTRL [SDCLKFS] for prescaler configuration while the [DVS] for divisor configuration. Details can be found in the register function description. See the following figure for the structure of the divider. The term "Base" represents the frequency of peripheral source clock.

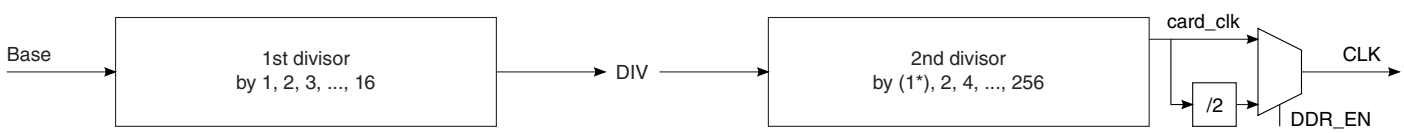


Figure 26-16. Two stages of the clock divider

The first stage outputs an intermediate clock (DIV) that can be Base, Base/2, Base/3, ..., or Base/16.

The second stage is a prescaler and outputs the actual internal working clock (card_clk). This clock is the driving clock for all the sub modules of the SD protocol unit, and helps in syncing FIFOs (see Figure 26-3) to synchronize with the data rate from the internal data buffer. The frequency of the clock output from this stage can be DIV, DIV/2, DIV/

4,..., or DIV/256. Therefore, the highest frequency of the card_clk is base, and the next highest is Base/2, while the lowest frequency is Base/4096. If the duty cycle of the base clock is 50%, the duty cycle of card_clk is also 50%, even when the compound divisor is an odd value.

NOTE

CLK is different for the SDR and DDR modes.

- In the SDR mode, CLK is equal to the internal working clock (card_clk).

- In the DDR mode, CLK is equal to card_clk/2.

26.3.12 SDIO card interrupt

Information on interrupts in 1-bit mode, interrupts in 4-bit mode, and card interrupt handling are detailed in the sections below.

26.3.12.1 Interrupts in 1-bit mode

In this case, the DATA1 pin provides the interrupt function. An interrupt is asserted by pulling the DATA1 low from the SDIO card, until the interrupt service is finished to clear the interrupt.

26.3.12.2 Interrupt in 4-bit mode

As the interrupt and data line 1 share pin 8 in a 4-bit mode, an interrupt is only sent by the card and recognized by the host during a specific time. This is known as the interrupt period. The uSDHC module only provides samples the level on pin 8 during the interrupt period. At all other times, the host treats it as the data signal. The definition of the interrupt period is different for operations with single block and multiple block data transfers.

In the case of normal single data block transmissions, the interrupt period becomes active two clock cycles after the completion of a data packet. This interrupt period lasts until after the card receives the end bit of the next command that has a data block transfer associated with it.

For multiple block data transfers in a 4-bit mode, there is only a limited period of time that the interrupt period can be active because of the limited period of data line availability between the multiple blocks of data. This requires stricter definition of the interrupt period. For this case, the interrupt period is limited to two clock cycles. This

begins two clocks after the end bit of the previous data block. During this 2-clock cycle interrupt period, if an interrupt is pending, the DATA1 line holds low for one clock cycle, then pulls high for the next clock cycle . On completion of the interrupt period, the card releases the DATA1 line into the high Z state. The uSDHC module provides sample of the DATA1 during the interrupt period when the IABG bit in the Protocol Control register is set.

Refer to SDIO Card Specification v1.10 for further information about the SDIO card interrupt.

26.3.12.3 Card interrupt handling

When the CINTIEN bit in the Interrupt Signal Enable Register is set to 0, uSDHC clears the interrupt request to the host system. The host driver should clear this bit before servicing the SDIO interrupt, and should set this bit again after all interrupt requests from the card are cleared to prevent inadvertent interrupts.

The SDIO Card Interrupt Status can be cleared by writing 1 to this bit. But as the interrupt source from the SDIO card does not clear, this bit is set again. To clear this bit, it is required to reset the interrupt source from the external card followed by writing 1 to this bit. In a 1-bit mode, uSDHC detects the SDIO interrupt with or without the SD clock (to support wakeup). In a 4-bit mode, the interrupt signal is sampled during the interrupt period, so there are some sample delays between the interrupt signal from the SDIO card and the interrupt to the Host System Interrupt Controller. When the SDIO status is set and the host driver needs to service this interrupt, the SDIO bit in the Interrupt Control Register of SDIO card is cleared. This is required to clear the SDIO interrupt status latched in uSDHC and to stop driving the interrupt signal to the System Interrupt Controller. The host driver must issue a CMD52 to clear the card interrupt. After completion of the card interrupt service, the SDIO Interrupt Status Enable bit is set to 1 and uSDHC starts sampling the interrupt signal again.

See the figure below for an illustration of the SDIO card interrupt scheme and for the sequences of software and hardware events that take place during a card interrupt handling procedure.

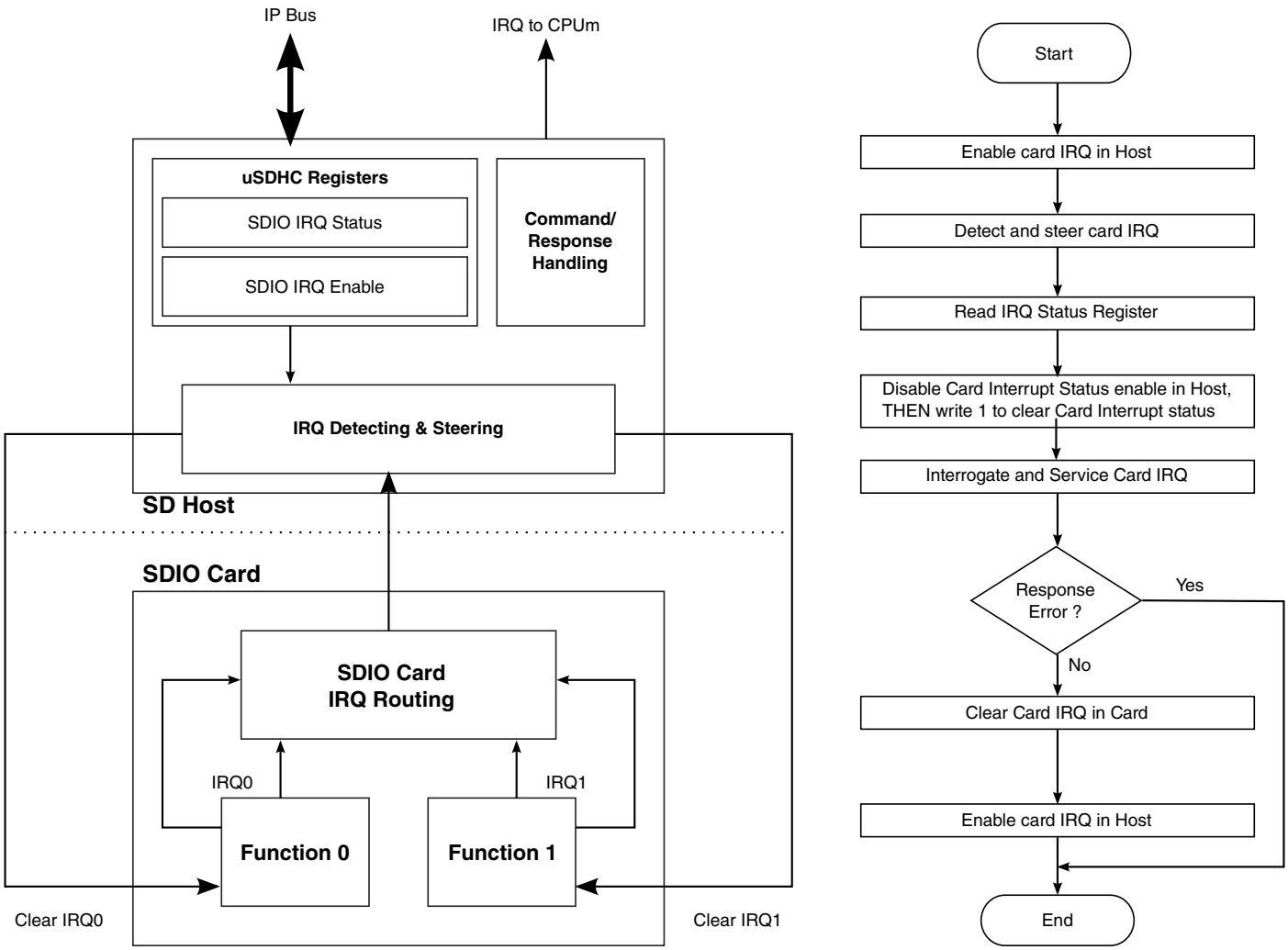


Figure 26-17. Card interrupt scheme, card interrupt detection, and handling procedure

26.3.13 Speed mode selection

To access card with card clock above 52 MHz, you must switch the timing mode to one of supported higher speed timing modes: “High Speed” for frequencies between 26 MHz and 52 MHz. HS200 mode or HS400 mode for frequencies above 52 MHz up to 200 MHz. Note that while the actual timing change is done, the behavior of any command given (such as CMD13) cannot be guaranteed due to the asynchronous operation. Therefore, you must not use CMD13 to check the busy completion of the timing change indication. In case, CMD13 is used, the host must ignore CRC errors, if appear. And, the max frequency for HS200 mode and HS400 mode could be slower than 200 MHz based on the chip implementation, you must confirm with SoC before setting the target frequency for these two speed modes. Following sections describe the mode selections for HS200 mode and HS400 mode in detail.

To operate a chip with a clock above 52MHz, we must switch the timing mode to one of supported higher speed timing modes: “High Speed” for frequencies between 26 MHz and 52 MHz. HS200 for frequencies above 52 MHz up to 200 MHz. Note that while the actual timing change is done, the behavior of any command given (such as CMD13) cannot be guaranteed due to the asynchronous operation. Therefore, you must not use CMD13 to check the busy completion of the timing change indication. In case, CMD13 is used, the host must ignore the CRC errors, if appear. And, the maximum frequency for HS200 mode could be slower than 200 MHz based on the chip implementation. You must confirm with SoC before setting the target frequency. Following sections describe the mode selections for HS200 mode in detail.

26.3.13.1 HS200 mode selection

To switch to HS200, the Driver must perform the following steps:

1. Select the chip (through sending CMD7) and make sure it is unlocked (through CMD42).
2. Read the DEVICE_TYPE [196] field of the Extended CSD register to validate if the chip supports HS200 at the IO voltage appropriate for both host and chip (through CMD8).
3. Read the DRIVER_STRENGTH [197] field of the Extended CSD register to find the supported chip Driver Strengths (through CMD8). The default setting is 50 Ohm. The Host Designer might simulate its specific system, using a device driver models. Host can select the optimal Driver Type that might drive the host system load at the required operating frequency with the minimal noise generated.

   NOTE

   This step can be skipped if changes of Driver strength is not needed.

4. Check the setting of PROT_CTRL[DTW], make sure the data transfer width is set as expected.

   Make sure the I/O pad voltage is 1.8V and corresponding drive strength is set to proper value.

   Set HS200 bit and Driver Strength value in the HS_TIMING [185] field of the Extended CSD register by issuing CMD6. If the host attempts to write an invalid value, the HS_TIMING byte is not changed, the HS200 interface timing is not enabled, the Driver Strength is not changed, and the SWITCH_ERROR field is set. After the chip responds with R1, it might assert Busy signal. Once the busy signal gets de-asserted, the host may send a SEND_STATUS Command (CMD13) using

the HS200 timing and after it receives a Trans State indication and No Error it means that the chip is set to HS200 timing and the Driver Strength is set to the selected settings.

5. At this point, the host can set the frequency to ≤ 200 MHz.
6. The host might invoke the HS200 standard tuning procedure, by sending CMD21 to the chip.

NOTE

The host must switch to the required bus width before starting the tuning operation to allow the tuning sequence to be done using the proper bus operating conditions.

26.3.13.2 Standard Tuning Procedure

By default, lower frequency operation, a fixed sampling clock is used to receive signals on CMD and DAT[3:0]. Before using the HS200, HS400, SDR104, or SDR50 modes, the Host Driver executes the tuning procedure at the mode switch sequence.

1. Issue uSDHC SW reset, set SYS_CTRL[RSTT] to 1.
2. Set VEND_SPEC[FRC_SDCLK_ON] to 1.
3. Set TUNING_CTRL[DIS_CMD_CHK_FOR_STD_TUNING] to 1.
4. Start the tuning procedure by setting TUNING_CTRL[STD_TUNING_EN] and MIX_CTRL[EXE_TUNE] to 1.
5. Issue CMD19(SD)/ CMD21(eMMC) with the proper Command Transfer Type (CMD_XFR_TYP) and Mixer Control (MIX_CTRL) settings.
6. Wait for uSDHC BRR (Buffer Read Ready) interrupt signal is 1.
7. Check MIX_CTRL[EXE_TUNE]. If MIX_CTRL[EXE_TUNE] = 1, repeat 5~6. If MIX_CTRL[EXE_TUNE] = 0, standard tuning has completed, or the tuning has not completed within 40 attempts. The Host Driver might abort this loop if the number of loops exceeds 40 or 150ms timeout occurs. In this case, a fixed sampling clock should be used, (AUTOCMD12_ERR_STATUS[SMP_CLK_SEL] = 0).
8. Sampling Clock Select, AUTOCMD12_ERR_STATUS[SMP_CLK_SEL] , is valid after MIX_CTRL[EXE_TUNE] has changed from 1 to 0. AUTOCMD12_ERR_STATUS[SMP_CLK_SEL] = 1, indicates tuning procedure passed. AUTOCMD12_ERR_STATUS[SMP_CLK_SEL] = 0, indicates tuning procedure failed. The tuning result is applied to the delay chain, CLK Tuning Control and Status (CLK_TUNE_CTRL_STATUS) [30:16], upon successful tuning procedure completion.
9. Clear VEND_SPEC[FRC_SDCLK_ON].
10. Set MIX_CTRL[AUTO_TUNE_EN] to 1.

While the tuning sequence is being performed, the Host Controller does not generate interrupts (including Command Complete), except Buffer Read Ready. CMD19 response errors are not indicated.

Writing AUTOCMD12_ERR_STATUS[SMP_CLK_SEL] to 0 forces the Host Controller to use a fixed sampling clock and resets the tuning circuit of the Host Controller.

NOTE

- There could be slight difference on delay cell delay in each project, and this difference can lead to different loop number needed. TUNING_CTRL[TUNING_STEP] can be used to control how may taps are to be added for each step.
- Manual tuning might be required in cases where standard tuning is not able to close passing window with CMD19/21 transfer fail.

26.3.13.3 Manual tuning procedure

Manual tuning controls more detail in the tuning procedure, and some corner cases. Manual tuning provides control to tuning start point, tuning steps, and delay chains. To start manual tuning:

1. Set TUNING_CTRL[STD_TUNING_EN] to 0.
2. Set SYS_CTRL[RSTA] to reset all.
3. Set MIX_CTRL[SMP_CLK_SEL] and MIX_CTRL[EXE_TUNE] to start tuning.
4. Set VEND_SPEC[FRC_SDCLK_ON] to 1.
5. Config delay chain, CLK_TUNE_CTRL_STATUS[DLY_CELL_SET_PRE] with the valueA. The valueA must start from 0.
6. Send CMD19/21 to card.
7. Poll for CC,CIE,CEBE,CCE,CTOE of Interrupt Status (INT_STATUS) assertion, then go to the step 5.
8. Poll for TC, DEBE, DCE and DTOE of Interrupt Status (INT_STATUS) assertion.
9. Check TP of Interrupt Status (INT_STATUS); TP = 1 & no dat/cmd error indicates current CMD19/21 transfer is passed; otherwise, indicates failed.
10. If TP = 0 or command/data error is observed, increase the valueA with the required tuning step(1~127) and repeat step 5~9.
11. If TP = 1 & no dat/cmd error was met after a loop with TP=0 (or cmd/data error), save the current valueA as value_start. Increase valueA with the required tuning step, and repeat step 5~9 until TP = 0 (or cmd/data error) again. Save the current valueA as value_end.
12. Set MIX_CTRL[EXE_TUNE] to 0.

13. Set VEND_SPEC[FRC_SDCLK_ON] to 0.
14. Set SYS_CTRL[RSTA] to reset all.
15. Set MIX_CTRL[SMP_CLK_SEL] to 1.
16. Set bit[14:0] of CLK Tuning Control and Status (CLK_TUNE_CTRL_STATUS) with the output of expression {(((value_start+value_end)/2)< < 8)& 0xffffff00 -0x300 } | 0x33.
17. Poll bit[30:16] of CLK Tuning Control and Status (CLK_TUNE_CTRL_STATUS) until the value written in the step 16 is read.
18. Set MIX_CTRL[AUTO_TUNE_EN] to 1.

The manual tuning procedure defines a passing window, inside which the tuned clock can capture the correct data, and delay chain is to be configured as centered within the passing window.

It is not a fixed flow as the standard tuning. Any step can be changed based on the real case. In cases where passing window is not closed with a tuning fail, the center of the passing region must be applied for the delay chain setting. In cases where two passing windows separated by tuning fail(s) is observed, the center of the first passing window must be applied for the delay chain setting.

26.4 External signals

The following table describes the uSDHC external signals:

Table 26-5. uSDHC external signals

Signal	Description	Direction
CLK	Clock for MMC/SD/SDIO card	O
CMD	CMD line connect to card	I/O
DATA7	DAT7 line in the 8-bit mode — Not used in other modes	I/O
DATA6	DAT6 line in the 8-bit mode — Not used in other modes	I/O
DATA5	DAT5 line in the 8-bit mode — Not used in other modes	I/O
DATA4	DAT4 line in the 8-bit mode — Not used in other modes	I/O
DATA3	DAT3 line in the 4/8-bit mode or configured as card detection pin. The bit may be configured as card detection pin in the 1-bit mode.	I/O
DATA2	DAT2 line or Read Wait in the 4-bit mode	I/O

Table continues on the next page...

Table 26-5. uSDHC external signals (continued)

Signal	Description	Direction
	Read Wait in 1-bit mode	
DATA1	DAT1 line in the 4/8-bit mode Also, used to detect interrupt in 1/4-bit mode	I/O
DATA0	DAT0 line in all the modes Also, used to detect busy state	I/O
RESET_B	Card hardware reset signal, active low	O
VSELECT	IO power voltage selection signal	O

26.4.1 Signals overview

uSDHC has 14 associated I/O signals.

- The CLK is an internally generated clock used to drive the MMC, SD, and SDIO cards.
- The CMD I/O is used to send commands and receive responses to and from the card. Eight data lines (DAT7~DAT0) are used to perform data transfers between the uSDHC module and the card.
- RST is an output signal used to reset the MMC card.
- VSELECT is an output signal used to change the voltage of the external power supplier. It is optional for system implementation.

If uSDHC needs to support a 4-bit data transfer, DAT7~DAT4 can also be optional and tied to high.

26.5 Application of uSDHC

All communication between the system and cards are controlled by the host. The host sends commands of two types: broadcast and addressed (point-to-point).

Broadcast commands are intended for all cards, such as GO_IDLE_STATE, SEND_OP_COND, and ALL_SEND_CID. In the Broadcast mode, all cards are in the open-drain mode to avoid bus contention. Refer to Commands for MMC/SD/SDIO for the commands of bc and bcr categories.

After the broadcast command CMD3 is issued, the cards enter the standby mode. Addressed type commands are used from this point. In this mode, the CMD/DATA I/O pads turns to the push-pull mode to have the driving capability for maximum frequency operation. Refer to Commands for MMC/SD/SDIO for the commands of ac and adtc categories.

A uSDHC FIFO control problem exists with a software reset for a single block read. See MIX_CTRL[DTDSEL] for more information.

26.5.1 Command send & response receive basic operation

Assuming that the data type WORD is an unsigned 32-bit integer, the flow indicated below presents a guideline for sending a command to the card(s):

```
send_command(cmd_index, cmd_arg, other requirements)
{
WORD wCmd; // 32-bit integer to make up the data to write into Transfer Type register, it is
recommended to implement in a bit-field manner
wCmd = (<cmd_index> & 0x3f) << 24; // set the first 8 bits as '00'+<cmd_index>
set CMDTYP, DPSEL, CICEN, CCCEN, RSTTYP, DTDSEL accorind to the command index;
if (internal DMA is used) wCmd |= 0x1;
if (multi-block transfer) {
          set MSBSEL bit;
          if (finite block number) {
                    set BCEN bit;
                    if (auto12 command is to use) set AC12EN bit;
          }
}
write_reg(CMDARG, <cmd_arg>); // configure the command argument
write_reg(XFERTYP, wCmd); // set Transfer Type register as wCmd value to issue the command
}
wait_for_response(cmd_index)
{
while (CC bit in IRQ Status register is not set); // wait until Command Complete bit is set
read IRQ Status register and check if any error bits about Command are set
if (any error bits are set) report error;
write 1 to clear CC bit and all Command Error bits;
```

```
}
```

For the sake of simplicity, the function wait_for_response is implemented here by means of polling. For an effective and formal way, the response is usually checked after the Command Complete Interrupt is received. When doing this, make sure that the corresponding interrupt status bits are enabled.

For some scenarios, the response time-out is expected. For instance, after all cards respond to CMD3 and move to the Standby state no response to the Host when CMD2 is sent. The host driver deals with "fake" errors like this with caution.

26.5.2 Card identification mode

When a card is inserted to the socket or the card is reset by the host, the host needs to validate the operation voltage range, identify the cards, request the cards to publish the Relative Card Address (RCA) or to set the RCA for MMC cards.

26.5.2.1 Card detect

See the figure below for a flow diagram showing the detection of MMC, SD, and SDIO cards using uSDHC.

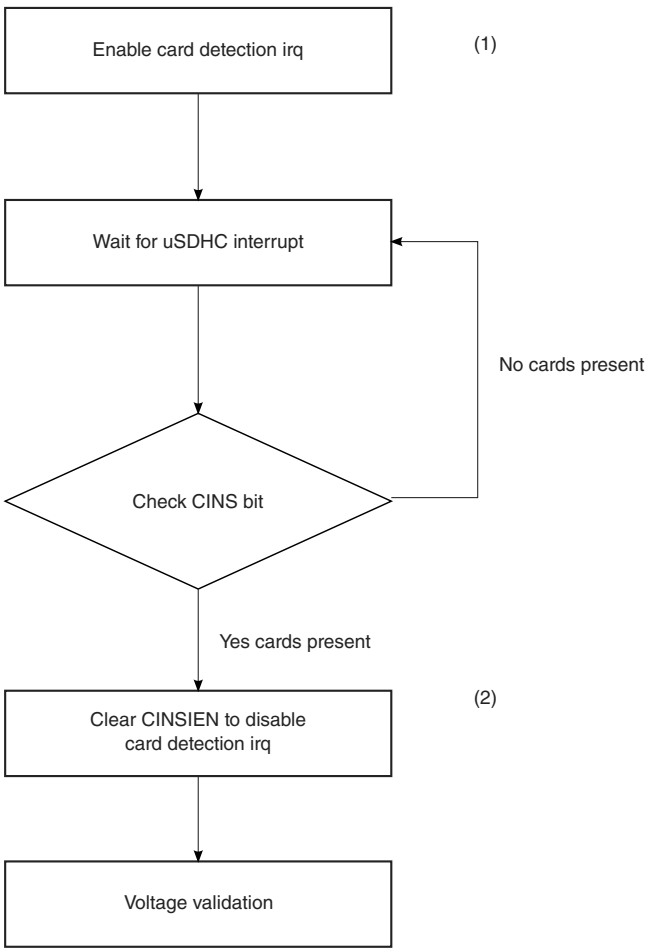


Figure 26-18. Flow diagram for card detection

Here is the card detect sequence:

- Set the CINSIEN bit to enable card detection interrupt.
- When an interrupt from uSDHC is received, check the CINS bit in the Interrupt Status register to see if it was caused by card insertion.
- Clear the CINSIEN bit to disable the card detection interrupt and ignore all card insertion interrupts afterwards.

26.5.2.2 Reset

The host consists of three types of resets:

- Hardware reset (Card and Host) that is driven by Power On Reset (POR).

- Software reset (Host only) is initiated by the write operation on the RSTD, RSTC, or RSTA bits of the System Control register to reset the data part, command part, or all parts of the host controller, respectively.
- Card reset (Card only): The command, "Go_Idle_State" (CMD0), is the software reset command for all types of MMC cards and SD memory cards. This command sets each card into the idle state regardless of the current card state. For an SDIO card, CMD52 is used to write an I/O reset in CCCR. The cards are initialized with a default relative card address (RCA=0x0000) and with a default driver stage register setting (lowest speed, highest driving current capability).

After the card is reset, the host needs to validate the voltage range of the card. See the figure below for the software flow to reset both uSDHC and the card.

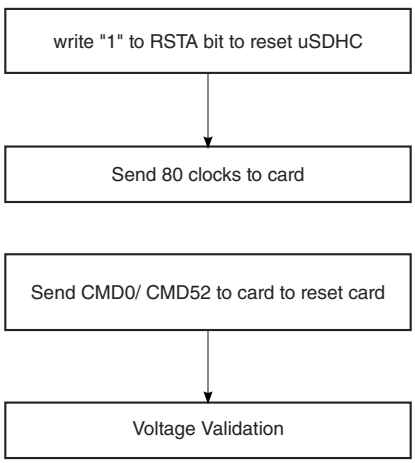


Figure 26-19. Flow chart for resetting uSDHC and SD I/O card

```
software_reset() { set_bit(SYSCTRL, RSTA); // software reset the Host set DTOCV and
      SDCLKFS bit fields to get the CLK of frequency around 400kHz configure IO pad to set
the power
      voltage of external card to around 3.0V poll bits CIHB and CDIHB bits of PRSSTAT to
wait both
      bits are cleared set_bit(SYSCTRL, INTIA); // send 80 clock ticks for card to power up
      send_command(CMD_GO_IDLE_STATE, <other parameters>); // reset the card with CMD0 or
      send_command(CMD_IO_RW_DIRECT, <other parameters>); }
```

26.5.2.3 Voltage validation

All cards should be able to establish communication with the host using any operation voltage in the maximum allowed voltage range specified in the card specification. However, the supported minimum and maximum values for VDD are defined in the Operation Conditions Register (OCR) and may not cover the whole range.

Cards that store the CID and CSD data in the preload memory are only able to communicate this information under data transfer VDD conditions. This means that if the host and card have non-common VDD ranges, the card is neither able to complete the identification cycle nor able to send CSD data.

Therefore, special commands Send_Op_Cont (CMD1 for MMC), SD_Send_Op_Cont (ACMD41 for SD Memory), and IO_Send_Op_Cont (CMD5 for SD I/O) are used. The voltage validation procedure is designed to provide a mechanism to identify and reject cards that do not match the VDD range(s) desired by the host. This is accomplished when the host sends the desired VDD voltage window as the operand of this command. Cards that cannot perform the data transfer in the specified range must discard themselves from further bus operations and go into the Inactive state. By omitting the voltage range in the command, the host can query each card and determine the common voltage range before sending out-of-range cards into the Inactive state. This query should be used if the host is able to select a common voltage range or if a notification is sent to the system when a non-usable card in the stack is detected.

The following steps show how to perform voltage validation when a card is inserted:

```
voltage_validation(voltage_range_arguement) { label the card as UNKNOWN;
      send_command(IO_SEND_OP_COND, 0x0, <other parameters are omitted>); // CMD5, check SDIO
      operation voltage, command argument is zero if (RESP_TIMEOUT !=
      wait_for_response(IO_SEND_OP_COND)) { // SDIO command is accepted if (0 < number of IO
      functions) { label the card as SDIO; IORDY = 0; while (!(IORDY in IO OCR response))
{ // set
      voltage range for each IO function send_command(IO_SEND_OP_COND, <voltage range>,
      <other parameter>); wait_for_response(IO_SEND_OP_COND); } // end of while ... } // end
      of if (0 < ... if (memory part is present inside SDIO card) Label the card as
SDCombo; //
      this is an SD-Combo card } // end of if (RESP_TIMEOUT ... if (the card is labelled as
SDIO
      card) return; // card type is identified and voltage range is set, so exit the
function;
      send_command(APP_CMD, 0x0, <other parameters are omitted>); // CMD55, Application
      specific CMD prefix if (no error calling wait_for_response(APP_CMD, <...>) { // CMD55
is
      accepted send_command(SD_APP_OP_COND, <voltage range>, <...>); // ACMD41, to set
      voltage range for memory part or SD card wait_for_response(SD_APP_OP_COND); // voltage
range
      is set if (card type is UNKNOWN) label the card as SD; return; // } // end of if (no
error ...
      else if (errors other than time-out occur) { // command/response pair is corrupted
deal with
      it by program specific manner; } // of else if (response time-out else { // CMD55 is
refuse,
      it must be MMC card if (card is already labelled as SDCombo) { // change label re-
label the
      card as SDIO; ignore the error or report it; return; // card is identified as SDIO
card } //
      of if (card is ... send_command(SEND_OP_COND, <voltage range>, <...>); if
      (RESP_TIMEOUT == wait_for_response(SEND_OP_COND)) { // CMD1 is not accepted, either
label the
      card as UNKNOWN; return; } // of if (RESP_TIMEOUT ... } // of else }
```

26.5.2.4 Card registry

This section briefly describes the registry flow. For details, please refer to the card specifications. Card registry for the MMC and SD/SDIO/SD combo cards are different. For the SD card, the identification process starts at a clock rate lower than 400 kHz and the power voltage higher than 2.7 V (as defined by the card spec). Currently, the CMD line output drives are push-pull drivers instead of open-drain. After the bus is activated, the host requests the card to send their valid operation conditions.

The response to ACMD41 is the operation condition register of the card. The same command is sent to all the new cards in the system. Incompatible cards are put into the Inactive state. The host then issues the command, All_Send_CID (CMD2), to each card to get its unique card identification (CID) number. Cards that are currently unidentified (in the Ready state), send their CID number as the response. After the CID is sent by the card, the card goes into the Identification state.

The host then issues Send_Relative_Addr (CMD3), requesting the card to publish a new relative card address (RCA) that is shorter than the CID. This RCA is used to address the card for future data transfer operations. After the RCA is received, the card changes its state to the Standby state. At this point, if the host wants the card to have an alternative RCA number, it may ask the card to publish a new number by sending another Send_Relative_Addr command to the card. The last published RCA is the actual RCA of the card.

The host repeats the identification process with CMD2 and CMD3 for each card in the system until the last CMD2 gets no response from any of the cards in the system.

For MMC operation, the host starts the card identification process in the open-drain mode with the identification clock rate lower than 400 kHz and the power voltage higher than 2.7 V. The open drain driver stages on the CMD line to allow parallel card operation during card identification. After the bus is activated, the host requests the cards to send their valid operation conditions (CMD1). The response to CMD1 is the "wired AND " operation on the condition restrictions of all cards in the system. Incompatible cards are sent into the Inactive state. The host then issues the broadcast command All_Send_CID (CMD2), asking all cards for their unique card identification (CID) number. All unidentified cards (the cards in the Ready state) simultaneously start sending their CID numbers serially, while bit-wise monitoring their outgoing bit stream. Those cards, whose outgoing CID bits do not match the corresponding bits on the command line in any one of the bit periods, stop sending their CID immediately and must wait for the next identification cycle. As the CID is unique for each card, only one card can be successfully sent its full CID to the host. This card then goes into the Identification state. Thereafter, the host issues Set_Relative_Addr (CMD3) to assign a relative card address (RCA) to the card. After the RCA is received, the state of the card changes to standby, and the card does not react in further identification cycles. Also, its output driver switches

from open-drain to push-pull. The host repeats the process, mainly CMD2 and CMD3, until the host receives a time-out condition to recognize the completion of the identification process.

The following steps show how to perform an operation using MMC cards:

```
card_registry() { do { // decide RCA for each card until response time-out
            if(card is labelled as SDCombo or SDIO) { // for SDIO card like device
            send_command(SET_RELATIVE_ADDR, 0x00, <...>); // ask SDIO card to publish its RCA
            retrieve RCA from response; } // end if (card is labelled as SDCombo ... else if
(card
            is labelled as SD) { // for SD card send_command(ALL_SEND_CID, <...>); if
            (RESP_TIMEOUT == wait_for_response(ALL_SEND_CID)) break;
send_command(SET_RELATIVE_ADDR,
            <...>); retrieve RCA from response; } // else if (card is labelled as SD ... else
            if (card is labelled as MMC) { send_command(ALL_SEND_CID, <...>); rca = 0x1; //
            arbitrarily set RCA, 1 here for example send_command(SET_RELATIVE_ADDR, 0x1 <<
16,
            <...>); // send RCA at upper 16 bits } // end of else if (card is labelled as MMC
            ... } while (response is not time-out); }
```

26.5.3 Card access

Information about Block Write, Block Read, Suspense Resume, ADMA Usage, Transfer Error, and Card Interrupt are detailed in the sections below.

26.5.3.1 Block write

Information on Normal Write, DDR Write, and Write with Pause are detailed in the sections below.

26.5.3.1.1 Normal Write

During a block write (CMD24 - 27, CMD60, CMD61), one or more blocks of data are transferred from the host to the card with a CRC appended to the end of each block by the host. If the CRC fails, the card indicates that the failure on the DATA line. The transferred data is discarded and not written, and all further transmitted blocks (in multiple block write mode) are ignored.

If the host uses partial blocks whose accumulated length is not block aligned and block misalignment is not allowed (CSD parameter WRITE_BLK_MISALIGN is not set), the card detects the block misalignment error and aborts the programming before the beginning of the first misaligned block. The card sets the ADDRESS_ERROR error bit in the status register, and while ignoring all further data transfer, waits in the Receive-data-State for a stop command. The write operation is also aborted if the host tries to write over a write-protected area.

For MMC and SD cards, programming of the CID and CSD registers does not require a previous block length setting. The transferred data is also CRC protected. If a part of the CSD or CID register is stored in ROM, then this unchangeable part must match the corresponding part of the receive buffer. If this match fails, then the card reports an error and does not change any register contents.

For all types of cards, some may require long and unpredictable periods of time to write a block of data. After receiving a block of data and completing the CRC check, the card begins writing and holds the DATA line low if its write buffer is full and unable to accept new data from a new WRITE_BLOCK command. The host may poll the status of the card with a SEND_STATUS command (CMD13) or other means for SDIO cards at any time, and the card responds with its status. The responded status indicates whether the card can accept new data or whether the write process is still in progress. The host may deselect the card by issuing a CMD7 (to select a different card) to place the card into the Standby state and release the DATA line without interrupting the write operation. When re-selecting the card, it reactivates the busy indication by pulling data to low if the programming is still in progress and the write buffer is unavailable.

The software flow to write to a card that incorporates the internal DMA and the write operation is a multi-block write with the Auto CMD12 enabled. For the other methods ((CPU polling status) with different transfer methods, the internal DMA parts should be removed and the alternative steps should be straightforward.

The software flow to write to a card is described below:

1. Check the card status, wait until the card is ready for data.
2. Set the card block length/size:
   - For SD/MMC cards, use SET_BLOCKLEN (CMD16).
   - For SDIO cards or the I/O portion of SDCombo cards, use IO_RW_DIRECT (CMD52) to set the I/O Block Size bit field in the CCCR register (for function 0) or FBR register (for functions 1~7).
3. Set the uSDHC block length register to be the same as the block length set for the card in step 2.
4. Set the uSDHC number block register (NOB), where nob is 5, for instance.
5. Disable the buffer write ready interrupt, configure the DMA settings and enable the uSDHC DMA when sending the command with data transfer. The AC12EN bit should also be set.
6. Wait for the Transfer Complete interrupt.
7. Check the status bit to see if a write CRC error occurred, or another error that occurred during the auto12 command sending and response receiving.

26.5.3.1.2 DDR write

uSDHC supports the dual data rate mode.

The software flow to write to a card in the DDR mode is described as below:

1. Check the card status and wait until the card is ready for data.

NOTE

For eMMC card, block length only can be set to 512byte.

2. Set the uSDHC number block register (NOB), where nob is 5, for instance.
3. Set the eMMC card to high-speed mode and use SWITCH(CMD6).
4. Set the eMMC card bus with 4-bit/8-bit DDR mode and use SWITCH(CMD6).
5. Disable the buffer write ready interrupt, configure the DMA settings and enable the uSDHC DMA when sending the command with data transfer. The DDR_EN and AC12EN bits should be set.
6. Wait for the Transfer Complete interrupt.
7. Check the status bit to see if a write CRC error occurred or another error that occurred during the auto12 command sending and response receiving.

26.5.3.1.3 Write with Pause

The write operation can be paused during the transfer. Instead of stopping the CLK at any time to pause all the operations, which is also inaccessible to the host driver, the driver can set the Stop At Block Gap Request (SABGREQ) bit in the Protocol Control register to pause the transfer between the data blocks. As there is no time-out condition in a write operation during the data blocks, a write to all types of cards can be paused in this way, and if the DATA0 line is not required to de-assert to release the busy state, no suspend command is needed.

Similar to the flow described in Normal Write, the write with pause is shown with the same kind of write operation:

1. Check the card status and wait until the card is ready for data.
2. Set the card block length/size:
   - For SD/MMC, use SET_BLOCKLEN (CMD16).
   - For SDIO cards or the I/O portion of SDCombo cards, use IO_RW_DIRECT(CMD52) to set the I/O Block Size bit field in the CCCR register (for function 0) or FBR register (for functions 1~7).
3. Set the uSDHC block length register to be the same as the block length set for the card in step 2.
4. Set the uSDHC number block register (NOB), where nob is 5, for instance.

5. Disable the buffer write ready interrupt, configure the DMA settings and enable the uSDHC DMA when sending the command with data transfer. The AC12EN bit should also be set.
6. Set the SABGREQ bit.
7. Wait for the Transfer Complete interrupt.
8. Clear the SABGREQ bit.
9. Check the status bit to see if a write CRC error occurred.
10. Set the CREQ bit to continue the write operation.
11. Wait for the Transfer Complete interrupt.
12. Check the status bit to see if a write CRC error occurred or some another error that occurred during the auto12 command sending and response receiving.

The number of blocks left during the data transfer is accessible by reading the contents of the BLKCNT field in the Block Attribute register. As the data transfer and the setting of the SABGREQ bit are concurrent, and the delay of register read and the register setting, the actual number of blocks left may not be exactly the value read earlier. The host driver reads the value of BLKCNT after the transfer is paused and the Transfer Complete interrupt is received.

It is also possible that the last block has begun when the Stop At Block Gap Request is sent to the buffer. In this case, the next block gap is the end of the transfer. These types of requests are ignored, the driver should treat these as a non-pause transfer, and deal with it as a common write operation.

When the write operation is paused, the data transfer inside the host system is not stopped, and the transfer is active until the data buffer is full. Because of this, it is recommended to avoid using the Suspend command for the SDIO card. This is because when such a command is sent, uSDHC interprets the system that switches to another function on the SDIO card and flush the data buffer. uSDHC takes the Resume command as a normal command with data transfer, and it is left for the host driver to set all the relevant registers before the transfer is resumed. If there is only one block to send when the transfer is resumed, the MSBSEL and BCEN bits of the Transfer Type register are set as well as the AC12EN bit. However, the uSDHC module automatically sends a CMD12 to mark the end of the multi-block transfer.

26.5.3.2 Block read

Information about Normal read, DDR read, Read with Pause, and Delay Line (DLL) in Read Path are detailed in the sections below.

26.5.3.2.1 Normal read

For block reads, the basic unit of data transfer is a block whose maximum size is stored in areas defined by the corresponding card specification. A CRC is appended to the end of each block, ensuring data transfer integrity. The CMD17, CMD18, CMD53, CMD60, CMD61, and so on, can initiate a block read. After completing the transfer, the card returns to the Transfer state. For multi blocks read, data blocks are continuously transferred until a stop command is issued.

The software flow to read from a card that incorporates the internal DMA and the read operation is a multi-block read with the Auto CMD12 enabled. For the other methods ((CPU polling status) with different transfer methods, the internal DMA parts should be removed and the alternative steps should be straightforward.

The software flow to read from a card is described below:

1. Check the card status and wait until card is ready for data.
2. Set the card block length/size:
   - For SD/MMC, use SET_BLOCKLEN (CMD16).
   - For SDIO cards or the I/O portion of SDCombo cards, use IO_RW_DIRECT(CMD52) to set the I/O Block Size bit field in the CCCR register (for function 0) or FBR register (for functions 1~7).
3. Set the uSDHC block length register to be the same as the block length set for the card in step 2.
4. Set the uSDHC number block register (NOB), where nob is 5, for instance.
5. Disable the buffer read ready interrupt, configure the DMA settings, and enable the uSDHC DMA when sending the command with data transfer. The AC12EN bit should also be set.
6. Wait for the Transfer Complete interrupt.
7. Check the status bit to see if a read CRC error occurred, or another error occurred during the auto12 command sending and response receiving.

26.5.3.2.2 DDR read

The uSDHC module supports dual data rate mode.

The software flow to write to a card in the DDR mode is described below:

1. Check the card status and wait until the card is ready for data.

NOTE

For eMMC card, block length can be set to only 512 bytes.

2. Set the uSDHC number block register (NOB) where nob is 5, for instance.
3. Set the eMMC card to high-speed mode and use SWITCH (CMD6).

4. Set the eMMC card bus with 4-bit /8-bit DDR mode and use SWITCH(CMD6).
5. Disable the buffer write ready interrupt, configure the DMA settings, and enable the uSDHC DMA when sending the command with data transfer. The DDR_EN and AC12EN bits should be set.
6. Wait for the Transfer Complete interrupt.
7. Check the status bit to see if a write CRC error occurred, or another error that occurred during the auto12 command sending and response receiving.

26.5.3.2.3 Read with Pause

The read operation is not generally able to pause. Only the SDIO card (and SDCombo card working under I/O mode) supporting the Read Wait feature can pause during the read operation. If the SDIO card supports Read Wait (SRW bit in CCCR register is 1), the host driver can set the SABGREQ bit in the Protocol Control register to pause the transfer between the data blocks.

Before setting the SABGREQ bit, ensure that the RWCTL bit in the Protocol Control register is set, otherwise uSDHC does not assert the Read Wait signal during the block gap and data corruption occurs. It is recommended to set the RWCTL bit after the Read Wait capability of the SDIO card is recognized.

Similar to the flow described in Normal read, the read with pause is shown with the same kind of read operation:

1. Check the SRW bit in the CCR register on the SDIO card to confirm that the card supports the Read Wait mode.
2. Set the RWCTL bit.
3. Check the card status and wait until the card is ready for data.
4. Set the card block length/size:
   - For SD/MMC, use SET_BLOCKLEN (CMD16).
   - For SDIO cards or the I/O portion of SDCombo cards, use IO_RW_DIRECT(CMD52) to set the I/O Block Size bit field in the CCCR register (for function 0) or FBR register (for functions 1~7).
5. Set the uSDHC block length register to be the same as the block length set for the card in step 2.
6. Set the uSDHC number block register (NOB), where nob is 5, for instance.
7. Disable the buffer read ready interrupt, configure the DMA setting, and enable the uSDHC DMA when sending the command with data transfer. The AC12EN bit should also be set.
8. Set the SABGREQ bit.
9. Wait for the Transfer Complete interrupt.
10. Clear the SABGREQ bit.
11. Check the status bit to see if read CRC error occurred.

12. Set the CREQ bit to continue the read operation.
13. Wait for the Transfer Complete interrupt.
14. Check the status bit to see if a read CRC error occurred, or another error occurred during the auto12 command sending and response receiving.

Similar to the Write operation, it is possible to meet the ending block of the transfer when paused. In this case, uSDHC ignores the Stop At Block Gap Request and treats it as a command read operation.

Unlike the write operation, there is no remaining data inside the buffer when the transfer is paused. All data received before the pause is transferred to the host system. No matter if the Suspend command is sent or not, the internal data buffer is not flushed.

If the Suspend command is sent and the transfer is later resumed by means of a Resume command, uSDHC takes the command as a normal one accompanied with data transfer. It is left for the host driver to set all the relevant registers before the transfer is resumed. If there is only one block to send when the transfer is resumed, the MSBSEL and BCEN bits of the Transfer Type register are set, as well as the AC12EN bit. However, the uSDHC automatically sends CMD12 to mark the end of multi-block transfer.

26.5.3.2.4 Delay Line (DLL) in read path

The DLL is newly added to assist in sampling read data. The DLL provides the ability to programmatically select a quantized delay (in fractions of the clock period) regardless of on-chip variations such as process, voltage, and temperature (PVT).

The reasons why DLL is needed for uSDHC are these:

- The path of read data traveling from card to host varies.
- In the SD/MMC DDR mode, the minimum input setup and hold time are both at 2.5 ns.

The data sampling window is so small that the delay of loopback clock needs to be accurate and consistent regardless of PVT. The DLL takes the divided card_clk as the reference clock and loopback clock as the input clock. It then generates a delayed version of the input clock according to the programmed target delay.

The DLL can be disabled or bypassed, and it can also be manually set for a fixed delay in the override mode. The override value set is the number of delay cells. In the override mode, there is no need to set the DLL_enable. Another DLL mode is target value mode. In this mode, the DLL automatically adjusts the number of delay cells according to the target value set by the user and PVT changes. Be aware that the target value is in units of 1/32 of the clock reference period. If the card_clk is 100Mhz, then the reference clock period is 10ns; setting target value of 16 means 5ns =(16/32)*10ns. The software can disable automatic update by the setting dll_gate_update bit.

As you might change the frequency of card_clk from time to time by changing SDCLKFS[7:0]/DVS[3:0], the software must adjust the delay value to ensure it works correctly when the reference clock (card_clk) is changed. If DLL is to be used, make sure SDCLKFS is used (at least be set to divided by 2) so that the REF_CLK generated are the same frequency as card_clk.

Step 1: Set the DLL_CTRL_RESET and DLL_CTRL_ENABLE fields

Step 2: Configure the SDCLKFS[7:0] and DVS[3:0]

Step 3: Wait until SDSTB is asserted

Step 4: Clear the DLL_CTRL_RESET field

Step 5: Wait until both the DLL_STS_SLV_LOCK and DLL_STS_REF_LOCK are asserted

Step 6: Set the DLL_CTRL_SLV_FORCE_UPD

Step 7: Clear the DLL_CTRL_SLV_FORCE_UPD

NOTE

The software should make sure that the DLL_CTRL_SLV_FORCE_UPD lasts for at least one card_clk. So, the software may need to add some delay between step 6 and step 7.

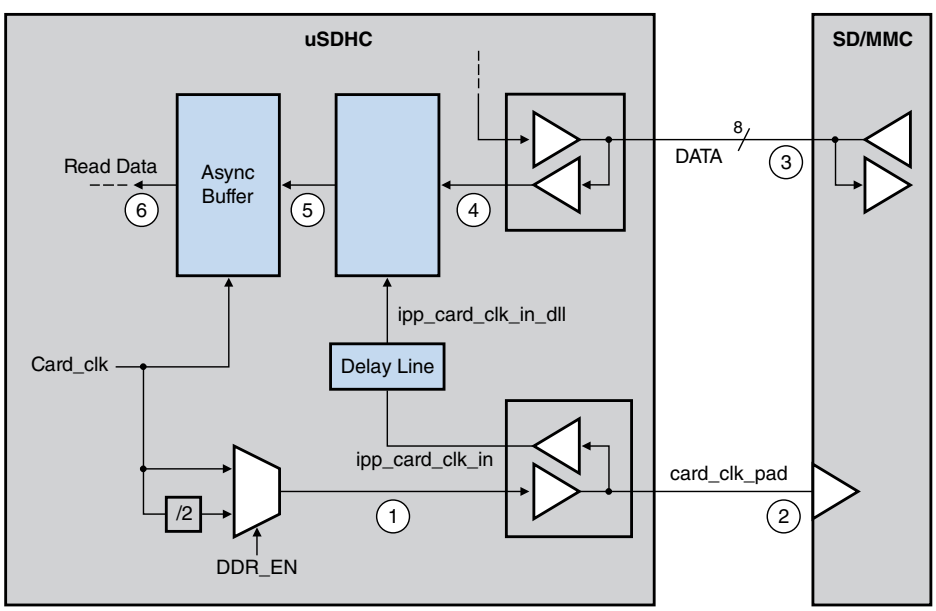


Figure 26-20. DLL in read path

26.5.3.3 Suspend Resume

The uSDHC module supports the Suspend Resume operations of SDIO cards, although slightly different than the suggested implementation of Suspend in the SDIO card specification.

26.5.3.3.1 Suspend

After setting the SABGREQ bit, the host driver may send a Suspend command to switch to another function of the SDIO card. The uSDHC module does not monitor the content of the response, so it does not know if the Suspend command succeeded or not. Accordingly, it does not de-assert Read Wait for read pause. To solve this problem, the host driver does not mark the Suspend command as "Suspend", (that is, setting the CMDTYP bits to 01). Instead, the driver sends this command as if it were a normal command, and only when the command succeeds, and the BS bit is set in the response, can the Driver send another command marked as "Suspend" to inform uSDHC that the current transfer is suspended. Here is the sequence for the Suspend operation:

1. Set the SABREQ bit to pause the current data transfer at block gap.
2. After the BGE bit is set, send the Suspend command to suspend the active function. The CMDTYP bit field must be 2'b00.
3. Check the BS bit of the CCCR in the response. If it is 1, repeat this step until the BS bit is cleared or abandon the suspend operation according to the Driver strategy.
4. Send another normal I/O command to the suspended function. The CMDTYP of this command must be 2'b01, so uSDHC can detect this special setting and be informed that the paused operation has successfully suspended. If the paused transfer is a read operation, uSDHC stops driving DATA2 and goes to the idle state.
5. Save the context registers in the system memory for later use, including the DMA System Address Register (for internal DMA operation), and the Block Attribute Register.
6. Begin operation for another function on the SDIO card.

26.5.3.3.2 Resume

To resume the data transfer, a Resume command is issued:

1. To resume the suspended function, restore the context register with the saved value in step #5 of the Suspend operation.
2. Send the Resume command: In the Transfer Type register, all the bit fields are set to the value as if this were another ordinary data transfer instead of a transfer resume (except the CMDTYP is set to 2'b10).
3. If the Resume command has responded, the data transfer is resumed.

26.5.3.4 ADMA usage

To use the ADMA in a data transfer, the host driver must prepare the correct descriptor chain prior to sending the read/write command.

The steps to prepare the correct descriptor chain are these:

1. Create a descriptor to set the data length that the current descriptor group is about to transfer. The data length should be even numbers of the block size.
2. Create another descriptor to transfer the data from the address setting in this descriptor. The data address must be at a page boundary (4KB address aligned).
3. If necessary, create a Link descriptor containing the address of the next descriptor. The descriptor group is created in steps 1 ~ 3.
4. Repeat steps 1 ~ 3 until all descriptors are created.
5. In the last descriptor, set the End flag to 1 and make sure the total length of all descriptors matches the product of the block size and block number configured in the Block Attribute Register.
6. Set the ADMA System Address Register to the address of the first descriptor and set the DMAS field in the Protocol Control Register to 01 to select the ADMA.
7. Issue a write or read command with the DMAEN bit set to 1 in the Transfer Type Register.

Steps 1 ~ 5 are independent of step 6, so step 6 can finish before steps 1 ~ 5. Regarding the descriptor configuration, it is recommended not to use the Link descriptor as it requires extra system memory access.

26.5.3.5 Transfer error

Information about CRC, Internal DMA, Transfer ADMA, and Auto CMD12 errors are detailed in the sections below.

26.5.3.5.1 CRC error

It is possible at the end of a block transfer that a write CRC status error or read CRC error occurs. For this type of error, the latest block received is discarded. This is because the integrity of the data block is not guaranteed. It is recommended to discard the following data blocks and re-transfer the block from the corrupted one.

For a multi-block transfer, the host driver issues a CMD12 to abort the current process and start the transfer by a new data command. In this scenario, even when the AC12EN and BCEND bits are set, uSDHC does not automatically send a CMD12 because the last

block is not transferred. On the other hand, if it is within the last block that the CRC error occurs, an Auto CMD12 is sent by uSDHC. In this case, the host driver re-sends or re-obtains the last block with a single block transfer.

26.5.3.5.2 Internal DMA error

During the data transfer with internal Simple DMA, if the DMA engine encounters an error on the AHB bus, the DMA operation is aborted, and the DMA error interrupt is sent to the host system. When acknowledged by such an interrupt, the host driver calculates the start address of data block in which the error occurs.

The start address can be calculated by either:

- Reading the DMA System Address register: The error occurs during the previous burst. Considering the block size, the previous burst length and the start address of the next burst transfer, it is straight forward to obtain the start address of the corrupted block.
- Reading the BLKCNT field of the Block Attribute register: Considering the number of blocks left, the total number to transfer, the start address of transfer, and the size of each block, the start address of corrupted block can be determined.To use this method, MIX_CTRL[BCEN] bit has to be set to enable Block Attribute register update.

When a DMA error occurs, it is recommended to abort the current transfer by means of a CMD12 (for multi block transfer), apply a reset for data, and restart the transfer from the corrupted block to recover from the error.

26.5.3.5.3 Transfer ADMA error

There are three kinds of possible ADMA errors: The AHB transfer, invalid descriptor, and data-length mismatch. Whenever these errors occur, the DMA transfer stops and the corresponding error status bit is set.

For acknowledging the status, the host driver should recover the error as shown below and re-transfer from the place of interruption.

- AHB transfer error: Such errors may occur during data transfer or descriptor fetch. For either scenario, it is recommended to retrieve the transfer context, reset for the data part and re-transfer the block that was corrupted, or to the next block if no block is corrupted.
- Invalid descriptor error: For such errors, it is recommended to retrieve the transfer context, reset for the data part and recreate the descriptor chain from the invalid descriptor and issue a new transfer. As the data to transfer now may be less than the

previous setting, the data length configured in the new descriptor chain should match the new value.
- Data-length mismatch error: It is similar to recover from this error. The Host Driver polls relating registers to retrieve the transfer context, apply a reset for the data part, configure a new descriptor chain and make another transfer if there is data left. Like the previous scenario of the invalid descriptor error, the data length must match the new transfer.

26.5.3.5.4 Auto CMD12 error

After the last block of the multi-block transfer is sent or received, and the AC12EN bit is set when the data transfer is initiated by the data command, uSDHC automatically sends a CMD12 to the card to stop the transfer.

When errors with this command occur, it is recommended to the host driver to deal with the situations in the following manner:

- Auto CMD12 response time-out: It is not certain whether the command is accepted by the card or not. The host driver clears the auto CMD12 error status bits and re-send CMD12 until it is accepted by the card.
- Auto CMD12 response CRC error: As the card responds to CMD12, it aborts the transfer. The host driver may ignore the error and clear the error status bit.
- Auto CMD12 conflict error or not sent: The command is not sent; therefore, the host driver sends a CMD12 manually.

26.5.3.6 Card interrupt

The external cards can inform the host controller by means of some special signals. For the SDIO card, it can be the low-level on the DATA1 line during some special period. The uSDHC module only monitors the DATA1 line and supports the SDIO interrupt.

When the SDIO interrupt is captured by uSDHC, and the host system is informed by uSDHC asserting the uSDHC interrupt line, the interrupt service from the host driver is called.

As the interrupt source is controlled by the external card, the interrupt from the SDIO card must be serviced before the CINT bit is cleared. Refer to Card interrupt handling for the card interrupt handling flow.

26.5.4 Switch function

A switch command is issued by the host driver to enable new features added to the SD/MMC spec. The SD/MMC cards can transfer data at bus widths other than 1-bit. Different speed modes are also defined. To enable these features, a switch command is issued by the host driver.

For SDIO cards, the high-speed mode/DDR50/SDR50/SDR104 are enabled by writing the EHS bit in the CCCR register after the SHS bit is confirmed as high. For SD cards, the high-speed mode/DDR50/SDR50/SDR104 are queried and enabled by a CMD6 (with the mnemonic symbol as SWITCH_FUNC). For MMC cards, the high-speed mode/ HS200 are queried by a CMD8 and enabled by a CMD6 (with the mnemonic symbol as SWITCH).

The SDR4-bit, SDR8-bit, DDR4-bit, and DDR8-bit width of MMC is also enabled by the SWITCH command, but with a different argument.

These new functions can also be disabled by a software reset. For SDIO cards, it can be done by setting the RES bit in the CCCR register. For other cards, it can be accomplished by issuing a CMD0. This method of restoring to the normal mode is not recommended because a complete identification process is needed before the card is ready for data transfer.

For the sake of simplicity, the following pseudocode examples do not show current capability check, which is recommended in the function switch process.

26.5.4.1 Query, enable, and disable SDIO high-speed mode

```
enable_sdio_high_speed_mode(void)
{
send CMD52 to query bit SHS at address 0x13;
if (SHS bit is '0') report the SDIO card does not support high speed mode and return;
send CMD52 to set bit EHS at address 0x13 and read after write to confirm EHS bit is set;
change clock divisor value or configure the system clock feeding into uSDHC to generate the
card_clk of around 50MHz;
(data transactions like normal peers)
}
disable_sdio_high_speed_mode(void)
{
send CMD52 to clear bit EHS at address 0x13 and read after write to confirm EHS bit is
cleared;
change clock divisor value or configure the system clock feeding into uSDHC to generate the
card_clk of the desired value below 25MHz;
(data transactions like normal peers)
}
```

26.5.4.2 Query, enable, and disable SD high-speed mode/DDR50/ SDR50/SDR104

```
enable_sd_speed_mode(void)
{
set BLKCNT field to 1 (block), set BLKSIZE field to 64 (bytes);
send CMD6, with argument 0xFFFFFx and read 64 bytes of data accompanying the R1 response;
(high speed mode,x=1; SDR50,x=2; SDR104,x=3; DDR50,x=4;)
wait data transfer done bit is set;
check if the bit x of received 512 bits is set;
if (bit 401 is '0') report the SD card does not support high speed mode and return;
if (bit 402 is '0') report the SD card does not support SDR50 mode and return;
if (bit 403 is '0') report the SD card does not support SDR104 mode and return;
if (bit 404 is '0') report the SD card does not support DDR50 mode and return;
send CMD6, with argument 0x80FFFFFx and read 64 bytes of data accompanying the R1 response;
(high speed mode,x=1; SDR50,x=2; SDR104 x=3; DDR50 x=4;)
check if the bit field 379~376 is 0xF;
if (the bit field is 0xF) report the function switch failed and return;
change clock divisor value or configure the system clock feeding into uSDHC to generate the
card_clk of around 50MHz for high speed mode, 100MHz for SDR50, 200MHz for SDR104, 50MHz for
DDR50;
(data transactions like normal peers)
}
disable_sd_speed_mode(void)
{
set BLKCNT field to 1 (block), set BLKSIZE field to 64 (bytes);
send CMD6, with argument 0x80FFFFF0 and read 64 bytes of data accompanying the R1 response;
check if the bit field 379~376 is 0xF;
if (the bit field is 0xF) report the function switch failed and return;
change clock divisor value or configure the system clock feeding into uSDHC to generate the
card_clk of the desired value below 25MHz;
(data transactions like normal peers)
}
```

26.5.4.3 Query, enable, and disable MMC high-speed mode

```
enable_mmc_high_speed_mode(void)
{
send CMD9 to get CSD value of MMC;
check if the value of SPEC_VER field is 4 or above;
if (SPEC_VER value is less than 4) report the MMC does not support high speed mode and
return;
set BLKCNT field to 1 (block), set BLKSIZE field to 512 (bytes);
send CMD8 to get EXT_CSD value of MMC;
extract the value of CARD_TYPE field to check the 'high speed mode' in this MMC is 26MHz or
52MHz;
send CMD6 with argument 0x1B90100;
send CMD13 to wait card ready (busy line released);
send CMD8 to get EXT_CSD value of MMC;
check if HS_TIMING byte (byte number 185) is 1;
if (HS_TIMING is not 1) report MMC switching to high speed mode failed and return;
change clock divisor value or configure the system clock feeding into uSDHC to generate the
card_clk of around 26MHz or 52MHz according to the CARD_TYPE;
(data transactions like normal peers)
}
disable_mmc_high_speed_mode(void)
{
send CMD6 with argument 0x2B90100;
set BLKCNT field to 1 (block), set BLKSIZE field to 512 (bytes);
send CMD8 to get EXT_CSD value of MMC;
check if HS_TIMING byte (byte number 185) is 0;
if (HS_TIMING is not 0) report the function switch failed and return;
change clock divisor value or configure the system clock feeding into uSDHC to generate the
card_clk of the desired value below 20MHz;
```

```
(data transactions like normal peers)
}
```

26.5.4.4 Set MMC bus width

```
change_mmc_bus_width(void)
{
send CMD9 to get CSD value of MMC;
check if the value of SPEC_VER field is 4 or above;
if (SPEC_VER value is less than 4) report the MMC does not support multiple bit width and
return;
send CMD6 with argument 0x3B70x00; (8-bit(dual data rate), x=6; 4-bit(dual data rate), x=5;8-
bit, x=2; 4-bit, x=1; 1-bit, x=0)
send CMD13 to wait card ready (busy line released);
(data transactions like normal peers)
}
```

26.5.5 ADMA operation

Here are the codes for the ADMA1 and ADMA2 operations.

26.5.5.1 ADMA1 operation

```
Set_adma1_descriptor
{
if (to start data transfer) {
// Make sure the address is 4KB align.
Set 'Set' type descriptor;
{
Set Act bits to 01;
Set [31:12] bits data length (byte unit);
}
Set 'Tran' type descriptor;
{
Set Act bits to 10;
Set [31:12] bits address (4KB align);
}
}
else if (to fetch descriptor at non-continuous address) {
Set Act bits to 11;
Set [31:12] bits the next descriptor address (4KB aligned);
}
else { // other types of descriptor
Set Act bits accordingly
}
if (this descriptor is the last one) {
Set End bit to 1;
}
if (to generate interrupt for this descriptor) {
Set Int bit to 1;
}
Set Valid bit to 1;
}
```

26.5.5.2 ADMA2 operation

```
Set_adma2_descriptor
{
if (to start data transfer) {
// Make sure the address is a 32-bit boundary (lower 2-bit are always '00').
Set higher 32-bit of descriptor for this data transfer initial address;
Set [31:16] bits data length (byte unit);
Set Act bits to '10';
}
else if (to fetch descriptor at non-continuous address) {
Set Act bits to '11';
// Make sure the address is 32-bit boundary (lower 2-bit are always set to '00').
Set higher 32-bit of descriptor for the next descriptor address;
}
else { // other types of descriptor
Set Act bits accordingly
}
if (this descriptor is the last one) {
Set 'End' bit '1';
}
if (to generate interrupt for this descriptor) {
Set 'Int' bit '1';
}
Set the 'Valid' bit to '1';
}
```

26.5.6 Fast boot operation

26.5.6.1 Normal fast boot flow

Here are the steps for normal fast boot flow:

1. Software must configure the SYS_CTRL[INITA] bit to make sure that 74 card clocks are finished.
2. Software must configure the MMC Boot Register (offset 0xc4) bit 6 to 1 (enable boot), and bit 5 to 0 (normal fast boot), and bit 4 to select the ack mode. If the data is sent through the DMA mode, the software should configure bit 7 to enable the automatic stop at block gap feature, and configure bit 3-bit 0 to select the ack timeout value according to the SD CLK frequency.
3. Software then needs to configure the Block Attributes Register to set the block size and count. If in DDR fast boot mode, the block size only can be configured to 512 bytes.
4. Software must configure the Protocol control register to set Data Transfer Width (DTW). If in the DDR fast boot mode, DTW only can be configured to 4-bit/8-bit dataline mode.
5. Software needs to configure the Command Argument Register to set argument if needed (no need in normal fast boot).
6. Software must configure the Transfer Type Register to start the boot process. In normal boot mode, CMDINX, CMDTYP, RSPTYP, CICEN, CCCEN, AC12EN,

BCEN and DMAEN retain the default value, where DPSEL bit is set to 1, DTDSEL is set to 1 and MSBSEL is set to 1.

7. DMAEN should be configured as 0 in the polling mode and if BCEN is configured as 1, it is recommended to configure the number of blocks in the Block Attributes Register to the maximum value. If in DDR fast boot mode, DDR_EN needs to be set to 1.
8. When step 6 is configured, the boot process begins. The software needs to poll the data buffer ready status to read the data from the buffer in time. If a boot timeout happens (ack times out or the first data read times out), an interrupt is triggered, and the software must configure MMC Boot Register to bit 6 to 0 to disable boot. This makes CMD high, then after at least 56 clocks, it is ready to begin a normal initialization process.
9. If there is no timeout, software needs to determine when the data read is finished and then configure MMC Boot Register bit 6 to 0 to disable boot. This render CMD line high and command completed asserted. After at least 56 clocks, it is ready to begin the normal initialization process.
10. You must reset the host and begin the normal process.

26.5.6.2 Alternative fast boot flow

Here are the steps for alternative fast boot flow:

1. Software needs to configure SYS_CTRL[INITA] to make sure 74 card clocks are finished.
2. Software needs to configure MMC Boot Register (offset 0xc4) bit 6 to 1 (enable boot), and bit 5 to 1 (alternative boot), and bit 4 to select the ack mode or not. If data needs to be sent through the DMA mode, then configure bit 7 to enable the automatic stop at block gap feature. Software should also configure bit 3-bit 0 to select the ack timeout value according to the SD clock frequency.
3. Software then needs to configure the Block Attributes Register to set the block size and count. If in the DDR fast boot mode, the block size only can be configured to 512 bytes.
4. Software needs to configure the Protocol control register to set the data transfer width (DTW). If in the DDR fast boot mode, DTW only can be configured to 4-bit/8-bit dataline mode.
5. Software needs to configure Command Argument Register to set argument to 0xFFFFFFFA.
6. Software needs to configure the Transfer Type Register to start the boot process by CMD0 with the 0xFFFFFFFA argument. In alternative boot, CMDINX, CMDTYP, RSPTYP, CICEN, CCCEN, AC12EN, BCEN, and DMAEN retain the default value. DPSEL bit is set to 1, DTDSEL is set to 1, and MSBSEL is set to 1. Note DMAEN

should be configured as 0 in the polling mode, and if BCEN is configured as 1 in polling mode, it is recommended to configure the block count in the Block Attributes Register to the maximum value. If in the DDR fast boot mode, DDR_EN needs to be set to 1.

7. When step 6 is configured, the boot process begins. Software needs to poll the data buffer ready status to read the data from the buffer in time. If there is a boot timeout (ack data timeout in 50ms or data timeout in 1s), the host sends out the interrupt and the software needs to send CMD0 with reset and then configure the boot enable bit to 0 to stop this process.
8. If there is no time out, the software needs to decide when to stop the boot process, and send out the CMD0 with reset and then after the command is completed, configure the MMC Boot Register bit 6 to stop the process. After 8 clocks from the command completion, the slave (card) is ready for the identification step.
9. You must reset the host and begin the normal process.

26.5.6.3 Fast boot application case (in DMA mode)

In the boot application case, because the image destination and the image size are contained in the beginning of the image, it is necessary to switch DMA parameters on the fly during MMC fast boot.

In fast boot, the host can use Advanced DMA2 (ADMA2) with two destinations.

The detailed flow is described below:

1. The software needs to configure INIT_ACTIVE bit (system control register bit 27) to make sure that 74 card clocks are finished.
2. The software needs to configure the MMC Boot Register (offset 0xc4) bit 6 to 1 (enable boot); and bit 5 to 0 (normal fast boot) or 1 (alternative boot); and bit 4 to select the ack mode. In DMA mode, configure bit 7 to 1 to enable the automatic stop at block gap feature. Also configure bits[31-16] to set the (BLK_CNT - VALUE1). Here VALUE1 is the value of the block count that needs to transfer the first time, so that the host stops at the block gap when the uSDHC controller gets VAULE1 blocks from the device. Also, configure bits[3-0] to select the ack timeout value according to the SD clock frequency.
3. The software then needs to configure the Block Attributes Register to set block size and count. If in DDR fast boot mode, the block size only can be configured to 512 bytes. In DMA mode, it is recommended to set the block count (BLK_CNT) to the max value (16'hffff).
4. The software needs to configure Protocol Control Register to set DTW (data transfer width). If in DDR fast boot mode, the DTW only can be configured to 4-bit/8-bit dataline mode.

5. Software enable ADMA2 by configuring Protocol Control Register bits [9-8].
6. The software needs to set at least three pairs of ADMA2 descriptor in boot memory (that is, in IRAM, at least six words). The first pair descriptor defines the start address (that is, IRAM) and data length (that is,512byte*VALUE1) of the first part boot code. The software also needs to set the second pair descriptor, the second start address (any value that is writable), and data length is suggested to set 1~2word (record as VALUE2). Note that the second couple desc also transfers useful data even at lease 1 word, because our ADMA2 cannot support 0 data_length data transfer descriptor.
7. The software needs to configure Command Argument Register to set argument to 0xFFFFFFFA in alternative fast boot and do not need to be set in normal fast boot.
8. The software needs to configure Transfer Type Register to start the boot process. CMDINX, CMDTYP, RSPTYP, CICEN, CCCEN, AC12EN, BCEN, and DMAEN retain the default value. DPSEL bit is set to 1, DTDSEL is set to 1, and MSBSEL is set to 1. DMAEN is configured as 1 in the DMA mode. And, if BCEN is configured as 1, then configure blk no in Bock Attributes Register to the max value. And, if in the DDR fast boot mode, DDR_EN needs to be set to 1.
9. When step 8 is configured, boot process begins, the first VALUE1 block number data gets transferred. The software needs to poll the TC bit (bit1 in Interrupt Status Register) to determine first transfer is ended. Also, the software needs to polling the BGE bit (bit2 in Interrupt Status Register) to determine if the first transfer stops at the block gap.
10. When TC and BGE bits are set to 1, the software can analyze the first code of VALUE1 block, initializes the new memory device, if required, and sets the third pair of descriptors to define the start address and length of the remaining part of the boot code (VALUE3, the remain boot code block). Remember to set the last descriptor with END.
11. The software needs to configure the MMC Boot Register (offset 0xc4) again. Set bit 6 to 1 (enable boot); and bit 5 to 0 (normal fast boot), to 1 (alternative boot); and bit 4 to select the ack mode or not. In the DMA mode, configure bit 7 to 1 for enabling the automatically stop at block gap feature. Also, configure bit31-bit16 to set the (BLK_CNT - (VALUE1+1+VALUE3)), that host stops at block gap when the uSDHC controller gets (VALUE1+1+VALUE3)) blocks from device totally include the blocks received in step 9. And need to configure bit 3-bit0 to select the ack timeout value according to the sd clk frequency. Note that the software does not need to configure the BLK_CNT again, because it is counted down automatically by the uSDHC controller.
12. The software needs to clear the TC and BGE bits and the software needs to clear SABGREQ (bit 16 in the Protocol control register) and set CREQ (bit17 in the Protocol control register) to 1 to resume the data transfer. Host transfers the VALUE2 and VALUE3 data to the destination that is set by descriptor.

13. The software needs to do poll BGE bit to determine if the fast boot is over.

Note:

- When ADMA boot flow starts, for uSDHC, it is like a normal ADMA read operation. So, set ADMA2 descriptor as the normal ADMA2 transfer.
- Need a few words length memory to keep descriptor.
- For the 1~2-word data in second descriptor setting, it is a useful data, so the software needs to deal the data because of the application case.

26.6 Software restrictions

26.6.1 Initialization active

The driver cannot set INITA bit in System Control register when any of the command line or data lines are active, so the driver must ensure both CDIHB and CIHB bits are cleared.

26.6.2 Software polling procedure

For polling read or write, after the software begins a buffer read or write, it must access exactly the number of times as the values set in the Watermark Level Register; moreover, if the block size is not a multiple of the value in the Watermark Level Register (read and write respectively), the software must access exactly the remaining number of words at the end of each block.

For example, for a read operation, if the RD_WML is 4, indicating the watermark level is 16 bytes, block size is 40 bytes, and the block number is 2, then the access times for the burst sequence in the whole transfer process must be 4, 4, 2, 4, 4, 2.

26.6.3 Suspend operation

To suspend the data transfer, the software must inform uSDHC that the suspend command is successfully accepted. To achieve this, after the Suspend command is accepted by the SDIO card, software must send another normal command marked as suspend command (CMDTYP bits set as '01') to inform uSDHC that the transfer is suspended.

If software needs to resume the suspended transfer, it should read the value in BLKCNT register to save the remaining number of blocks before sending the normal command marked as suspend, otherwise on sending such a 'suspend' command, uSDHC treats the current transfer is aborted and change the BLKCNT register to its original value, instead of retaining the remaining number of blocks.

26.6.4 Data length setting

For either ADMA (ADMA1 or ADMA2) transfer, the data in the data buffer must be word aligned, so the data length set in the descriptor must be a multiple of 4.

ADMA1 is 4KB aligned.

26.6.5 (A)DMA address setting

To configure the ADMA1/ADMA2/DMA address register, when TC bit is set, the register always updates itself with the internal address value to support dynamic address synchronization, so the software must ensure that the TC bit is cleared prior to configuring the ADMA1/ADMA2/DMA address register.

26.6.6 Data port access

Data port does not support parallel access. For example, during an internal DMA access, it is not allowed to write any data to the data port by CPU; or during a CPU read operation, it is also prohibited to write any data to the data port, by either CPU or internal DMA. Otherwise the data is corrupted inside the uSDHC buffer.

26.6.7 Change clock frequency

The uSDHC module does not automatically gate off the card clock when the host driver changes the clock frequency. To prevent possible glitch on the card clock, clear the FRC_SDCLK_ON bit when changing the clock divisor value (SDCLKFS or DVS in System Control Register) or setting the RSTA bit.

Also, before changing the clock divisor value, the host driver should make sure that the SDSTB bit is high.

26.6.8 Multi-block read

For pre-defined multi-block read operation, that is, the number of blocks to read has been defined by previous CMD23 for MMC, or pre-defined number of blocks in CMD53 for SDIO/SDCombo, or whatever multi-block read without abort command at card side, an abort command, either automatic or manual CMD12/CMD52, is still required by uSDHC after the pre-defined number of blocks are done, to drive the internal state machine to idle mode.

In this case, the card may not respond to this extra abort command and uSDHC gets response timeout. It is recommended to manually send an abort command with RSPTYP[1:0] both bits cleared.

26.7 uSDHC memory map/register definition

26.7.1 uSDHC register descriptions

This section includes the module memory map and detailed descriptions of all registers.

See the table below for the register memory map of uSDHC. All these registers only support 32-bit accesses.

NOTE

The uSDHC registers are 32-bit wide and only support 32-bit access.

26.7.1.1 uSDHC memory map

uSDHC1 base address: 402C_0000h

uSDHC2 base address: 402C_4000h

Offset	Register	Width (In bits)	Access	Reset value
0h	DMA System Address (DS_ADDR)	32	RW	0000_0000h
4h	Block Attributes (BLK_ATT)	32	RW	0001_0000h
8h	Command Argument (CMD_ARG)	32	RW	0000_0000h
Ch	Command Transfer Type (CMD_XFR_TYP)	32	RW	0000_0000h
10h	Command Response0 (CMD_RSP0)	32	RO	0000_0000h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
14h	Command Response1 (CMD_RSP1)	32	RO	0000_0000h
18h	Command Response2 (CMD_RSP2)	32	RO	0000_0000h
1Ch	Command Response3 (CMD_RSP3)	32	RO	0000_0000h
20h	Data Buffer Access Port (DATA_BUFF_ACC_PORT)	32	RW	0000_0000h
24h	Present State (PRES_STATE)	32	RO	Table 26-8
28h	Protocol Control (PROT_CTRL)	32	RW	0880_0020h
2Ch	System Control (SYS_CTRL)	32	RW	0080_800Fh
30h	Interrupt Status (INT_STATUS)	32	W1C	0000_0000h
34h	Interrupt Status Enable (INT_STATUS_EN)	32	RW	0000_0000h
38h	Interrupt Signal Enable (INT_SIGNAL_EN)	32	RW	0000_0000h
3Ch	Auto CMD12 Error Status (AUTOCMD12_ERR_STATUS)	32	RW	0000_0000h
40h	Host Controller Capabilities (HOST_CTRL_CAP)	32	RW	07F3_B407h
44h	Watermark Level (WTMK_LVL)	32	RW	0810_0810h
48h	Mixer Control (MIX_CTRL)	32	RW	8000_0000h
50h	Force Event (FORCE_EVENT)	32	WORZ	0000_0000h
54h	ADMA Error Status (ADMA_ERR_STATUS)	32	RO	0000_0000h
58h	ADMA System Address (ADMA_SYS_ADDR)	32	RW	0000_0000h
60h	DLL (Delay Line) Control (DLL_CTRL)	32	RW	0000_0000h
64h	DLL Status (DLL_STATUS)	32	RO	0000_0200h
68h	CLK Tuning Control and Status (CLK_TUNE_CTRL_STATUS)	32	RW	0000_0000h
C0h	Vendor Specific Register (VEND_SPEC)	32	RW	3000_7809h
C4h	MMC Boot (MMC_BOOT)	32	RW	0000_0000h
C8h	Vendor Specific 2 Register (VEND_SPEC2)	32	RW	0001_9006h
CCh	Tuning Control (TUNING_CTRL)	32	RW	0021_2800h

26.7.1.2 DMA System Address (DS_ADDR)

26.7.1.2.1 Offset

Register	Offset
DS_ADDR	0h

26.7.1.2.2 Function

This register contains the physical system memory address used for DMA transfers.

26.7.1.2.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DS_ADDR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DS_ADDR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

26.7.1.2.4 Fields

Field	Function
31-0 DS_ADDR	System address **DMA system address / argument 2** When ACMD23_ARGU2_EN is set to 0, SDMA uses this register as system address and supports only 32-bit addressing mode. Auto CMD23 cannot be used with SDMA. When ACMD23_ARGU2_EN is set to 1, SDMA uses ADMA System Address register (05Fh – 058h) instead of this register to support both 32-bit and 64-bit addressing. This register is used only for Argument2 and SDMA may use Auto CMD23. 1. **SDMA system address** Because the address must be word (4 bytes) aligned, the least two bits are reserved and always set to 0. When uSDHC stops a DMA transfer, this register points out the system address of the next contiguous data position. It can be accessed only when no transaction is executing (that is, after a transaction has stopped). Read operation during transfers may return an invalid value. The host driver initializes this register before starting a DMA transaction. After DMA has stopped, the system address of the next contiguous data position can be read from this register. This register is protected during a data transfer. When data lines are active, write to this register is ignored. The host driver waits until the DLA field in the Present State register is cleared, before writing to this register. The uSDHC internal DMA does not support a virtual memory system. It only supports continuous physical memory access. And due to AHB burst limitations, if the burst must cross the 1 KB boundary, uSDHC automatically changes SEQ burst type to NSEQ. Because this register supports dynamic address reflecting, when TC field is set, it automatically alters the value of internal address counter, so the software cannot change this register when TC field is set. Such restriction is also listed in Software restrictions. 2. **Argument 2** This register is used with the Auto CMD23 to set a 32-bit block count value to the argument of the CMD23 while executing Auto CMD23. If Auto CMD23 is used with ADMA, the full 32-bit block count value can be used. If Auto CMD23 is used without ADMA, the available block count value is limited by the Block Count register. 65535 blocks is the maximum value in this case.

26.7.1.3 Block Attributes (BLK_ATT)

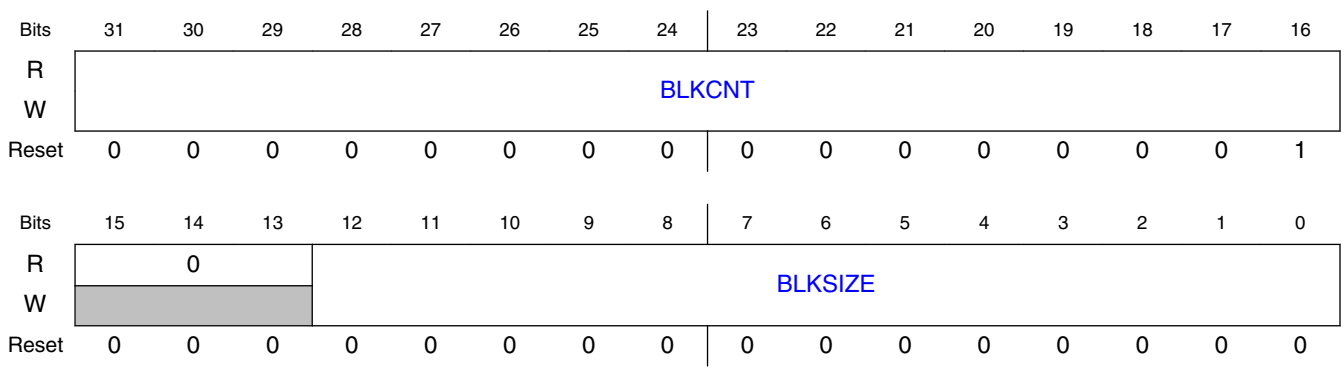
26.7.1.3.1 Offset

Register	Offset
BLK_ATT	4h

26.7.1.3.2 Function

This register is used to configure the number of data blocks and the number of bytes in each block.

26.7.1.3.3 Diagram



26.7.1.3.4 Fields

Field	Function
31-16 BLKCNT	Blocks count for current transfer This field is enabled when the Block Count Enable field in the Transfer Mode register is set to 1 and is valid only for multiple block transfers. For single block transfer, this field always reads as 1. The host driver sets this field to a value between 1 and the maximum block count. The uSDHC module decrements the block count after each block transfer and stops when the count reaches zero. Setting the block count to zero results in no data blocks being transferred. This field should be accessed only when no transaction is executing (that is, after transactions are stopped). During data transfer, read operations on this field may return an invalid value and write operations are ignored. When saving transfer content because of a Suspend command, the number of blocks yet to be transferred can be determined by reading this field. The reading of this field should be applied after transfer is paused by stop at block gap operation and before sending the command marked as suspend.

Table continues on the next page...

Field	Function
	This is because when the Suspend command is sent out, uSDHC treats the current transfer as aborted and change the BLKCNT field back to its original value instead of keeping the dynamical indicator of the remaining block count. When restoring transfer content prior to issuing a Resume command, the host driver restores the previously saved block count. **NOTE:** Although the BLKCNT field is 0 after reset, the read of reset value is 0x1. This is because when MSBSEL field is indicating a single block transfer, the read value of BLKCNT is always 1. 0000000000000000b - Stop count 0000000000000001b - 1 block 0000000000000010b - 2 blocks 1111111111111111b - 65535 blocks
15-13 —	Reserved
12-0 BLKSIZE	Transfer block size This field specifies the block size for block data transfers. Values ranging from 1 byte up to the maximum buffer size can be set. It can be accessed only when no transaction is executing (that is, after a transaction has stopped). Read operations during transfers may return an invalid value, and write operations are ignored. 0000000000000b - No data transfer 0000000000001b - 1 byte 0000000000010b - 2 bytes 0000000000011b - 3 bytes 0000000000100b - 4 bytes 0000111111111b - 511 bytes 0001000000000b - 512 bytes 0100000000000b - 2048 bytes 1000000000000b - 4096 bytes

26.7.1.4 Command Argument (CMD_ARG)

26.7.1.4.1 Offset

Register	Offset
CMD_ARG	8h

26.7.1.4.2 Function

This register contains the SD/MMC command argument.

26.7.1.4.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CMDARG															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CMDARG															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

26.7.1.4.4 Fields

Field	Function
31-0 CMDARG	Command argument The SD/MMC command argument is specified as bits 39-8 of the command format in the SD or MMC specification. This field is write protected when the Command Inhibit (CMD) field in the Present State register is set.

26.7.1.5 Command Transfer Type (CMD_XFR_TYP)

26.7.1.5.1 Offset

Register	Offset
CMD_XFR_TYP	Ch

26.7.1.5.2 Function

This register is used to control the operation of data transfers. The host driver sets this register before issuing a command followed by a data transfer or before issuing a Resume command. To prevent data loss, uSDHC prevents writing to the bits, which are involved in the data transfer of this register, when data transfer is active. These fields are DPSEL, MBSEL, DTDSEL, AC12EN, BCEN, and DMAEN.

The host driver checks the Command Inhibit DAT field (PRES_STATE[CDIHB]) and the Command Inhibit CMD field (PRES_STATE[CIHB]) in the Present State register before writing to this register. When the CDIHB field in the Present State register is set, any attempt to send a command with data by writing to this register is ignored; when the CIHB field is set, any write to this register is ignored.

On sending commands with data transfer involved, it is mandatory that the block size is non-zero. Block count must also be non-zero, or indicated as a single block transfer (bit 5 of this register is '0' when written), or block count is disabled (bit 1 of this register is '0' when written), otherwise uSDHC ignores the sending of this command and do nothing.

If the commands with data transfer do not receive the response in 64 clock cycles, that is, if response time-out happens, uSDHC treats the external device, does not accept the command, and aborts the data transfer. In this scenario, the driver should issue the command again to retry the transfer. It is also possible that for some reason the card responds to the command but uSDHC does not receive the response, and if it is internal DMA (either simple DMA or ADMA) read operation, the external system memory is over-written by the internal DMA with data sent back from the card.

The table below shows the summary of how register settings determine the type of data transfer.

Table 26-6. Transfer type register setting for various transfer types

Multi/single block select	Block count enable	Block count	Function
0	Do not care	Do not care	Single transfer
1	0	Do not care	Infinite transfer
1	1	Positive number	Multiple transfer
1	1	Zero	No data transfer

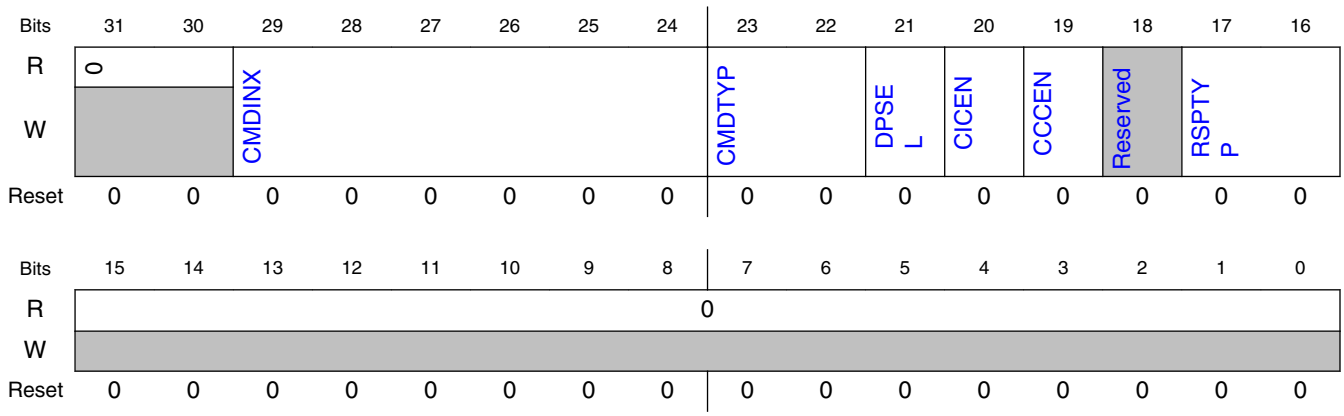
The table below shows the relationship between the Command Index Check Enable and the Command CRC Check Enable, regarding the Response Type bits as well as the name of the response type.

Table 26-7. Relationship between parameters and the name of the response type

Response type	Index check enable	CRC check enable	Name of response type
00	0	0	No response
01	0	1	R2
10	0	0	R3, R4
10	1	1	R1, R5, R6
11	1	1	R1b,R5b

- In the SDIO specification, response type notation for R5b is not defined. R5 includes R5b in the SDIO specification, but R5b is defined in this specification to specify that uSDHC checks the busy status after receiving a response. For example, usually CMD52 is used with R5, but the I/O abort command is used with R5b.
- The CRC fields for R3 and R4 are expected to be all 1 bits. The CRC check is disabled for these response types.

26.7.1.5.3 Diagram



26.7.1.5.4 Fields

Field	Function
31-30 —	Reserved
29-24 CMDINX	Command index These fields are set to the command number that is specified in bits 45-40 of the command-format in the SD Memory Card Physical Layer Specification and SDIO Card Specification.
23-22 CMDTYP	Command type There are three types of special commands: Suspend, Resume, and Abort. These fields are set to 00b for all other commands. • Suspend command: If the Suspend command succeeds, uSDHC assumes that the card bus has been released and that it is possible to issue the next command that uses the DATA line. Because uSDHC does not monitor the content of command response, it does not know if the Suspend command succeeded or not. It is the host driver's responsibility to check the status of the Suspend command and send another command marked as Suspend to inform uSDHC that a Suspend command was successfully issued. See Suspend Resume for more details. After the end bit of command is sent, uSDHC deasserts Read Wait for read transactions and stops checking busy for write transactions. In a 4-bit mode, the interrupt cycle starts. If the Suspend command fails, uSDHC maintains its current state, and the host driver restarts the transfer by setting the Continue Request field in the Protocol Control register.

Table continues on the next page...

Field	Function
	• Resume command: The host driver re-starts the data transfer by restoring the registers saved before sending the Suspend command and then sends the Resume command. The uSDHC module checks for a pending busy state before starting write transfers. • Abort command: If this command is set when executing a read transfer, uSDHC stops reads to the buffer. If this command is set when executing a write transfer, uSDHC stops driving the DATA line. After issuing the Abort command, the host driver should issue a software reset (Abort Transaction). 00b - Normal other commands 01b - Suspend CMD52 for writing bus suspend in CCCR 10b - Resume CMD52 for writing function select in CCCR 11b - Abort CMD12, CMD52 for writing I/O Abort in CCCR
21 DPSEL	Data present select This field is set to 1 to indicate that data is present and is transferred using the DATA line. It is set to 0 for the following: • Commands using only the CMD line (for example, CMD52) • Commands with no data transfer, but using the busy signal on DATA0 line (R1b or R5b (for example, CMD38)) **NOTE:** In resume command, this field is set, and other bits in this register is set the same as when the transfer was initially launched. When this field is set, while the DTDSEL field is 0, writes to the register Transfer Type are ignored. 0b - No data present 1b - Data present
20 CICEN	Command index check enable If this field is set to 1, uSDHC checks the Index field in the response to see if it has the same value as the command index. If it is not, it is reported as a Command Index Error. If this field is set to 0, the Index field is not checked. 0b - Disable command index check 1b - Enables command index check
19 CCCEN	Command CRC check enable If this field is set to 1, uSDHC checks the CRC field in the response. If an error is detected, it is reported as a Command CRC Error. If this field is set to 0, the CRC field is not checked. The number of bits checked by the CRC field value changes according to the length of the response. See RSPTYP[1:0] and Command Transfer Type (CMD_XFR_TYP). 0b - Disables command CRC check 1b - Enables command CRC check
18 —	Reserved
17-16 RSPTYP	Response type select 00b - No response 01b - Response length 136 10b - Response length 48 11b - Response length 48, check busy after response
15-0 —	Reserved

26.7.1.6 Command Response0 (CMD_RSP0)

26.7.1.6.1 Offset

Register	Offset
CMD_RSP0	10h

26.7.1.6.2 Function

This register is used to store part 0 of the response bits from the card.

26.7.1.6.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CMDRSP0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMDRSP0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

26.7.1.6.4 Fields

Field	Function
31-0 CMDRSP0	Command response 0 See Command Response3 (CMD_RSP3) for the mapping of command responses from the SD bus to this field for each response type.

26.7.1.7 Command Response1 (CMD_RSP1)

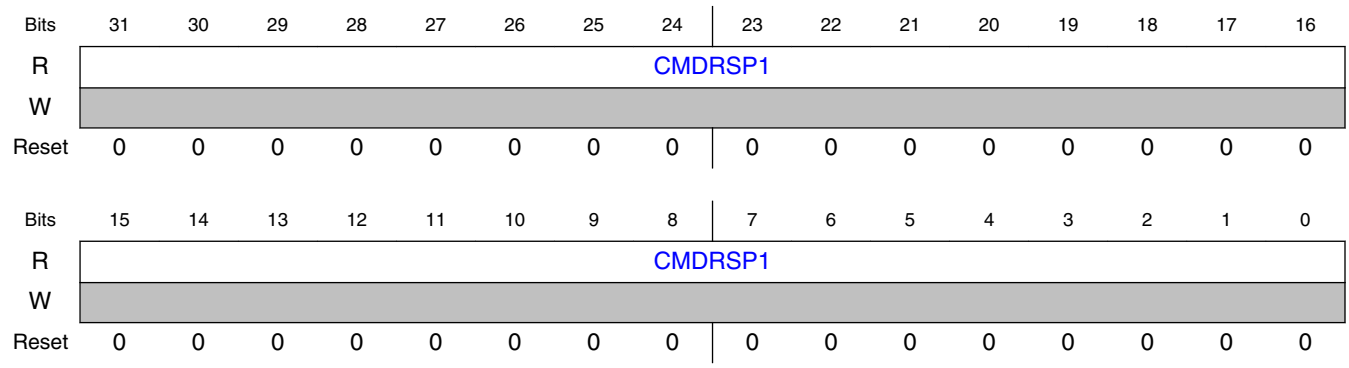
26.7.1.7.1 Offset

Register	Offset
CMD_RSP1	14h

26.7.1.7.2 Function

This register is used to store part 1 of the response bits from the card.

26.7.1.7.3 Diagram



26.7.1.7.4 Fields

Field	Function
31-0 CMDRSP1	Command response 1 See Command Response3 (CMD_RSP3) for the mapping of command responses from the SD bus to this field for each response type.

26.7.1.8 Command Response2 (CMD_RSP2)

26.7.1.8.1 Offset

Register	Offset
CMD_RSP2	18h

26.7.1.8.2 Function

This register is used to store part 2 of the response bits from the card.

26.7.1.8.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CMDRSP2															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMDRSP2															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

26.7.1.8.4 Fields

Field	Function
31-0 CMDRSP2	Command response 2 See Command Response3 (CMD_RSP3) for the mapping of command responses from the SD bus to this field for each response type.

26.7.1.9 Command Response3 (CMD_RSP3)

26.7.1.9.1 Offset

Register	Offset
CMD_RSP3	1Ch

26.7.1.9.2 Function

This register is used to store part 3 of the response bits from the card.

The table below describes the mapping of command responses from the SD bus to Command Response registers for each response type. In this table, R[ ] refers to a bit range within the response data as transmitted on the SD bus.

Table 26-8. Response bit definition for each response type

Response type	Meaning of response	Response field	Response register
R1,R1b (normal response)	Card status	R[39:8]	CMDRSP0

Table continues on the next page...

Table 26-8. Response bit definition for each response type (continued)

Response type	Meaning of response	Response field	Response register
R1b (auto CMD12 response)	Card status for auto CMD12	R[39:8]	CMDRSP3
R2 (CID, CSD register)	CID/CSD register [127:8]	R[127:8]	{CMDRSP3[23:0], CMDRSP2, CMDRSP1, CMDRSP0}
R3 (OCR register)	OCR register for memory	R[39:8]	CMDRSP0
R4 (OCR register)	OCR register for I/O etc.	R[39:8]	CMDRSP0
R5, R5b	SDIO response	R[39:8]	CMDRSP0
R6 (publish RCA)	New published RCA[31:16] and card status[15:0]	R[39:9]	CMDRSP0

This table shows that most responses with a length of 48 (R[47:0]) have 32-bits of the response data (R[39:8]) stored in the CMDRSP0 register. Responses of type R1b (Auto CMD12 responses) have response data bits (R[39:8]) stored in the CMDRSP3 register. Responses with length 136 (R[135:0]) have 120-bits of the response data (R[127:8]) stored in the CMDRSP0, 1, 2, and 3 registers.

To be able to read the response status efficiently, uSDHC only stores part of the response data in the Command Response registers. This enables the host driver to efficiently read 32-bits of response data in one read cycle on a 32-bit bus system. Parts of the response, the Index field and the CRC, are checked by uSDHC (as specified by the Command Index Check Enable and the Command CRC Check Enable bits in the Transfer Type register) and generate an error interrupt if any error is detected. The bit range for the CRC check depends on the response length. If the response length is 48, uSDHC checks R[47:1], and if the response length is 136 the uSDHC checks R[119:1].

Because uSDHC may have a multiple block data transfer executing concurrently with a CMD_wo_DAT command, uSDHC stores the Auto CMD12 response in the CMDRSP3 register. The CMD_wo_DAT response is stored in CMDRSP0. This allows uSDHC to avoid overwriting the Auto CMD12 response with the CMD_wo_DAT and vice versa. When uSDHC modifies part of the Command Response registers, as shown in the table above, it preserves the unmodified bits.

26.7.1.9.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CMDRSP3															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMDRSP3															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

26.7.1.9.4 Fields

Field	Function
31-0 CMDRSP3	Command response 3 See Command Response3 (CMD_RSP3) for the mapping of command responses from the SD bus to this field for each response type.

26.7.1.10 Data Buffer Access Port (DATA_BUFF_ACC_PORT)

26.7.1.10.1 Offset

Register	Offset
DATA_BUFF_ACC_PORT	20h

26.7.1.10.2 Function

The Buffer Data Port register is for 32-bit data access by the Arm platform. When the internal DMA is enabled, any write to this field is ignored, and any read from this field always yields 0s.

26.7.1.10.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DATCONT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATCONT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

26.7.1.10.4 Fields

Field	Function
31-0 DATCONT	Data content This field is used to access the internal buffer.

26.7.1.11 Present State (PRES_STATE)

26.7.1.11.1 Offset

Register	Offset
PRES_STATE	24h

26.7.1.11.2 Function

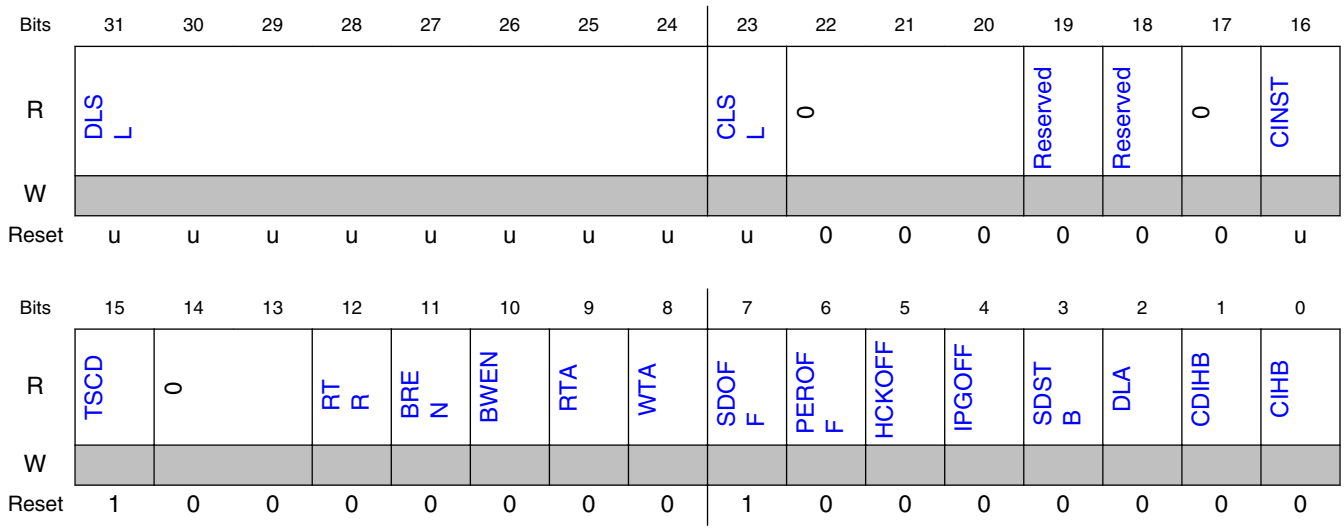
The host driver can get status of uSDHC from this 32-bit read only register.

The host driver can issue CMD0, CMD12, CMD13 (for memory) and CMD52 (for SDIO) when the DATA lines are busy during a data transfer. These commands can be issued when Command Inhibit (CMD) is set to zero. Other commands are issued when Command Inhibit (DATA) is set to zero. Possible changes to the SD Physical Specification may add other commands to this list in the future.

NOTE

The reset value of Present State register depends on board connectivity.

26.7.1.11.3 Diagram



26.7.1.11.4 Fields

Field	Function
31-24 DLSL	DATA[7:0] line signal level This field is used to check the DATA line level to recover from errors, and for debugging. This is especially useful in detecting the busy signal level from DATA0. The reset value is affected by the external pull-up / pull-down resistors. By default, the read value of this field after reset is 8'b11110111, when DATA3 is pulled down and the other lines are pulled up. 00000000b - Data 0 line signal level 00000001b - Data 1 line signal level 00000010b - Data 2 line signal level 00000011b - Data 3 line signal level 00000100b - Data 4 line signal level 00000101b - Data 5 line signal level 00000110b - Data 6 line signal level 00000111b - Data 7 line signal level
23 CLSL	CMD line signal level This field is used to check the CMD line level to recover from errors, and for debugging. The reset value is affected by the external pull-up / pull-down resistor, by default, the read value of this field after reset is 1'b1, when the command line is pulled up.
22-20 —	Reserved
19 —	Reserved
18 —	Reserved
17	Reserved

Table continues on the next page...

Field	Function
—	
16 CINST	Card inserted This field indicates whether a card has been inserted. The uSDHC module debounces this signal so that the host driver does not need to wait for it to stabilize. Changing from a 0 to 1 generates a Card Insertion interrupt in the Interrupt Status register. Changing from a 1 to 0 generates a Card Removal interrupt in the Interrupt Status register. A write to the Force Event Register does not affect this field. The Software Reset For All in the System Control register does not affect this field. A software reset does not affect this field. 0b - Power on reset or no card 1b - Card inserted
15 TSCD	Tap select change done This field indicates the delay setting is effective after write CLK_TUNE_CTRL_STATUS register. 0b - Delay cell select change is not finished. 1b - Delay cell select change is finished.
14-13 —	Reserved
12 RTR	Re-Tuning Request (only for SD3.0 SDR104 mode,and EMMC HS200 mode) Host controller may request host driver to execute re-tuning sequence by setting this field when the data window is shifted by temperature drift and a tuned sampling point does not have a good margin to receive correct data. This field is cleared when a command is issued with setting Execute Tuning field in MIX_CTRL register. Changing of this field from 0 to 1 generates Re-Tuning Event. See Interrupt status registers for more detail. This field isn't set to 1 if Sampling Clock Select in the MIX_CTRL register is set to 0 (using fixed sampling clock). 0b - Fixed or well tuned sampling clock 1b - Sampling clock needs re-tuning
11 BREN	Buffer read enable This status field is used for non-DMA read transfers. The uSDHC module implements an internal buffer to transfer data efficiently. This read only flag indicates that valid data exists in the host side buffer. If this field is high, valid data greater than the watermark level exist in the buffer. A change of this field from 1 to 0 occurs when some reads from the buffer (read DATPORT (Base + 0x20)) are made and the buffer hasn't valid data greater than the watermark level. A change of this field from 0 to1 occurs when there is enough valid data ready in the buffer and the Buffer Read Ready interrupt has been generated and enabled. 0b - Read disable 1b - Read enable
10 BWEN	Buffer write enable This status field is used for non-DMA write transfers. The uSDHC module implements an internal buffer to transfer data efficiently. This read only flag indicates if space is available for write data. If this field is 1, valid data greater than the watermark level can be written to the buffer. A change of this field from 1 to 0 occurs when some writes to the buffer (write DATPORT (Base + 0x20)) are made and the buffer hasn't valid space greater than the watermark level. A change of this field from 0 to 1 occurs when the buffer can hold valid data greater than the write watermark level and the Buffer Write Ready interrupt is generated and enabled. 0b - Write disable 1b - Write enable

Table continues on the next page...

Field	Function
9 RTA	Read transfer active This status field is used for detecting completion of a read transfer. This field is set for either of the following conditions: • After the end field of the read command • When writing a 1 to the Continue Request field in the Protocol Control register to restart a read transfer A transfer complete interrupt is generated when this field changes to 0. This field is cleared for either of the following conditions: • When the last data block as specified by block length is transferred to the System, that is, all data are read away from uSDHC internal buffer. • When all valid data blocks have been transferred from uSDHC internal buffer to the System and no current block transfers are being sent because of the Stop At Block Gap Request being set to 1. 0b - No valid data 1b - Transferring data
8 WTA	Write transfer active This status field indicates a write transfer is active. If this field is 0, it means no valid write data exists in uSDHC. This field is set in either of the following cases: • After the end field of the write command • When writing 1 to the Continue Request field in the Protocol Control register to restart a write transfer This field is cleared in either of the following cases: • After getting the CRC status of the last data block as specified by the transfer count (Single and Multiple) • After getting the CRC status of any block where data transmission is about to be stopped by a Stop At Block Gap Request During a write transaction, a Block Gap Event interrupt is generated when this field is changed to 0, as result of the Stop At Block Gap Request being set. This status is useful for the host driver in determining when to issue commands during Write Busy state. 0b - No valid data 1b - Transferring data
7 SDOFF	SD clock gated off internally This status field indicates that the SD clock is internally gated off, because of buffer over / under-run or read pause without read wait assertion, or the driver set FRC_SDCLK_ON field is 0 to stop the SD clock in idle status. This field is for the host driver to debug data transaction on the SD bus. 0b - SD clock is active. 1b - SD clock is gated off.
6 PEROFF	IPG_PERCLK gated off internally This status field indicates that the IPG_PERCLK is internally gated off. This field is for the host driver to debug transaction on the SD bus. When IPG_CLK_SOFT_EN is cleared, IPG_PERCLK is gated off, otherwise IPG_PERCLK is always active. 0b - IPG_PERCLK is active. 1b - IPG_PERCLK is gated off.
5 HCKOFF	HCLK gated off internally

Table continues on the next page...

Field	Function
	This status field indicates that the HCLK is internally gated off. This field is for the host driver to debug during a data transfer. 0b - HCLK is active. 1b - HCLK is gated off.
4 IPGOFF	Peripheral clock gated off internally This status field indicates that the peripheral clock is internally gated off. This field is for the host driver to debug. 0b - Peripheral clock is active. 1b - Peripheral clock is gated off.
3 SDSTB	SD clock stable This status field indicates that the internal card clock is stable. This field is for the host driver to poll clock status when changing the clock frequency. It is recommended to clear FRC_SDCLK_ON field in System Control register to remove glitches on the card clock when the frequency is changing. Before changing clock divisor value (SDCLKFS or DVS), host driver should make sure the SDSTB field is high. 0b - Clock is changing frequency and not stable. 1b - Clock is stable.
2 DLA	Data line active This status field indicates whether one of the DATA lines on the SD bus is in use. In the case of read transactions: This status indicates if a read transfer is executing on the SD bus. Changes in this value from 1 to 0, between data blocks, generates a Block Gap Event interrupt in the Interrupt Status register. This field is set in either of the following cases: • After the end field of the read command • When writing a 1 to the Continue Request field in the Protocol Control register to restart a read transfer This field is cleared in either of the following cases: • When the end field of the last data block is sent from the SD bus to uSDHC. • When the Read Wait state is stopped by a Suspend command and the DATA2 line is released. The uSDHC module waits at the next block gap by driving Read Wait at the start of the interrupt cycle. If the Read Wait signal is already driven (data buffer cannot receive data), uSDHC can wait for a current block gap by continuing to drive the Read Wait signal. It is necessary to support Read Wait to use the suspend / resume function. This field remains 1 during Read Wait. In the case of write transactions: This status indicates that a write transfer is executing on the SD bus. Changes in this value from 1 to 0 generate a Transfer Complete interrupt in the Interrupt Status register. This field is set in either of the following cases: • After the end field of the write command • When writing to 1 to the Continue Request field in the Protocol Control register to continue a write transfer This field is cleared in either of the following cases:

Table continues on the next page...

Field	Function
	• When the SD card releases Write Busy of the last data block, uSDHC also detects if the output is not busy. If the SD card does not drive the busy signal after the CRC status is received, uSDHC assumes the card drive "Not Busy". • When the SD card releases write busy, prior to waiting for write transfer, and because of a Stop At Block Gap Request. In the case of command with busy pending: This status indicates that a busy state follows the command and the data line is in use. This field is cleared when the DATA0 line is released. 0b - DATA line inactive 1b - DATA line active
1 CDIHB	Command Inhibit Data (DATA) This status field is generated if either the DAT Line Active or the Read Transfer Active is set to 1. If this field is 0, it indicates that uSDHC can issue the next SD / MMC Command. Commands with a busy signal belong to Command Inhibit (DATA) (for example. R1b, R5b type). Changing from 1 to 0 generates a Transfer Complete interrupt in the Interrupt Status register. **NOTE:** The SD host driver can save registers for a suspend transaction after this field has changed from 1 to 0. 0b - Can issue command that uses the DATA line 1b - Cannot issue command that uses the DATA line
0 CIHB	Command inhibit (CMD) If this status bit is 0, it indicates that the CMD line is not in use and uSDHC can issue a SD / MMC command using the CMD line. This field is set also immediately after the Transfer Type register is written. This field is cleared when the command response is received. Even if the Command Inhibit (DATA) is set to 1, commands using only the CMD line can be issued if this field is 0. Changing from 1 to 0 generates a Command Complete interrupt in the Interrupt Status register. If uSDHC cannot issue the command because of a command conflict error (see Command CRC Error) or because of a Command Not Issued By Auto CMD12 Error, this field remains 1 and the Command Complete is not set. The Status of issuing an auto CMD12 does not show on this field. 0b - Can issue command using only CMD line 1b - Cannot issue command

26.7.1.12 Protocol Control (PROT_CTRL)

26.7.1.12.1 Offset

Register	Offset
PROT_CTRL	28h

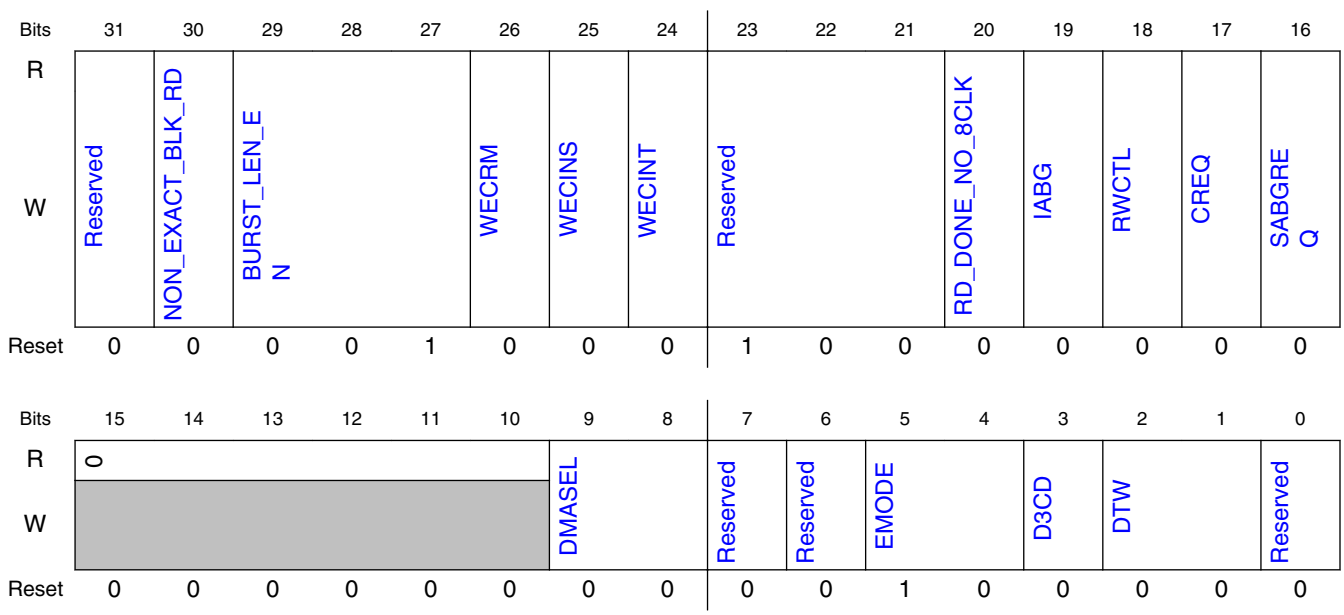
26.7.1.12.2 Function

This register controls three cases to restart the transfer after stop at the block gap. Which case is appropriate depends on whether uSDHC issues a Suspend command or the SD card accepts the Suspend command.

- If the host driver does not issue a Suspend command, the Continue request is used to restart the transfer.
- If the host driver issues a Suspend command and the SD card accepts it, a Resume command is used to restart the transfer.
- If the host driver issues a Suspend command and the SD card does not accept it, the Continue request is used to restart the transfer.

Any time stop at block gap request stops the data transfer, the host driver waits for a Transfer Complete (in the Interrupt Status register), before attempting to restart the transfer. When restarting the data transfer by Continue Request, the host driver clears the Stop At Block Gap Request before or simultaneously.

26.7.1.12.3 Diagram



26.7.1.12.4 Fields

Field	Function
31	Reserved
—	Always write as 0

Table continues on the next page...

Field	Function
30 NON_EXACT_BLK_RD	Non-exact block read Current block read is non-exact block read. It is only used for SDIO. 0b - The block read is exact block read. Host driver does not need to issue abort command to terminate this multi-block read. 1b - The block read is non-exact block read. Host driver needs to issue abort command to terminate this multi-block read.
29-27 BURST_LEN_EN	BURST length enable for INCR, INCR4 / INCR8 / INCR16, INCR4-WRAP / INCR8-WRAP / INCR16-WRAP This field is used to enable or disable the burst length for the external AHB2AXI bridge. It is useful especially for INCR transfer because without burst length indicator, the AHB2AXI bridge does not know the burst length in advance. Without burst length indicator, AHB INCR transfers can only be converted to SINGLEs on the AXI side. 1xxb - Burst length is enabled for INCR4-WRAP / INCR8-WRAP / INCR16-WRAP. x1xb - Burst length is enabled for INCR4 / INCR8 / INCR16. xx1b - Burst length is enabled for INCR.
26 WECRM	Wakeup event enable on SD card removal This field enables a wakeup event, via a card removal, in the Interrupt Status register. FN_WUS (Wakeup Support) in CIS does not affect this field. When this field is set, the Card Removal Status and uSDHC interrupt can be asserted without CLK toggling. When the wakeup feature is not enabled, the CLK must be active to assert the Card Removal Status and uSDHC interrupt. 0b - Disables wakeup event enable on SD card removal 1b - Enables wakeup event enable on SD card removal
25 WECINS	Wakeup event enable on SD card insertion This field enables a wakeup event, via a card insertion, in the Interrupt Status register. FN_WUS (Wakeup Support) in CIS does not affect this field. When this field is set, the Card Insertion Status and uSDHC interrupt can be asserted without CLK toggling. When the wakeup feature is not enabled, the CLK must be active to assert the Card Insertion Status and uSDHC interrupt. 0b - Disable wakeup event enable on SD card insertion 1b - Enable wakeup event enable on SD card insertion
24 WECINT	Wakeup event enable on card interrupt This field enables a wakeup event, via a card interrupt, in the Interrupt Status register. This field can be set to 1 if FN_WUS (Wakeup Support) in CIS is set to 1. When this field is set, the Card Interrupt Status and uSDHC interrupt can be asserted without CLK toggling. When the wakeup feature is not enabled, the CLK must be active to assert the Card Interrupt Status and uSDHC interrupt. 0b - Disables wakeup event enable on card interrupt 1b - Enables wakeup event enable on card interrupt
23-21 —	Reserved Always write as 3'b100
20 RD_DONE_NO_8CLK	Read performed number 8 clock According to the SD/MMC spec, for read data transaction, 8 clocks are needed after the end field of the last data block. So, by default(RD_DONE_NO_8CLK=0), eight clocks are active after the end field of the last read data transaction. However, these 8 clocks should not be active if user wants to use stop at block gap (include the auto stop at block gap in boot mode) feature for read and the RWCTL field (bit18) is not enabled. In this case, software should set RD_DONE_NO_8CLK to avoid these 8 clocks. Otherwise, the device might send extra data to uSDHC while uSDHC ignores these data. In a summary, this field should be set only if the use case needs to use stop at block gap feature while the device can't support the read wait feature.

Table continues on the next page...

Field	Function
19 IABG	Interrupt at block gap This field is valid only in 4-bit mode, of the SDIO card, and selects a sample point in the interrupt cycle. Setting to 1 enables interrupt detection at the block gap for a multiple block transfer. Setting to 0 disables interrupt detection during a multiple block transfer. If the SDIO card cannot signal an interrupt during a multiple block transfer, this field should be set to 0 to avoid an inadvertent interrupt. When the host driver detects an SDIO card insertion, it sets this field according to the CCCR of the card. 0b - Disables interrupt at block gap 1b - Enables interrupt at block gap
18 RWCTL	Read wait control The read wait function provided by this field is optional for SDIO cards. If the card supports read wait, set this field to enable use of the read wait protocol to stop read data using the DATA2 line. Otherwise, uSDHC has to stop the SD clock to hold read data, which restricts commands generation. When the host driver detects an SDIO card insertion, it sets this field according to the CCCR of the card. If the card does not support read wait, this field should never be set to 1; otherwise, DATA line conflicts might occur. If this field is set to 0, stop at block gap during read operation is also supported, but uSDHC stops the SD clock to pause reading operation. 0b - Disables read wait control and stop SD clock at block gap when SABGREQ field is set 1b - Enables read wait control and assert read wait without stopping SD clock at block gap when SABGREQ field is set
17 CREQ	Continue request This field is used to restart a transaction which was stopped using the stop at block gap request. When a suspend operation is not accepted by the card, it is also by setting this field to restart the paused transfer. To cancel stop at the block gap, set stop at block gap request to 0 and set this field to 1 to restart the transfer. The uSDHC module automatically clears this field, therefore it is not necessary for the host driver to set this field to 0. If both stop at block gap request and this field are 1, the continue request is ignored. 0b - No effect 1b - Restart
16 SABGREQ	Stop at block gap request This field is used to stop executing a transaction at the next block gap for both DMA and non-DMA transfers. Until the transfer complete is set to 1, indicating a transfer completion, the host driver leaves this field set to 1. Clearing both the stop at block gap request and continue request does not cause the transaction to restart. Read Wait is used to stop the read transaction at the block gap. The uSDHC module supports the stop at block gap request for write transfers, but for read transfers it requires that the SDIO card support read wait. Therefore, the host driver does not set this field during read transfers unless the SDIO card supports Read Wait and has set the read wait control to 1; otherwise, uSDHC stops the SD bus clock to pause the read operation during block gap. In the case of write transfers in which the host driver writes data to the Data Port register, the host driver sets this field after all block data is written. If this field is set to 1, the host driver does not write data to the Data Port register after a block is sent. Once this field is set, the host driver does not clear this field before the Transfer Complete field in Interrupt Status register is set, otherwise uSDHC's behavior is undefined. This field effects read transfer active, write transfer active, DATA Line Active and Command Inhibit (DATA) in the Present State register. 0b - Transfer 1b - Stop
15-10 —	Reserved
9-8 DMASEL	DMA select This field is valid while DMA (SDMA or ADMA) is enabled and selects the DMA operation.

Table continues on the next page...

Field	Function
	00b - No DMA or simple DMA is selected. 01b - ADMA1 is selected. 10b - ADMA2 is selected. 11b - Reserved
7 —	Reserved
6 —	Reserved
5-4 EMODE	Endian mode This field supports all three endian modes in data transfer. See Data buffer for more details. 00b - Big endian mode 01b - Half word big endian mode 10b - Little endian mode 11b - Reserved
3 D3CD	DATA3 as card detection pin If this field is set, DATA3 should be pulled down to act as a card detection pin. Be cautious when using this feature, because DATA3 is also a chip-select for the SPI mode. A pull-down on this pin and CMD0 might set the card into the SPI mode, which uSDHC does not support. 0b - DATA3 does not monitor card insertion 1b - DATA3 as card detection pin
2-1 DTW	Data transfer width This field selects the data width of the SD bus for a data transfer. The host driver sets it to match the data width of the card. Possible data transfer width is 1-bit, 4-bits or 8-bits. 00b - 1-bit mode 01b - 4-bit mode 10b - 8-bit mode 11b - Reserved
0 —	Reserved

26.7.1.13 System Control (SYS_CTRL)

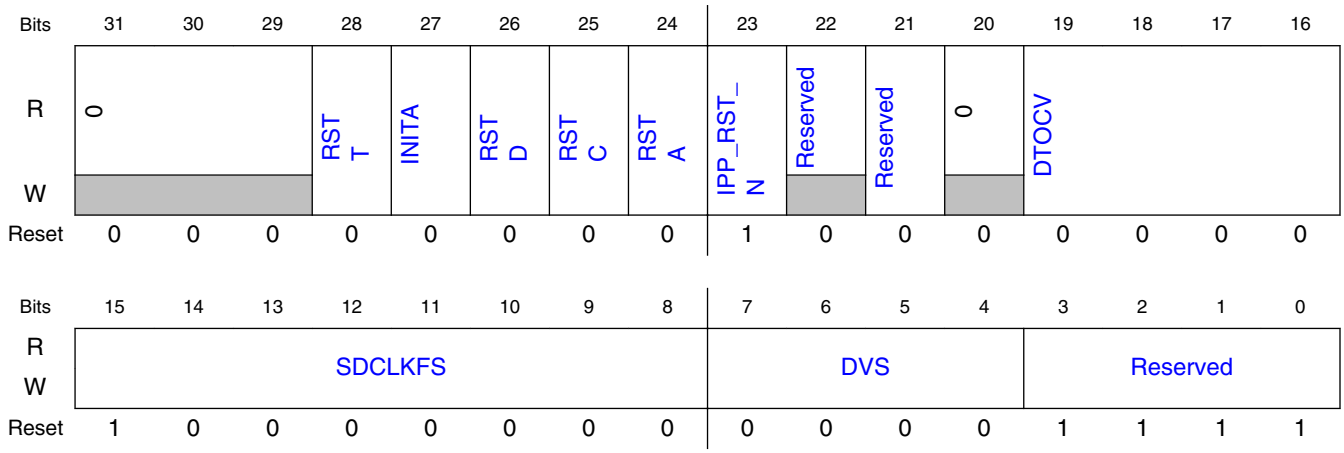
26.7.1.13.1 Offset

Register	Offset
SYS_CTRL	2Ch

26.7.1.13.2 Function

This register provides control of the system. See detail in the field description.

26.7.1.13.3 Diagram



26.7.1.13.4 Fields

Field	Function
31-29 —	Reserved
28 RSTT	Reset tuning When set this field to 1, it resets tuning circuit. After tuning circuits are reset, bit value is 0. Clearing execute_tuning field in AUTOCMD12_ERR_STATUS also sets this field to 1 to reset tuning circuit
27 INITA	Initialization active When this field is set, 80 SD-clocks are sent to the card. After the 80 clocks are sent, this field is self cleared. This field is very useful during the card power-up period when 74 SD-clocks are needed and the clock auto gating feature is enabled. Writing 1 to this bit when this field is already 1 has no effect. Writing 0 to this field at any time has no effect. When either of the CIHB and CDIHB fields in the Present State register are set, writing 1 to this field is ignored (that is, when command line or data lines are active, write to this field is not allowed). On the other-hand, when this field is set, that is, during initialization active period, it is allowed to issue command, and the command bit stream appears on the CMD pad after all 80 clock cycles are done. So, when this command ends, the driver can make sure the 80 clock cycles are sent out. This is very useful when the driver needs to send 80 cycles to the card and does not want to wait till this field is self cleared.

26
RSTD
	Software reset for data line
	Only part of the data circuit is reset. DMA circuit is also reset. After this field is set, the software waits for self-clear.
	The following registers and bits are cleared by this field:
	• Data Port register
	  • Buffer is cleared and initialized
	• Present State register
	  • Buffer read enable
	  • Buffer write enable
	  • Read transfer active
	  • Write transfer active
	  • DATA line active
	  • Command Inhibit (DATA)

Table continues on the next page...

Field	Function

	• Protocol Control register
	  • Continue request
	• Interrupt Status register
	  • Buffer read ready
	  • Buffer write ready
	  • DMA interrupt
	  • Block gap event
	  • Transfer complete
	
	NOTE: When reset, the software must make sure there is no incomplete data transferring. If there is data transfer going on, the software needs to wait TC or DC INT_STATUS register is set.
	0b - No reset
	1b - Reset

25
RSTC
	Software reset for CMD line
	Only part of the command circuit is reset. After this field is set, the software waits for self-clear.
	The following registers and bits are cleared by this field:
	• Present State Register
	  • Command Inhibit (CMD)
	• Interrupt Status Register
	  • Command Complete
	
	0b - No reset
	1b - Reset

24 RSTA	Software reset for all This reset affects the entire host controller except for the card detection circuit. RSTA resets all the registers that can be reset by RSTC/RSTD. During its initialization, the host driver is set this field to 1 to reset uSDHC. The uSDHC module resets this field to 0 when the capabilities registers are valid and the host driver can read them. Additional use of Software Reset For All does not affect the value of the capabilities registers. After this field is set, it is recommended that the host driver reset the external card and re-initialize it. After this field is set, the software should wait for self-clear. In tuning process, after every CMD19 is finished, this field is set to retest uSDHC. **NOTE:** When reset, the software must make sure there is no incomplete data transferring. If there is data transfer going on, the software needs to wait TC or DC INT_STATUS register is set. 0b - No reset 1b - Reset
23 IPP_RST_N	Hardware reset This field's value is output to card through pad directly to hardware reset pin of the card if the card supports this feature.
22 —	Reserved
21 —	Reserved
20 —	Reserved
19-16 DTOCV	Data timeout counter value This value determines the interval by which DAT line timeouts are detected. See the Data Timeout Error field in the Interrupt Status register for information on factors that dictate time-out generation. Time-out clock frequency is generated by dividing the base clock SDCLK value by this value.

Table continues on the next page...

Field	Function
	The host driver can clear the Data Timeout Error Status Enable (in the Interrupt Status Enable register) to prevent inadvertent time-out events. 0000b - SDCLK x 2^{14} 0001b - SDCLK x 2^{15} 0010b - SDCLK x 2^{16} 0011b - SDCLK x 2^{17} 0100b - SDCLK x 2^{18} 0101b - SDCLK x 2^{19} 0110b - SDCLK x 2^{20} 0111b - SDCLK x 2^{21} 1000b - SDCLK x 2^{22} 1001b - SDCLK x 2^{23} 1010b - SDCLK x 2^{24} 1011b - SDCLK x 2^{25} 1100b - SDCLK x 2^{26} 1101b - SDCLK x 2^{27} 1110b - SDCLK x 2^{28} 1111b - SDCLK x 2^{29}
15-8 SDCLKFS	SDCLK frequency select This field is used to select the frequency of the SDCLK pin. The frequency is not programmed directly, rather this field holds the prescaler (of this register) and divisor (next register) of the Base Clock Frequency register. In Single Data Rate mode (DDR_EN field of MIXERCTRL is '0') Only the following settings are allowed: 80h) Base clock divided by 256 40h) Base clock divided by 128 20h) Base clock divided by 64 10h) Base clock divided by 32 08h) Base clock divided by 16 04h) Base clock divided by 8 02h) Base clock divided by 4 01h) Base clock divided by 2 00h) Base clock divided by 1 While in Dual Data Rate mode (DDR_EN field of MIXERCTRL is '1') Only the following settings are allowed: 80h) Base clock divided by 512 40h) Base clock divided by 256 20h) Base clock divided by 128 10h) Base clock divided by 64 08h) Base clock divided by 32 04h) Base clock divided by 16 02h) Base clock divided by 8 01h) Base clock divided by 4 00h) Base clock divided by 2 When the software changes the DDR_EN field, SDCLKFS might need to be changed also.

Table continues on the next page...

Field	Function
	In Single Data Rate mode, setting 00h bypasses the frequency prescaler of the SD clock. Multiple bits must not be set, or the behavior of this prescaler is undefined. The two default divider values can be calculated by the frequency of ipg_perclk and the following Divisor bits. The frequency of SDCLK is set by the following formula: Clock Frequency = (Base Clock) / (prescaler x divisor) For example, in Single Data Rate mode, if the Base Clock Frequency is 96 MHz, and the target frequency is 25 MHz, then choosing the prescaler value of 01h and divisor value of 1h yields 24 MHz, which is the nearest frequency less than or equal to the target. Similarly, to approach a clock value of 400 kHz, the prescaler value of 08h and divisor value of eh yields the exact clock value of 400 kHz. The reset value of this field is 80h, so if the input Base Clock (ipg_perclk) is about 96 MHz, the default SD clock after reset is 375 kHz. Before changing clock divisor value (SDCLKFS or DVS), host driver should make sure the SDSTB field is high. If setting SDCLKFS and DVS can generate the same clock frequency,(for example, in SDR mode, SDCLKFS = 01h is same as DVS = 01h.), SDCLKFS is highly recommended. For HS200 and SDR104 mode, if DLL is to be used, make sure SDCLKFS is used so that REF_CLK, generated for PARTS DLL, is at the same frequency as SD_CLK.
7-4 DVS	Divisor This field is used to provide a more exact divisor to generate the desired SD clock frequency. Note the divider can even support odd divisors without deterioration of duty cycle. Before changing clock divisor value (SDCLKFS or DVS), Host driver should make sure the SDSTB field is high. The settings are as follows: 0000b - Divide-by-1 0001b - Divide-by-2 1110b - Divide-by-15 1111b - Divide-by-16
3-0 —	Reserved Always write as 1.

26.7.1.14 Interrupt Status (INT_STATUS)

26.7.1.14.1 Offset

Register	Offset
INT_STATUS	30h

26.7.1.14.2 Function

An interrupt is generated when the normal interrupt signal enable is enabled and at least one of the status fields is set to 1. For all fields, writing 1 to a bit clears it; writing to 0 keeps the bit unchanged. More than one status can be cleared with a single register write. For card interrupt, before writing 1 to clear, it is required that the card stops asserting the interrupt, meaning that when the card driver services the interrupt condition; otherwise, the CINT field is asserted again.

The table below shows the relationship between the command timeout error and the command complete.

Table 26-9. uSDHC status for command timeout error/command complete bit combinations

Command CRC Error	Command timeout error	Meaning of the status
0	0	X
X	1	Response not received within 64 SDCLK cycles
1	0	Response received

The table below shows the relationship between the transfer complete and the data timeout error.

Table 26-10. uSDHC status for data timeout error/transfer complete bit combinations

Transfer complete	Data timeout error	Meaning of the status
0	0	X
0	1	Timeout occurred during transfer
1	X	Data transfer complete

The table below shows the relationship between the command CRC error and command timeout error.

Table 26-11. uSDHC status for command CRC error/command timeout error bit combinations

Command complete	Command timeout error	Meaning of the status
0	0	No error
0	1	Response timeout error
1	0	Response CRC error
1	1	CMD line conflict

26.7.1.14.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	0		DMAE	0	TNE	0	AC12E	0	DEBE	DCE	DTOE	CIE	CEBE	CCE	CTOE
W				W1C		W1C		W1C		W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ERR_INT_STATUS	TP	0	RTE	0	0		CINT	CRM	CINS	BRR	BWR	DINT	BGE	TC	CC
W		W1C		W1C				W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

26.7.1.14.4 Fields

Field	Function
31 —	Reserved
30-29 —	Reserved
28 DMAE	DMA error Occurs when an Internal DMA transfer has failed. This field is set to 1, when some error occurs in the data transfer. This error can be caused by either simple DMA or ADMA, depending on which DMA is in use. The value in DMA System Address register is the next fetch address where the error occurs. Because any error corrupts the whole data block, the host driver restarts the transfer from the corrupted block boundary. The address of the block boundary can be calculated either from the current DS_ADDR value or from the remaining number of blocks and the block size. 0b - No error 1b - Error
27 —	Reserved
26 TNE	Tuning error: (only for SD3.0 SDR104 mode and EMMC HS200 mode) This field is set when an unrecoverable error is detected in a tuning circuit. By detecting Tuning Error, host driver needs to abort a command executing and perform tuning.
25 —	Reserved

Table continues on the next page...

Field	Function
24 AC12E	Auto CMD12 error Occurs when detecting that one of the fields in the Auto CMD12 Error Status register has changed from 0 to 1. This field is set to 1, not only when the errors in Auto CMD12 occur, but also, when the Auto CMD12 is not executed due to the previous command error. 0b - No error 1b - Error
23 —	Reserved
22 DEBE	Data end bit error Occurs either when detecting 0 at the end field position of read data that uses the DATA line, or at the end field position of the CRC. This field is not asserted in tuning process. 0b - No error 1b - Error
21 DCE	Data CRC error Occurs when detecting a CRC error when transferring read data that uses the DATA line, or when detecting the Write CRC status having a value other than 010. This field is not asserted in tuning process. 0b - No error 1b - Error
20 DTOE	Data timeout error Occurs when detecting one of following time-out conditions. • Busy time-out for R1b, R5b type • Busy time-out after Write CRC status • Read Data time-out. This field is not asserted in tuning process. 0b - No error 1b - Time out
19 CIE	Command index error Occurs if a command index error occurs in the command response. This field is not asserted in tuning process. 0b - No error 1b - Error
18 CEBE	Command end bit error Occurs when detecting that the end field of a command response is 0. This field is not asserted in tuning process. 0b - No error 1b - End bit error generated
17 CCE	Command CRC error Command CRC Error is generated in two cases.

Table continues on the next page...

Field	Function
	• If a response is returned and the Command Timeout Error is set to 0 (indicating no time-out), this field is set when detecting a CRC error in the command response. • The uSDHC module detects a CMD line conflict by monitoring the CMD line when a command is issued. If uSDHC drives the CMD line to 1, but detects 0 on the CMD line at the next SDCLK edge, then uSDHC aborts the command (Stop driving CMD line) and set this bit to 1. The Command Timeout Error should also be set to 1 to distinguish CMD line conflict. This field is not asserted in tuning process. 0b - No error 1b - CRC error generated
16 CTOE	Command timeout error Occurs only if no response is returned within 64 SDCLK cycles from the end field of the command. If uSDHC detects a CMD line conflict, in which case a Command CRC Error is also be set (as shown in Interrupt Status (INT_STATUS)), this field is set without waiting for 64 SDCLK cycles. This is because the command is aborted by uSDHC. This field is not asserted in tuning process. 0b - No error 1b - Time out
15 ERR_INT_STATUS	Error Interrupt Status This bit is set when any of the error status bit DMAE, AC12E, DEBE, DCE, DTOE, CIE, CEBE, CCE and CTOE is set.
14 TP	Tuning pass:(only for SD3.0 SDR104 mode and EMMC HS200 mode) Current CMD19 transfer is done successfully. That is, current sampling point is correct.
13 —	Reserved
12 RTE	Re-tuning event: (only for SD3.0 SDR104 mode and EMMC HS200 mode) This status is set if Re-Tuning Request in the Present State register changes from 0 to 1. host controller requests host driver to perform re-tuning for next data transfer. Current data transfer (not large block count) can be completed without re-tuning. 0b - Re-tuning is not required. 1b - Re-tuning should be performed.
11 —	Reserved
10-9 —	Reserved
8 CINT	Card interrupt This status field is set when an interrupt signal is detected from the external card. In 1-bit mode, uSDHC detects the Card Interrupt without the SD clock to support wakeup. In 4-bit mode, the card interrupt signal is sampled during the interrupt cycle, so the interrupt from card can only be sampled during interrupt cycle, introducing some delay between the interrupt signal from the SDIO card and the interrupt to the host system. Writing this field to 1 can clear this field, but as the interrupt source from the SDIO card does not clear, this field is set again. to clear this field, it is required to reset the interrupt source from the external card followed by a writing 1 to this field. When this status has been set, and the host driver needs to service this interrupt, the Card Interrupt Signal Enable in the Interrupt Signal Enable register should be 0 to stop driving the interrupt signal to the host system. After completion of the card interrupt service (It should reset the interrupt sources in the SDIO card and the interrupt signal might not be asserted), write 1 to clear this field, set the Card Interrupt Signal Enable to 1, and start sampling the interrupt signal again.

Table continues on the next page...

Field	Function
	0b - No card interrupt 1b - Generate card interrupt
7 CRM	Card removal This status field is set if the Card Inserted field in the Present State register changes from 1 to 0. When the host driver writes this field to 1 to clear this status, the status of the Card Inserted in the Present State register should be confirmed. Because the card state might possibly be changed when the host driver clears this field and the interrupt event might not be generated. When this field is cleared, it is set again if no card is inserted. to leave it cleared, clear the Card Removal Status Enable field in Interrupt Status Enable register. 0b - Card state unstable or inserted 1b - Card removed
6 CINS	Card insertion This status field is set if the Card Inserted field in the Present State register changes from 0 to 1. When the host driver writes this field to 1 to clear this status, the status of the Card Inserted in the Present State register should be confirmed. Because the card state might possibly be changed when the host driver clears this field and the interrupt event might not be generated. When this field is cleared, it is set again if a card is inserted. to leave it cleared, clear the Card Inserted Status Enable field in Interrupt Status Enable register. 0b - Card state unstable or removed 1b - Card inserted
5 BRR	Buffer read ready This status field is set if the Buffer Read Enable field, in the Present State register, changes from 0 to 1. See the Buffer Read Enable field in the Present State register for additional information. This field indicates that cmd19 is finished in tuning process. 0b - Not ready to read buffer 1b - Ready to read buffer
4 BWR	Buffer write ready This status field is set if the Buffer Write Enable field, in the Present State register, changes from 0 to 1. See the Buffer Write Enable field in the Present State register for additional information. 0b - Not ready to write buffer 1b - Ready to write buffer
3 DINT	DMA interrupt Occurs only when the internal DMA finishes the data transfer successfully. Whenever errors occur during data transfer, this field does not be set. Instead, the DMAE field is set. Either Simple DMA or ADMA finishes data transferring, this field is set. 0b - No DMA interrupt 1b - DMA interrupt is generated.
2 BGE	Block gap event If the Stop At Block Gap Request field in the Protocol Control register is set, this field is set when a read or write transaction is stopped at a block gap. If Stop At Block Gap Request is not set to 1, this field is not set to 1. In the case of a Read Transaction: This field is set at the falling edge of the DATA Line Active Status (When the transaction is stopped at SD bus timing). The Read Wait must be supported to use this function. In the case of Write Transaction: This field is set at the falling edge of Write Transfer Active Status (After getting CRC status at SD Bus timing). 0b - No block gap event

Table continues on the next page...

Field	Function
	1b - Transaction stopped at block gap
1 TC	Transfer complete This field is set when a read or write transfer is completed. In the case of a Read Transaction: This field is set at the falling edge of the Read Transfer Active Status. There are two cases in which this interrupt is generated. The first is when a data transfer is completed as specified by the data length (after the last data has been read to the host system). The second is when data has stopped at the block gap and completed the data transfer by setting the Stop At Block Gap Request field in the Protocol Control register (after valid data has been read to the host system). In the case of a Write Transaction: This field is set at the falling edge of the DATA Line Active Status. There are two cases in which this interrupt is generated. The first is when the last data is written to the SD card as specified by the data length and the busy signal is released. The second is when data transfers are stopped at the block gap, by setting the Stop At Block Gap Request field in the Protocol Control register, and the data transfers are completed. (after valid data is written to the SD card and the busy signal released). In the case of a command with busy, this field is set when busy is deasserted. This field is not asserted in tuning process. 0b - Transfer does not complete 1b - Transfer complete
0 CC	Command complete This field is set when you receive the end field of the command response (except auto CMD12). See the Command Inhibit (CMD) in the Present State register. This field is not asserted in tuning process. 0b - Command not complete 1b - Command complete

26.7.1.15 Interrupt Status Enable (INT_STATUS_EN)

26.7.1.15.1 Offset

Register	Offset
INT_STATUS_EN	34h

26.7.1.15.2 Function

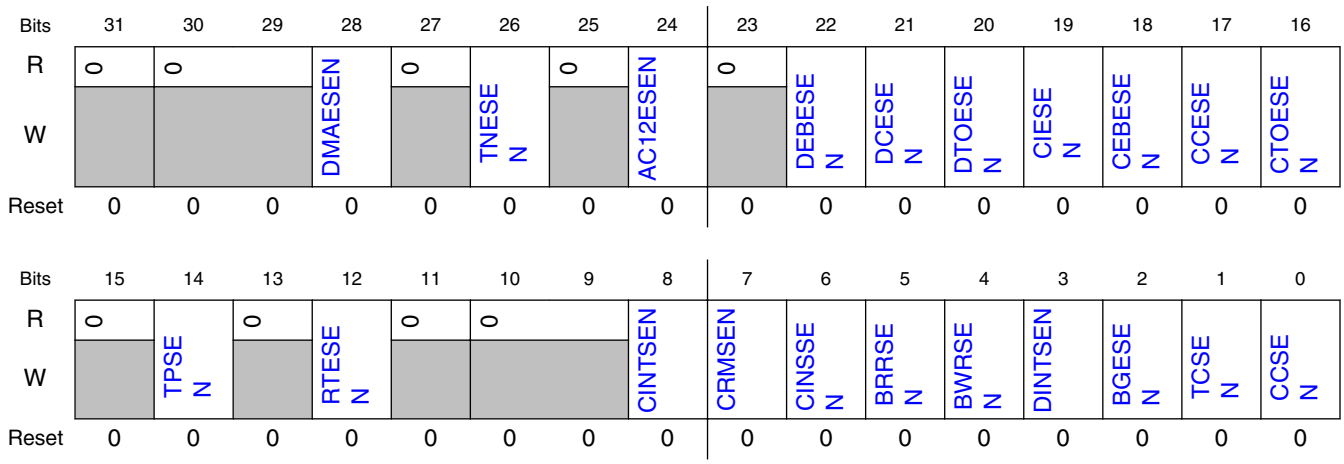
Setting any bit in this register to 1 enables the corresponding interrupt status be requested to the system. If any bit is set to 0, the corresponding interrupt request gets blocked.

- Depending on IABG field setting, uSDHC might be programmed to sample the card interrupt signal during the interrupt period and hold its value in the flip-flop. There

are some delays on the Card Interrupt, asserted from the card, to the time the host system is informed.

- To detect a CMD line conflict, the host driver must set both Command Timeout Error Status Enable and Command CRC Error Status Enable to 1.

26.7.1.15.3 Diagram



26.7.1.15.4 Fields

Field	Function
31 —	Reserved
30-29 —	Reserved
28 DMAESEN	DMA error status enable 0b - Masked 1b - Enabled
27 —	Reserved
26 TNESEN	Tuning error status enable 0b - Masked 1b - Enabled
25 —	Reserved
24 AC12ESEN	Auto CMD12 error status enable 0b - Masked 1b - Enabled
23 —	Reserved

Table continues on the next page...

Field	Function
22 DEBESEN	Data end bit error status enable 0b - Masked 1b - Enabled
21 DCESEN	Data CRC error status enable 0b - Masked 1b - Enabled
20 DTOESEN	Data timeout error status enable 0b - Masked 1b - Enabled
19 CIESEN	Command index error status enable 0b - Masked 1b - Enabled
18 CEBESEN	Command end bit error status enable 0b - Masked 1b - Enabled
17 CCESEN	Command CRC error status enable 0b - Masked 1b - Enabled
16 CTOESEN	Command timeout error status enable 0b - Masked 1b - Enabled
15 —	Reserved
14 TPSEN	Tuning pass status enable 0b - Masked 1b - Enabled
13 —	Reserved
12 RTESEN	Re-tuning event status enable 0b - Masked 1b - Enabled
11 —	Reserved
10-9 —	Reserved
8 CINTSEN	Card interrupt status enable If this field is set to 0, uSDHC clears the interrupt request to the system. The Card Interrupt detection is stopped when this field is cleared and restarted when this field is set to 1. The host driver should clear the Card Interrupt Status Enable before servicing the Card Interrupt and should set this field again after all interrupt requests from the card are cleared to prevent inadvertent interrupts. 0b - Masked 1b - Enabled
7 CRMSEN	Card removal status enable 0b - Masked 1b - Enabled
6 CINSSEN	Card insertion status enable 0b - Masked 1b - Enabled

Table continues on the next page...

Field	Function
5 BRRSEN	Buffer read ready status enable 0b - Masked 1b - Enabled
4 BWRSEN	Buffer write ready status enable 0b - Masked 1b - Enabled
3 DINTSEN	DMA interrupt status enable 0b - Masked 1b - Enabled
2 BGESEN	Block gap event status enable 0b - Masked 1b - Enabled
1 TCSEN	Transfer complete status enable 0b - Masked 1b - Enabled
0 CCSEN	Command complete status enable 0b - Masked 1b - Enabled

26.7.1.16 Interrupt Signal Enable (INT_SIGNAL_EN)

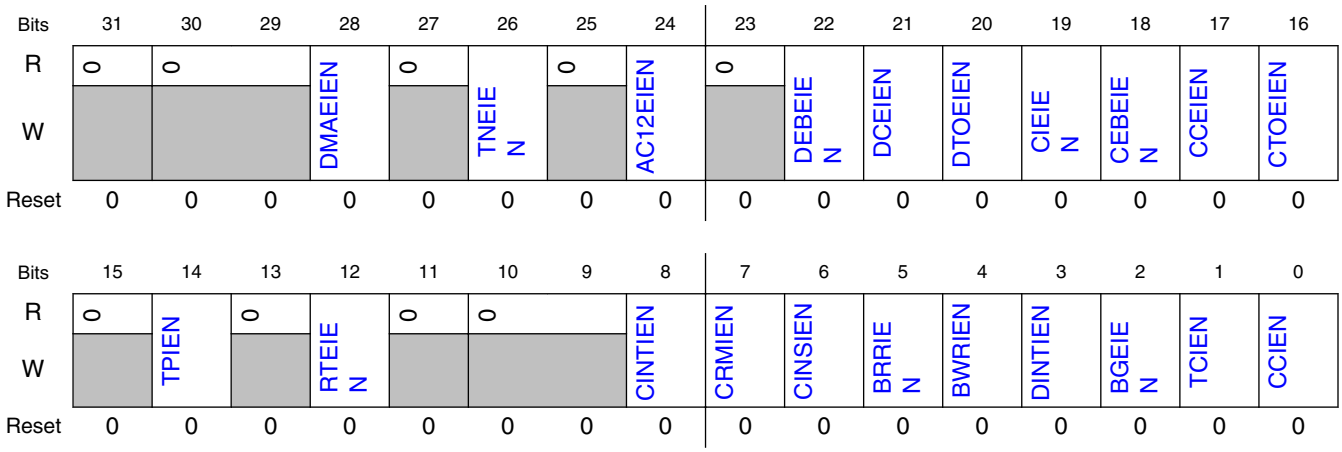
26.7.1.16.1 Offset

Register	Offset
INT_SIGNAL_EN	38h

26.7.1.16.2 Function

This register is used to select which interrupt status is indicated to the host system as the interrupt. These status fields all share the same interrupt lines. Setting any of these fields to 1 enables interrupt generation. The corresponding Status register field generates an interrupt when the corresponding interrupt signal enable field is set.

26.7.1.16.3 Diagram



26.7.1.16.4 Fields

Field	Function
31 —	Reserved
30-29 —	Reserved
28 DMAEIEN	DMA error interrupt enable 0b - Masked 1b - Enable
27 —	Reserved
26 TNEIEN	Tuning error interrupt enable 0b - Masked 1b - Enabled
25 —	Reserved
24 AC12EIEN	Auto CMD12 error interrupt enable 0b - Masked 1b - Enabled
23 —	Reserved
22 DEBEIEN	Data end bit error interrupt enable 0b - Masked 1b - Enabled
21 DCEIEN	Data CRC error interrupt enable 0b - Masked 1b - Enabled
20	Data timeout error interrupt enable 0b - Masked

Table continues on the next page...

Field	Function
DTOEIEN	1b - Enabled
19 CIEIEN	Command index error interrupt enable 0b - Masked 1b - Enabled
18 CEBEIEN	Command end bit error interrupt enable 0b - Masked 1b - Enabled
17 CCEIEN	Command CRC error interrupt enable 0b - Masked 1b - Enabled
16 CTOEIEN	Command timeout error interrupt enable 0b - Masked 1b - Enabled
15 —	Reserved
14 TPIEN	Tuning Pass interrupt enable 0b - Masked 1b - Enabled
13 —	Reserved
12 RTEIEN	Re-tuning event interrupt enable 0b - Masked 1b - Enabled
11 —	Reserved
10-9 —	Reserved
8 CINTIEN	Card interrupt enable 0b - Masked 1b - Enabled
7 CRMIEN	Card removal interrupt enable 0b - Masked 1b - Enabled
6 CINSIEN	Card insertion interrupt enable 0b - Masked 1b - Enabled
5 BRRIEN	Buffer read ready interrupt enable 0b - Masked 1b - Enabled
4 BWRIEN	Buffer write ready interrupt enable 0b - Masked 1b - Enabled
3 DINTIEN	DMA interrupt enable 0b - Masked 1b - Enabled
2 BGEIEN	Block gap event interrupt enable 0b - Masked 1b - Enabled

Table continues on the next page...

Field	Function
1 TCIEN	Transfer complete interrupt enable 0b - Masked 1b - Enabled
0 CCIEN	Command complete interrupt enable 0b - Masked 1b - Enabled

26.7.1.17 Auto CMD12 Error Status (AUTOCMD12_ERR_STATUS)

26.7.1.17.1 Offset

Register	Offset
AUTOCMD12_ERR_STATUS	3Ch

26.7.1.17.2 Function

When the Auto CMD12 Error Status field in the Status register is set, the host driver checks this register to identify what kind of error the Auto CMD12 / CMD 23 indicated. Auto CMD23 errors are indicated in field 04-01. This register is valid only when the Auto CMD12 Error status field is set.

The table below shows the relationship between the Auto CMGD12 CRC Error and the Auto CMD12 Command Timeout Error.

Table 26-12. Relationship between command CRC error and command timeout error for auto CMD12

Auto CMD12 CRC error	Auto CMD12 timeout error	Type of error
0	0	No error
0	1	Response timeout error
1	0	Response CRC error
1	1	CMD line conflict

Changes in Auto CMD12 Error Status register can be classified in three scenarios:

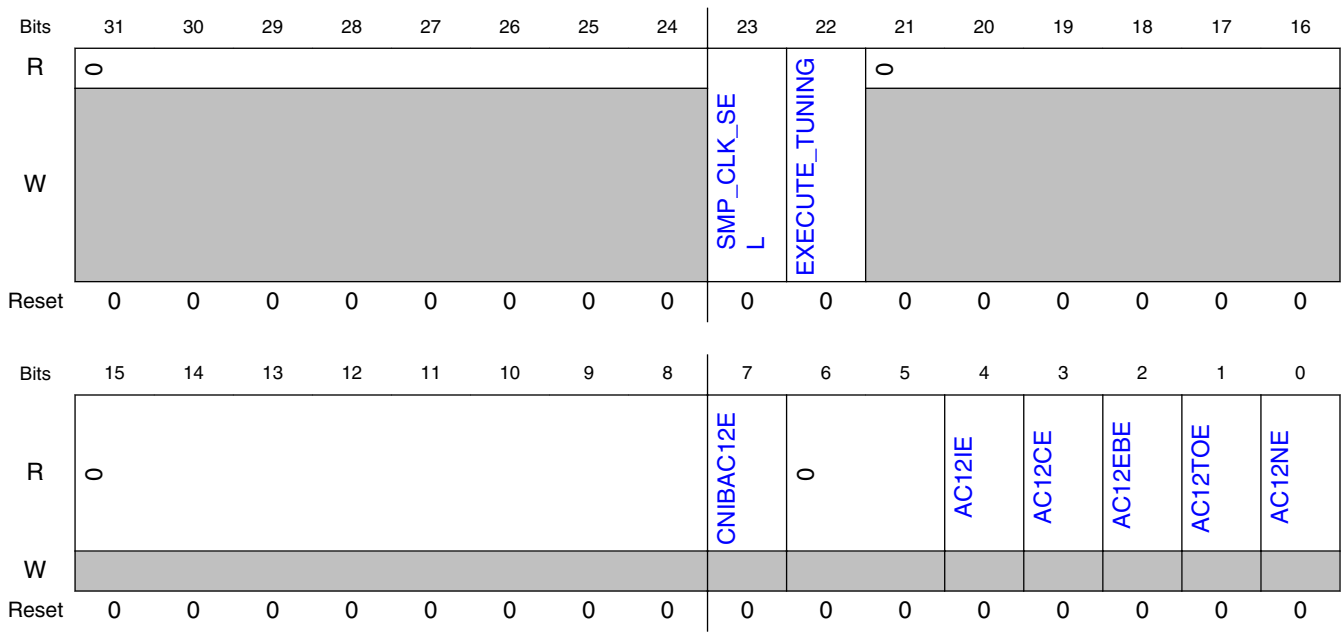
- When uSDHC is going to issue an Auto CMD12

- Set field 0 to 1 if the Auto CMD12 cannot be issued due to an error in the previous command
- Set field 0 to 0 if the Auto CMD12 is issued

- At the end field of an Auto CMD12 response
  - Check errors correspond to fields 1-4
  - Set fields 1-4 corresponding to detected errors
  - Clear fields 1-4 corresponding to detected errors
- Before reading the Auto CMD12 Error Status field 7
  - Set field 7 to 1 if there is a command that can't be issued
  - Clear field 7 if there is no command to issue

The timing for generating the Auto CMD12 Error and writing to the Command register are asynchronous. After that, field 7 is sampled when the driver is not writing to the Command register. So, it is suggested to read this register only when the AC12E field in Interrupt Status register is set. An Auto CMD12 Error Interrupt is generated when one of the error fields (0-4) is set to 1. The Command Not Issued By Auto CMD12 Error does not generate an interrupt.

26.7.1.17.3 Diagram



26.7.1.17.4 Fields

Field	Function
31-24	Reserved

Table continues on the next page...

Field	Function
—	
23 SMP_CLK_SEL	Sample clock select This field is used to select sampling clock to receive CMD and DATA. This field is functional only when TUNING_CTRL[STD_TUNING_EN] is set. Otherwise, this field is reserved. This field is controlled by re-tuning procedure and is valid after the tuning procedure is complete (AUTOCMD12_ERR_STATUS[EXECUTE_TUNING]=0). A value of '1' means that tuning has been completed successfully and a value of '0' implies the tuning has failed. Writing '1' to this field is meaningless and ignored when the tuning has failed. The tuning circuit is reset by writing '0' to this field. This field can be cleared by setting AUTOCMD12_ERR_STATUS[EXECUTE_TUNING]=1. Once the tuning circuit is reset, it takes time to complete tuning sequence. Therefore, host driver should keep this field as 1 to perform the re-tuning sequence in a short time. This field cannot be changed while the Host controller is receiving response or a read data block. 0b - Fixed clock is used to sample data 1b - Tuned clock is used to sample data
22 EXECUTE_TUNING	Execute tuning This field is used to start the tuning procedure. This field is functional only when TUNING_CTRL[STD_TUNING_EN] is set. Otherwise, this field is reserved. This field is set to start tuning procedure and automatically cleared when tuning procedure is completed. The result of tuning is indicated to AUTOCMD12_ERR_STATUS[SMP_CLK_SEL]. 0b - Tuning procedure is aborted 1b - Start tuning procedure
21-8 —	Reserved
7 CNIBAC12E	Command not issued by Auto CMD12 error Setting this field to 1 means CMD_wo_DAT is not executed due to an Auto CMD12 error (D04-D01) in this register. 0b - No error 1b - Not issued
6-5 —	Reserved
4 AC12IE	Auto CMD12 / 23 index error Occurs if the command index error occurs in response to a command. 0b - No error 1b - Error, the CMD index in response is not CMD12/23
3 AC12CE	Auto CMD12 / 23 CRC error Occurs when detecting a CRC error in the command response. 0b - No CRC error 1b - CRC error met in Auto CMD12/23 response
2 AC12EBE	Auto CMD12 / 23 end bit error Occurs when detecting that the end field of command response is 0 which should be 1. 0b - No error 1b - End bit error generated
1 AC12TOE	Auto CMD12 / 23 timeout error Occurs if no response is returned within 64 SDCLK cycles from the end field of the command. If this field is set to1, the other error status fields (2-4) have no meaning. 0b - No error

Table continues on the next page...

Field	Function
	1b - Time out
0 AC12NE	Auto CMD12 not executed If memory multiple block data transfer is not started, due to a command error, this field is not set because it is not necessary to issue an Auto CMD12. Setting this field to 1 means uSDHC cannot issue the Auto CMD12 to stop a memory multiple block data transfer due to some error. If this field is set to 1, other error status fields (1-4) have no meaning. 0b - Executed 1b - Not executed

26.7.1.18 Host Controller Capabilities (HOST_CTRL_CAP)

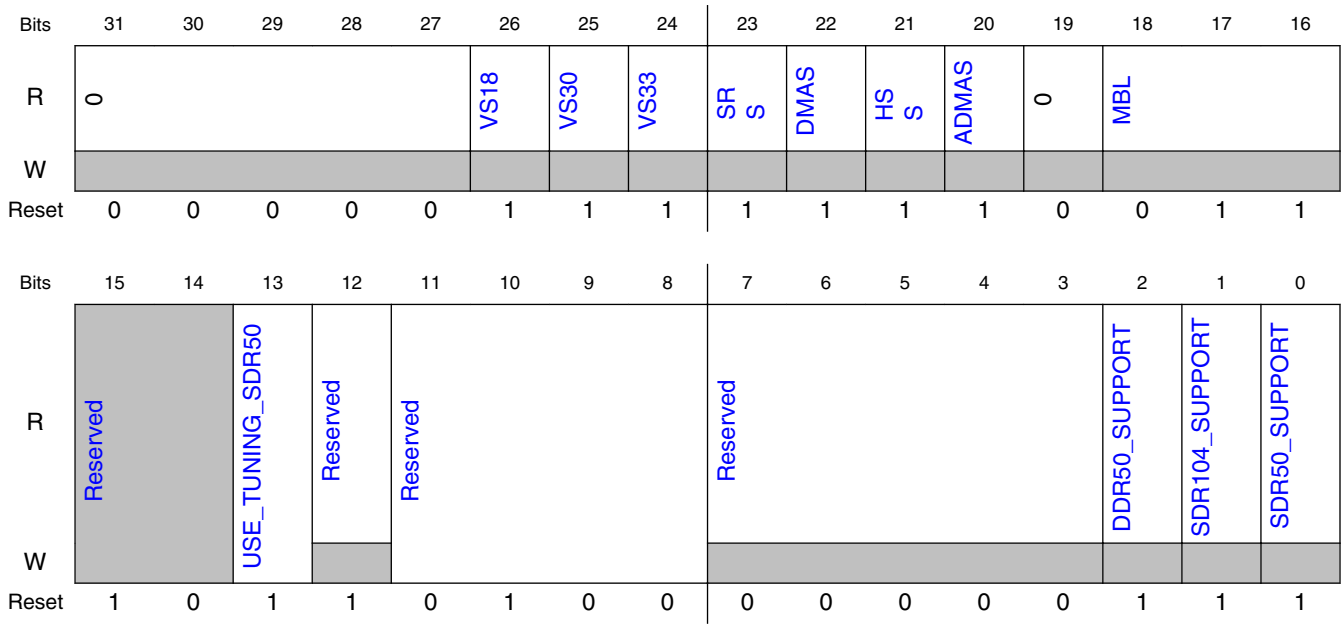
26.7.1.18.1 Offset

Register	Offset
HOST_CTRL_CAP	40h

26.7.1.18.2 Function

This register provides the host driver with information specific to uSDHC implementation. The value in this register is the power-on-reset value and does not change with a software reset.

26.7.1.18.3 Diagram



26.7.1.18.4 Fields

Field	Function
31-27 —	Reserved
26 VS18	Voltage support 1.8 V This field depends on the host system ability. 0b - 1.8 V not supported 1b - 1.8 V supported
25 VS30	Voltage support 3.0 V This field depends on the host system ability. 0b - 3.0 V not supported 1b - 3.0 V supported
24 VS33	Voltage support 3.3 V This field depends on the host system ability. 0b - 3.3 V not supported 1b - 3.3 V supported
23 SRS	Suspend / resume support This field indicates whether uSDHC supports Suspend / Resume functionality. If this field is 0, the Suspend and Resume mechanism, as well as the read wait, are not supported, and the host driver does not issue either Suspend or Resume commands. 0b - Not supported 1b - Supported
22	DMA support

Table continues on the next page...

Field	Function
DMAS	This field indicates whether uSDHC can use the internal DMA to transfer data between system memory and the data buffer directly. 0b - DMA not supported 1b - DMA supported
21 HSS	High speed support This field indicates whether uSDHC supports High Speed mode and the host system can supply a SD clock frequency from 25 MHz to 50 MHz. 0b - High speed not supported 1b - High speed supported
20 ADMAS	ADMA support This field indicates whether uSDHC supports the ADMA feature. 0b - Advanced DMA not supported 1b - Advanced DMA supported
19 —	Reserved
18-16 MBL	Max block length This field indicates the maximum block size that the host driver can read and write to the buffer in uSDHC. The buffer transfers block size without wait cycles. 000b - 512 bytes 001b - 1024 bytes 010b - 2048 bytes 011b - 4096 bytes
15-14 —	Reserved
13 USE_TUNING_SDR50	Use Tuning for SDR50 This field is used to enable tuning for SDR50 mode 0b - SDR50 does not support tuning 1b - SDR50 supports tuning
12 —	Reserved
11-8 —	Reserved
7-3 —	Reserved
2 DDR50_SUPPORT	DDR50 support This field indicates support of DDR50 mode. **NOTE:** This field is not supported in every instance. The following table includes only supported registers.

Instance	Field supported in	Field not supported in
uSDHC1	HOST_CTRL_CAP	—
uSDHC2	—	HOST_CTRL_CAP

Table continues on the next page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td>1
SDR104_SUPPORT</td><td>SDR104 support
This field indicates support of SDR104 mode.
NOTE: This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>uSDHC1</td><td>HOST_CTRL_CAP</td><td>—</td></tr>
<tr><td>uSDHC2</td><td>—</td><td>HOST_CTRL_CAP</td></tr>
</table></td></tr>
<tr><td>0
SDR50_SUPPORT</td><td>SDR50 support
This field indicates support of SDR50 mode.
NOTE: This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>uSDHC1</td><td>HOST_CTRL_CAP</td><td>—</td></tr>
<tr><td>uSDHC2</td><td>—</td><td>HOST_CTRL_CAP</td></tr>
</table></td></tr>
</table>

26.7.1.19 Watermark Level (WTMK_LVL)

26.7.1.19.1 Offset

Register	Offset
WTMK_LVL	44h

26.7.1.19.2 Function

This register indicates configurability of write and read watermark levels (FIFO threshold). Their value can range from 1 to 128 words. Both write and read burst lengths are also configurable. Their value can range from 1 to 31 words.

26.7.1.19.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0			WR_BRST_LEN					WR_WML							
W																
Reset	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0			RD_BRST_LEN					RD_WML							
W																
Reset	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0

26.7.1.19.4 Fields

Field	Function
31-29 —	Reserved
28-24 WR_BRST_LEN	Write burst length due to system restriction, the actual burst length might not exceed 16 This field indicates the number of words uSDHC writes in a single burst. The write burst length must be less than or equal to the write watermark level, and all bursts within a watermark level transfer is in back-to-back mode. On reset, this field is 8. Writing 0 to this field results in '01000' (that is, it is not able to clear this field).
23-16 WR_WML	Write watermark level This field indicates the number of words used as the watermark level (FIFO threshold) in a DMA write operation. Also, the number of words as a sequence of write bursts in back-to-back mode. The maximum legal value for the write watermark level is 128.
15-13 —	Reserved
12-8 RD_BRST_LEN	Read burst length due to system restriction, the actual burst length might not exceed 16 This field indicates the number of words uSDHC reads in a single burst. The read burst length must be less than or equal to the read watermark level, and all bursts within a watermark level transfer is in back-to-back mode. On reset, this field is 8. Writing 0 to this field results in '01000' (that is, it is not able to clear this field).
7-0 RD_WML	Read watermark level This field indicates the number of words used as the watermark level (FIFO threshold) in a DMA read operation. Also, the number of words as a sequence of read bursts in back-to-back mode. The maximum legal value for the read water mark level is 128.

26.7.1.20 Mixer Control (MIX_CTRL)

26.7.1.20.1 Offset

Register	Offset
MIX_CTRL	48h

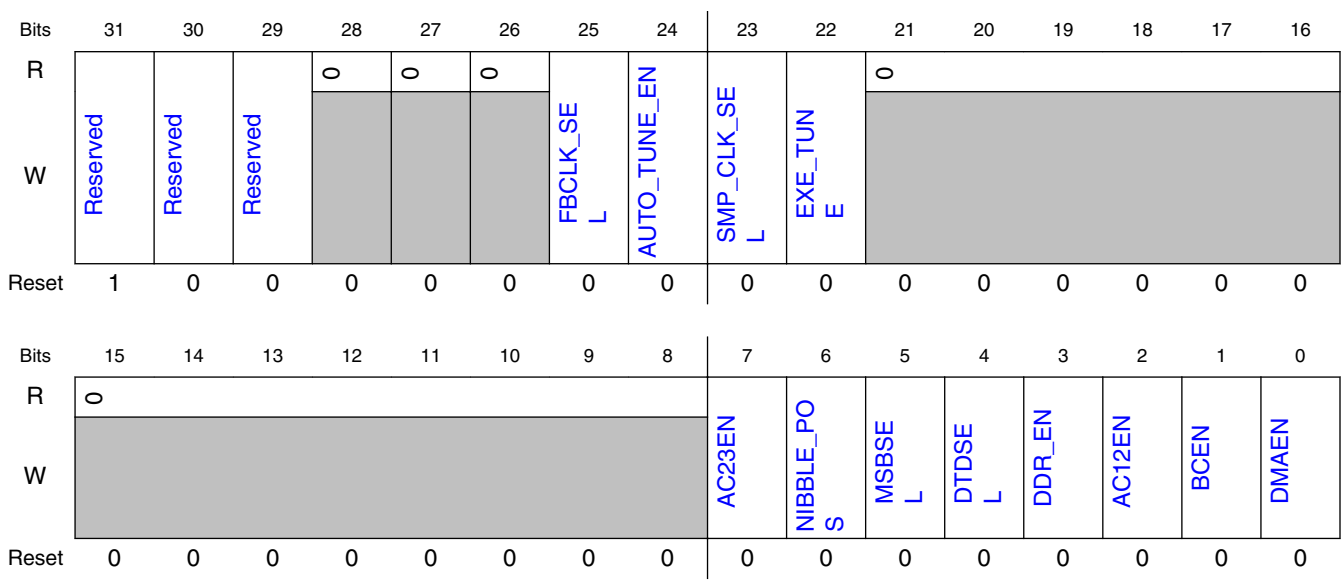
26.7.1.20.2 Function

This register is used to DMA and data transfer. To prevent data loss, the software should check if data transfer is active before writing this register. These fields are DPSEL, MBSEL, DTDSEL, AC12EN, BCEN, and DMAEN.

Table 26-13. Transfer type register setting for various transfer types

Multi/single block select	Block count enable	Block count	Function
0	Do not care	Do not care	Single transfer
1	0	Do not care	Infinite transfer
1	1	Positive number	Multiple transfer
1	1	Zero	No data transfer

26.7.1.20.3 Diagram



26.7.1.20.4 Fields

Field	Function
31	Reserved

Table continues on the next page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td>—</td><td>Always write as 1.</td></tr>
<tr><td>30
—</td><td>Reserved
Always write as 0.</td></tr>
<tr><td>29
—</td><td>Reserved
Always write as 0.</td></tr>
<tr><td>28
—</td><td>Reserved</td></tr>
<tr><td>27
—</td><td>Reserved</td></tr>
<tr><td>26
—</td><td>Reserved</td></tr>
<tr><td>25
FBCLK_SEL</td><td>Feedback clock source selection (Only used for SD3.0, SDR104 mode and EMMC HS200 mode)
NOTE: This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>uSDHC1</td><td>MIX_CTRL</td><td>—</td></tr>
<tr><td>uSDHC2</td><td>—</td><td>MIX_CTRL</td></tr>
</table>
0b - Feedback clock comes from the loopback CLK
1b - Feedback clock comes from the ipp_card_clk_out</td></tr>
<tr><td>24
AUTO_TUNE_EN</td><td>Auto tuning enable (Only used for SD3.0, SDR104 mode and and EMMC HS200 mode)
NOTE: This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>uSDHC1</td><td>MIX_CTRL</td><td>—</td></tr>
<tr><td>uSDHC2</td><td>—</td><td>MIX_CTRL</td></tr>
</table>
0b - Disable auto tuning
1b - Enable auto tuning</td></tr>
<tr><td>23
SMP_CLK_SEL</td><td>Clock selection
When TUNING_CTRL[STD_TUNING_EN] is 0, this field is used to select Tuned clock or Fixed clock to sample data / cmd (Only used for SD3.0, SDR104 mode and EMMC HS200 mode)
NOTE: This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>uSDHC1</td><td>MIX_CTRL</td><td>—</td></tr>
<tr><td>uSDHC2</td><td>—</td><td>MIX_CTRL</td></tr>
</table></td></tr>
</table>

Table continues on the next page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>0b - Fixed clock is used to sample data / cmd
1b - Tuned clock is used to sample data / cmd</td></tr>
<tr><td>22
EXE_TUNE</td><td>Execute tuning: (Only used for SD3.0, SDR104 mode and EMMC HS200 mode)
When TUNING_CTRL[STD_TUNING_EN] is 0, this field is set to 1 to indicate the host driver is starting tuning procedure. Tuning procedure is aborted by writing 0.
NOTE: This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>uSDHC1</td><td>MIX_CTRL</td><td>—</td></tr>
<tr><td>uSDHC2</td><td>—</td><td>MIX_CTRL</td></tr>
</table>
0b - Not tuned or tuning completed
1b - Execute tuning</td></tr>
<tr><td>21-8
—</td><td>Reserved</td></tr>
<tr><td>7
AC23EN</td><td>Auto CMD23 enable
When this field is set to 1, the host controller issues a CMD23 automatically before issuing a command specified in the Command Register.</td></tr>
<tr><td>6
NIBBLE_POS</td><td>Nibble position indication
This field indicates the nibble position in the DDR 4-bit mode. 0- the sequence is 'odd high nibble -> even high nibble -> odd low nibble -> even low nibble'; 1- the sequence is 'odd high nibble -> odd low nibble -> even high nibble -> even low nibble'.</td></tr>
<tr><td>5
MSBSEL</td><td>Multi / Single block select
This field enables multiple block DATA line data transfers. For any other commands, this field can be set to 0. If this field is 0, it is not necessary to set the Block Count register. See Command Transfer Type (CMD_XFR_TYP).
0b - Single block
1b - Multiple blocks</td></tr>
<tr><td>4
DTDSEL</td><td>Data transfer direction select
This field defines the direction of DATA line data transfers. The field is set to 1 by the host driver to transfer data from the SD card to uSDHC and is set to 0 for all other commands.
If a CMD with the response data size cannot be divided by 64 bytes, is given without a STOP CMD, and is followed by a software reset, then you must write a 0 to DTDSEL before inserting a software reset. This is necessary to avoid any issue during the SD card reinitialization. Read multiple blocks will not be affected as it is required to have a STOP CMD.
0b - Write (Host to card)
1b - Read (Card to host)</td></tr>
<tr><td>3
DDR_EN</td><td>Dual data rate mode selection</td></tr>
<tr><td>2
AC12EN</td><td>Auto CMD12 enable</td></tr>
</table>

Table continues on the next page...

Field	Function
	Multiple block transfers for memory require a CMD12 to stop the transaction. When this field is set to 1, uSDHC issues a CMD12 automatically when the last block transfer has completed. The host driver is not set this field to issue commands that do not require CMD12 to stop a multiple block data transfer. In particular, secure commands defined in File Security Specification (see reference list) do not require CMD12. In single block transfer, uSDHC ignores this field no matter it is set or not. 0b - Disable 1b - Enable
1 BCEN	Block count enable This field is used to enable the Block Count register, which is only relevant for multiple block transfers. When this field is 0, the internal counter for block is disabled, which is useful in executing an infinite transfer. 0b - Disable 1b - Enable
0 DMAEN	DMA enable This field enables DMA functionality. If this field is set to 1, a DMA operation begins when the host driver sets the DPSEL field of this register. Whether the simple DMA or the advanced DMA is active depends on the DMA Select field of the Protocol Control register. 0b - Disable 1b - Enable

26.7.1.21 Force Event (FORCE_EVENT)

26.7.1.21.1 Offset

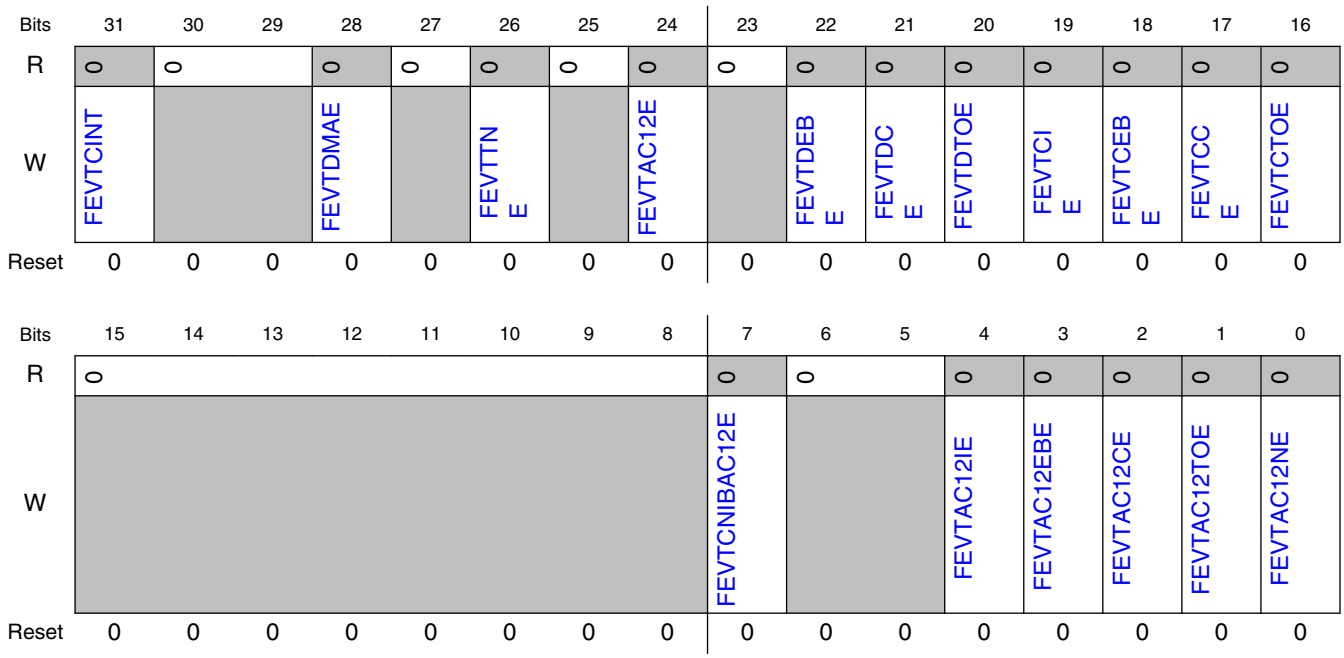
Register	Offset
FORCE_EVENT	50h

26.7.1.21.2 Function

This register is not a physically implemented register. Rather, it is an address at which the Interrupt Status register can be written if the corresponding field of the Interrupt Status Enable Register is set. This register is a write only register and writing 0 to it has no effect. Writing 1 to this register sets the corresponding field of Interrupt Status register. A read from this register always results in 0's. to change the corresponding status fields in the Interrupt Status register, make sure to set IPGEN field in System Control Register so that peripheral clock is always active.

Forcing a card interrupt generates a short pulse on the DATA1 line, and the driver might treat this interrupt as a normal interrupt. The interrupt service routine might skip polling the card interrupt factor as the interrupt is self cleared.

26.7.1.21.3 Diagram



26.7.1.21.4 Fields

Field	Function
31 FEVTCINT	Force event card interrupt Writing 1 to this field generates a short low-level pulse on the internal DATA1 line, as if a self-clearing interrupt was received from the external card. If enabled, the CINT field is set and the interrupt service routine might treat this interrupt as a normal interrupt from the external card.
30-29 —	Reserved
28 FEVTDMAE	Force event DMA error Forces the DMAE field of Interrupt Status register to be set
27 —	Reserved
26 FEVTTNE	Force tuning error Forces the TNE field of Interrupt Status register to be set
25 —	Reserved
24 FEVTAC12E	Force event Auto Command 12 error Forces the AC12E field of Interrupt Status register to be set
23 —	Reserved

Table continues on the next page...

Field	Function
22 FEVTDEBE	Force event data end bit error Forces the DEBE field of Interrupt Status register to be set
21 FEVTDCE	Force event data CRC error Forces the DCE field of Interrupt Status register to be set
20 FEVTDTOE	Force event data time out error Force the DTOE field of Interrupt Status register to be set
19 FEVTCIE	Force event command index error Forces the CCE field of Interrupt Status register to be set
18 FEVTCEBE	Force event command end bit error Forces the CEBE field of Interrupt Status register to be set
17 FEVTCCE	Force event command CRC error Forces the CCE field of Interrupt Status register to be set
16 FEVTCTOE	Force event command time out error Forces the CTOE field of Interrupt Status register to be set
15-8 —	Reserved
7 FEVTCNIBAC12E	Force event command not executed by Auto Command 12 error Forces the CNIBAC12E field in the Auto Command12 Error Status register to be set
6-5 —	Reserved
4 FEVTAC12IE	Force event Auto Command 12 index error Forces the AC12IE field in the Auto Command12 Error Status register to be set
3 FEVTAC12EBE	Force event Auto Command 12 end bit error Forces the AC12EBE field in the Auto Command12 Error Status register to be set
2 FEVTAC12CE	Force event auto command 12 CRC error Forces the AC12CE field in the Auto Command12 Error Status register to be set
1 FEVTAC12TOE	Force event auto command 12 time out error Forces the AC12TOE field in the Auto Command12 Error Status register to be set
0 FEVTAC12NE	Force event auto command 12 not executed Forces the AC12NE field in the Auto Command12 Error Status register to be set

26.7.1.22 ADMA Error Status (ADMA_ERR_STATUS)

26.7.1.22.1 Offset

Register	Offset
ADMA_ERR_STATUS	54h

26.7.1.22.2 Function

When an ADMA Error Interrupt has occurred, the ADMA error sates field in this register holds the ADMA state and the ADMA System Address register holds the address around the error descriptor.

For recovering from this error, the host driver requires the ADMA state to identify the error descriptor address as follows:

- ST_STOP: Previous location set in the ADMA System Address register is the error descriptor address.
- ST_FDS: Current location set in the ADMA System Address register is the error descriptor address.
- ST_CADR: This state is never set because it only increments the descriptor pointer and does not generate an ADMA error.
- ST_TFR: Previous location set in the ADMA System Address register is the error descriptor address.

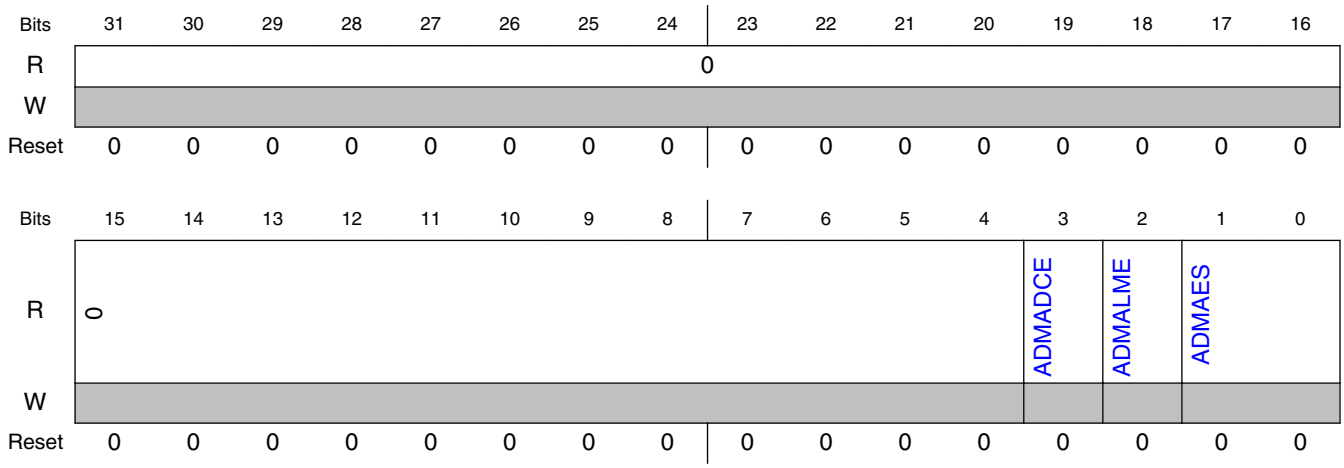
In case of a write operation, the host driver should use the ACMD22 to get the number of the written block, rather than using this information, because unwritten data might exist in the host controller.

The host controller generates the ADMA Error Interrupt when it detects invalid descriptor data (Valid=0) in the ST_FDS state. The host driver can distinguish this error by reading the Valid field of the error descriptor.

Table 26-14. ADMA error state coding

D01-D00	ADMA error state (when error has occurred)	Contents of ADMA System Address register
00	ST_STOP (Stop DMA)	Holds the address of the next executable descriptor command
01	ST_FDS (Fetch descriptor)	Holds the valid descriptor address
10	ST_CADR (Change address)	No ADMA error is generated
11	ST_TFR (Transfer data)	Holds the address of the next executable descriptor command

26.7.1.22.3 Diagram



26.7.1.22.4 Fields

Field	Function
31-4 —	Reserved
3 ADMADCE	ADMA descriptor error This error occurs when invalid descriptor fetched by ADMA. 0b - No error 1b - Error
2 ADMALME	ADMA length mismatch error This error occurs in the following two cases: • While the block count enable is being set, the total data length specified by the descriptor table is different from that specified by the block count and block length. • Total data length cannot be divided by the block length. 0b - No error 1b - Error
1-0 ADMAES	ADMA error state (when ADMA error is occurred) This field indicates the state of the ADMA when an error has occurred during an ADMA data transfer. See ADMA Error Status (ADMA_ERR_STATUS) for more details.

26.7.1.23 ADMA System Address (ADMA_SYS_ADDR)

26.7.1.23.1 Offset

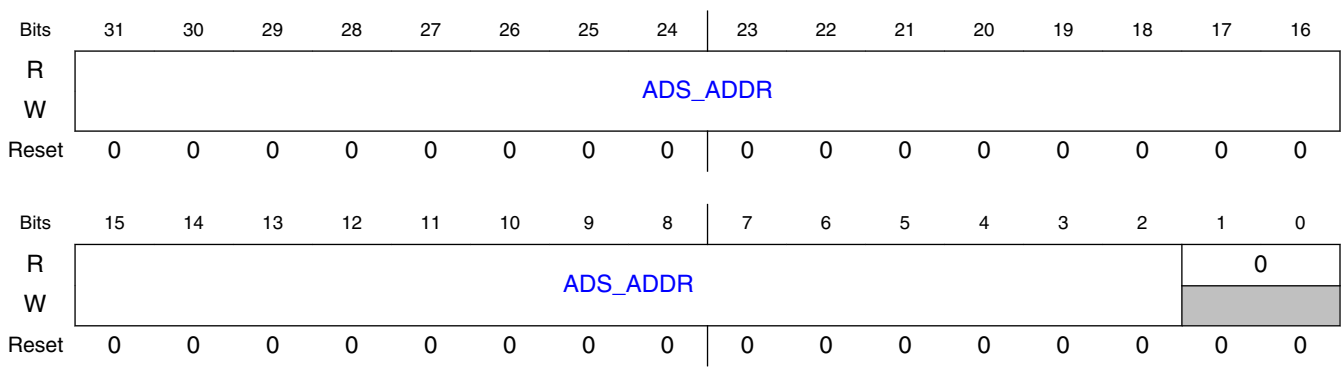
Register	Offset
ADMA_SYS_ADDR	58h

26.7.1.23.2 Function

This register holds the word address of the executing command in the Descriptor table. At the start of ADMA, the host driver sets the start address of the Descriptor table. The ADMA engine increments this register address whenever fetching a Descriptor command. When the ADMA is stopped at the Block Gap, this register indicates the address of the next executable Descriptor command. When the ADMA Error Interrupt is generated, this register holds the valid Descriptor address depending on the ADMA state. The lower 2 bits of this register is tied to '0' so the ADMA address is always word aligned.

Because this register supports dynamic address reflecting, when TC field is set, it automatically alters the value of internal address counter, so the software cannot change this register when TC field is set. Such restriction is also listed in Software restrictions.

26.7.1.23.3 Diagram



26.7.1.23.4 Fields

Field	Function
31-2 ADS_ADDR	ADMA system address This field contains the physical system memory address used for ADMA transfers.
1-0 —	Reserved

26.7.1.24 DLL (Delay Line) Control (DLL_CTRL)

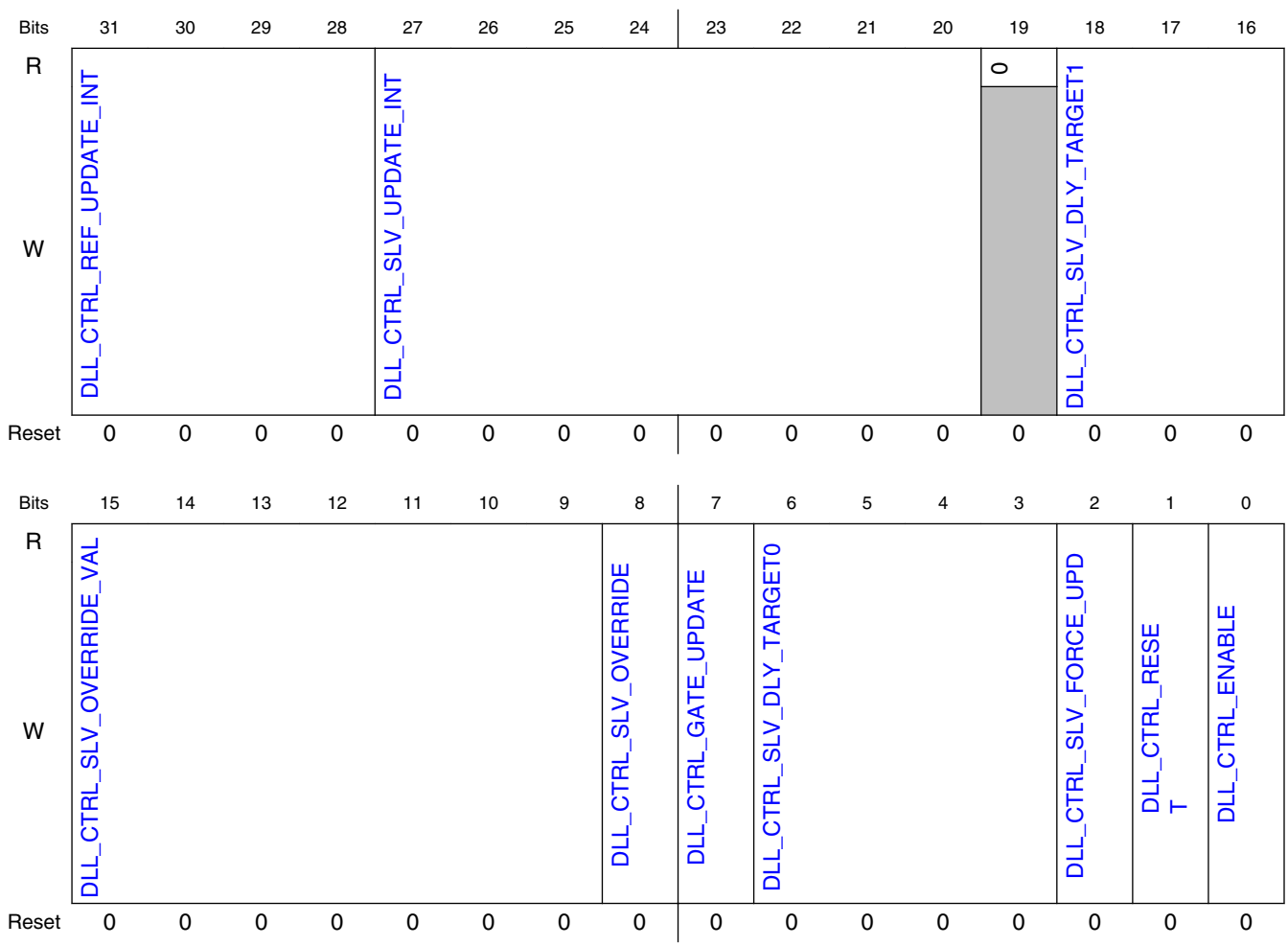
26.7.1.24.1 Offset

Register	Offset
DLL_CTRL	60h

26.7.1.24.2 Function

This register contains control fields for DLL.

26.7.1.24.3 Diagram



26.7.1.24.4 Fields

Field	Function
31-28 DLL_CTRL_REF_UPDATE_INT	DLL control loop update interval The interval cycle is (2 + REF_UPDATE_INT) * REF_CLOCK. By default, the DLL control loop updates every two REF_CLOCK cycles. It should be noted that increasing the reference delay-line update interval reduces the ability of the DLL to adjust to fast changes in conditions that might effect the delay (such as voltage and temperature)
27-20 DLL_CTRL_SLV_UPDATE_INT	Slave delay line update interval If default 0 is used, it means 256 cycles of REF_CLOCK. A value of 0x0f results in 15 cycles and so on. Note that software can always cause an update of the slave-delay line using the SLV_FORCE_UPDATE register. Note that the slave delay line also updates automatically when the reference DLL transitions to a locked state (from an un-locked state).
19 —	Reserved
18-16 DLL_CTRL_SLV_DLY_TARGET1	DLL slave delay target1 See DLL_CTRL_SLV_DLY_TARGET0 below.
15-9 DLL_CTRL_SLV_OVERRIDE_VAL	DLL slave override val When SLV_OVERRIDE = 1, this field is used to select 1 of 128 physical taps manually. A value of 0 selects tap 1, and a value of 0x7f selects tap 128.
8 DLL_CTRL_SLV_OVERRIDE	DLL slave override Set this field to 1 to Enable manual override for slave delay chain using SLV_OVERRIDE_VAL; to set 0 to disable manual override. This feature does not require the DLL to be enabled using the ENABLE field. In fact to reduce power, if SLV_OVERRIDE is used, it is recommended to disable the DLL with ENABLE = 0
7 DLL_CTRL_GATE_UPDATE	DLL gate update Set this field to 1 to prevent the DLL from updating (because when clock_in exists, glitches might appear during DLL updates). This field might be used by software if such a condition occurs. Clear the bit to 0 to allow the DLL to update automatically.
6-3 DLL_CTRL_SLV_DLY_TARGET0	DLL slave delay target0 The delay target for uSDHC loopback read clock can be programmed in 1/16th increments of an ref_clock half-period. The delay is (({DLL_CTRL_SLV_DLY_TARGET1,DLL_CTRL_SLV_DLY_TARGET0} +1)* REF_CLOCK / 2) / 16 So the input read-clock can be delayed relative input data from (REF_CLOCK / 2) / 16 to REF_CLOCK * 4.
2 DLL_CTRL_SLV_FORCE_UPD	DLL slave delay line Setting this field to 1, forces the slave delay line to update to the DLL calibrated value immediately. The slave delay line updates automatically based on the SLV_UPDATE_INT interval or when a DLL lock condition is sensed. Subsequent forcing of the slave-line update can only occur if SLV_FORCE_UP is set back to 0 and then asserted again (edge triggered). Be sure to use it when uSDHC is idle. This function might not work when uSDHC is working on data / cmd / response.
1 DLL_CTRL_RESET	DLL reset Setting this field to 1 force a reset on DLL. This causes the DLL to lose lock and re-calibrate to detect an REF_CLOCK half period phase shift. This signal is used by the DLL as edge-sensitive, so to create a subsequent reset, RESET must be taken low and then asserted again.
0	DLL and delay chain

Field	Function
DLL_CTRL_ENABLE	Set this field to 1 to enable the DLL and delay chain; otherwise; set to 0 to bypasses DLL. Note that using the slave delay line override feature with SLV_OVERRIDE and SLV_OVERRIDE VAL, the DLL does not need to be enabled.

26.7.1.25 DLL Status (DLL_STATUS)

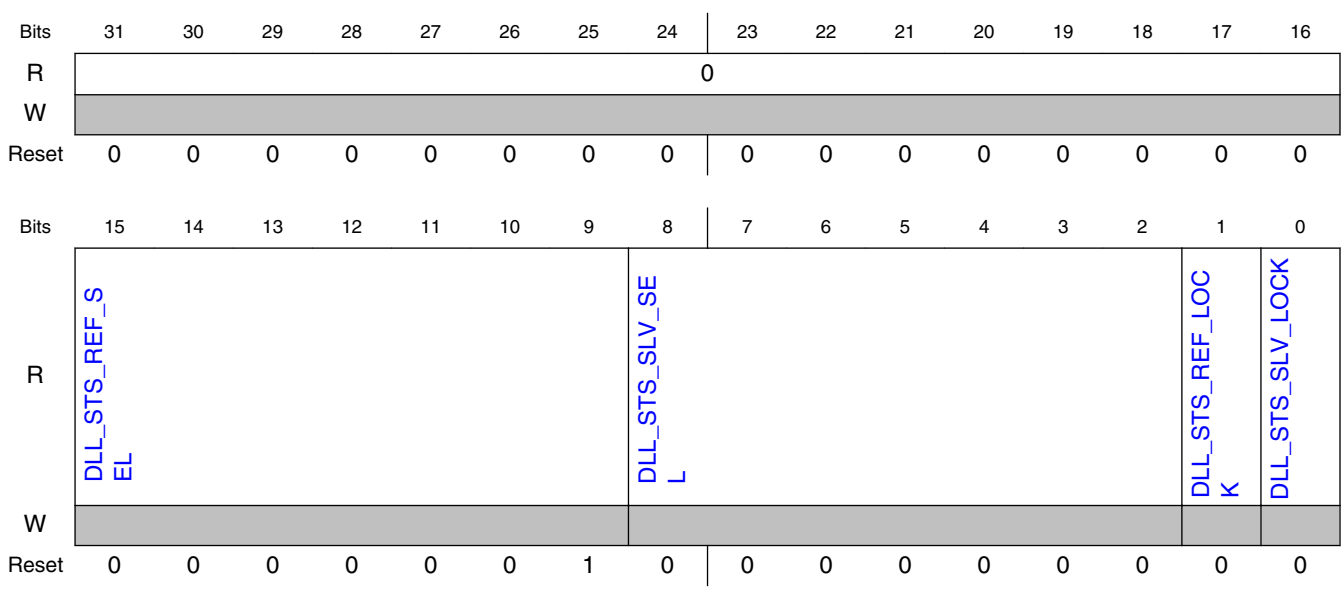
26.7.1.25.1 Offset

Register	Offset
DLL_STATUS	64h

26.7.1.25.2 Function

This register contains the DLL status information. All fields are read only and reads the same as the power-reset value.

26.7.1.25.3 Diagram



26.7.1.25.4 Fields

Field	Function
31-16 —	Reserved
15-9 DLL_STS_REF_SEL	Reference delay line select taps This is encoded by 7 fields for 127 taps.
8-2 DLL_STS_SLV_SEL	Slave delay line select status This is the instant value generated from reference chain. Because the reference chain can only be updated when REF_CLOCK is detected, this value should be the right value to be updated when the reference is locked.
1 DLL_STS_REF_LOCK	Reference DLL lock status This signifies that the DLL has detected and locked to a half-phase ref_clock shift, allowing the slave delay-line to perform programmed clock delays
0 DLL_STS_SLV_LOCK	Slave delay-line lock status This signifies that a valid calibration has been set to the slave-delay line and that the slave-delay line is implementing the programmed delay value

26.7.1.26 CLK Tuning Control and Status (CLK_TUNE_CTRL_STATUS)

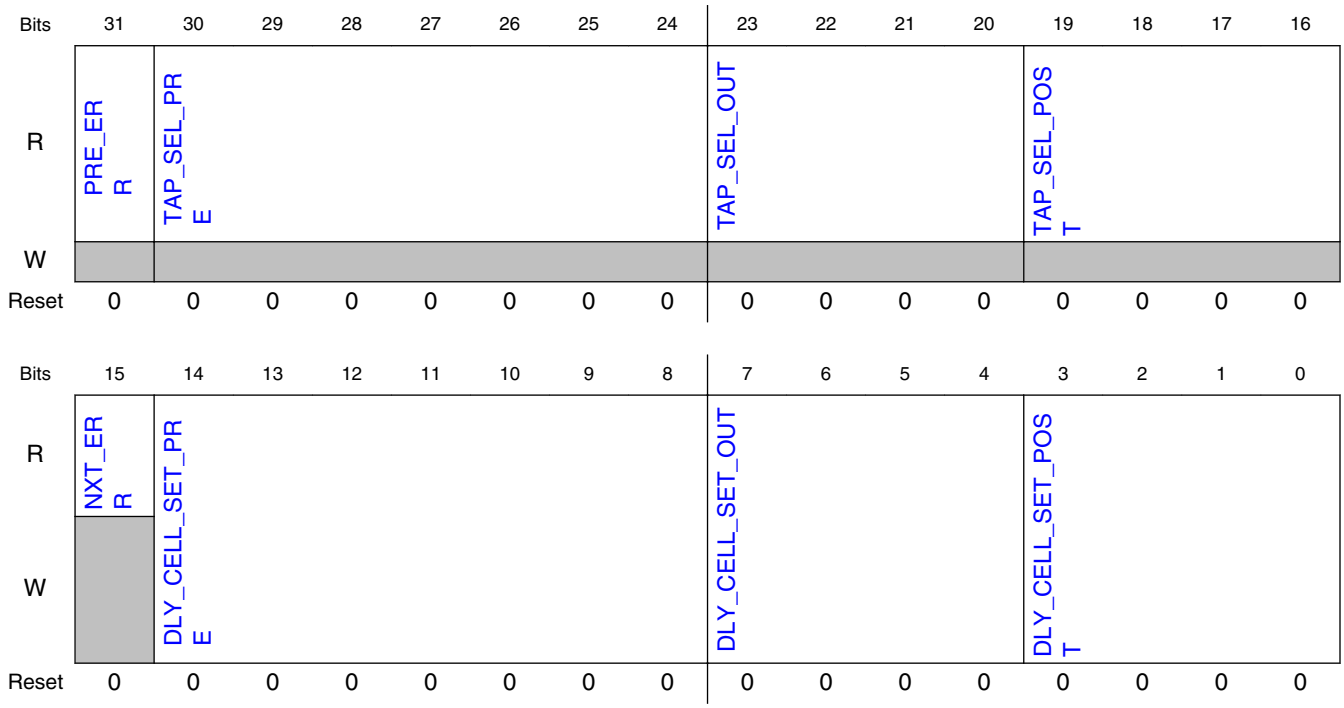
26.7.1.26.1 Offset

Register	Offset
CLK_TUNE_CTRL_STATUS	68h

26.7.1.26.2 Function

This register contains the Clock Tuning Control status information. All fields are read only and reads the same as the power-reset value. This register is added to support SD3.0 UHS-I SDR104 mode and EMMC HS200 mode.

26.7.1.26.3 Diagram



26.7.1.26.4 Fields

Field	Function
31 PRE_ERR	PRE error PRE error which means the number of delay cells added on the feedback clock is too small. It is valid only when SMP_CLK_SEL of Mix control register (bit23 of 0x48) is enabled.
30-24 TAP_SEL_PRE	TAP_SEL_PRE Reflects the number of delay cells added on the feedback clock between the feedback clock and CLK_PRE. When AUTO_TUNE_EN (bit24 of 0x48) is disabled, TAP_SEL_PRE is always equal to DLY_CELL_SET_PRE. When AUTO_TUNE_EN (bit24 of 0x48) is enabled, TAP_SEL_PRE is updated automatically according to the status of the auto tuning circuit to adjust the sample clock phase.
23-20 TAP_SEL_OUT	Delay cells added on the feedback clock between CLK_PRE and CLK_OUT Reflects the number of delay cells added on the feedback clock between CLK_PRE and CLK_OUT.
19-16 TAP_SEL_POST	Delay cells added on the feedback clock between CLK_OUT and CLK_POST Reflects the number of delay cells added on the feedback clock between CLK_OUT and CLK_POST.
15 NXT_ERR	NXT error NXT error which means the number of delay cells added on the feedback clock is too large. It's valid only when SMP_CLK_SEL of Mix control register (bit23 of 0x48) is enabled.
14-8	delay cells on the feedback clock between the feedback clock and CLK_PRE

Table continues on the next page...

Field	Function
DLY_CELL_SET_PRE	Set the number of delay cells on the feedback clock between the feedback clock and CLK_PRE.
7-4 DLY_CELL_SET_OUT	Delay cells on the feedback clock between CLK_PRE and CLK_OUT Set the number of delay cells on the feedback clock between CLK_PRE and CLK_OUT.
3-0 DLY_CELL_SET_POST	Delay cells on the feedback clock between CLK_OUT and CLK_POST Set the number of delay cells on the feedback clock between CLK_OUT and CLK_POST.

26.7.1.27 Vendor Specific Register (VEND_SPEC)

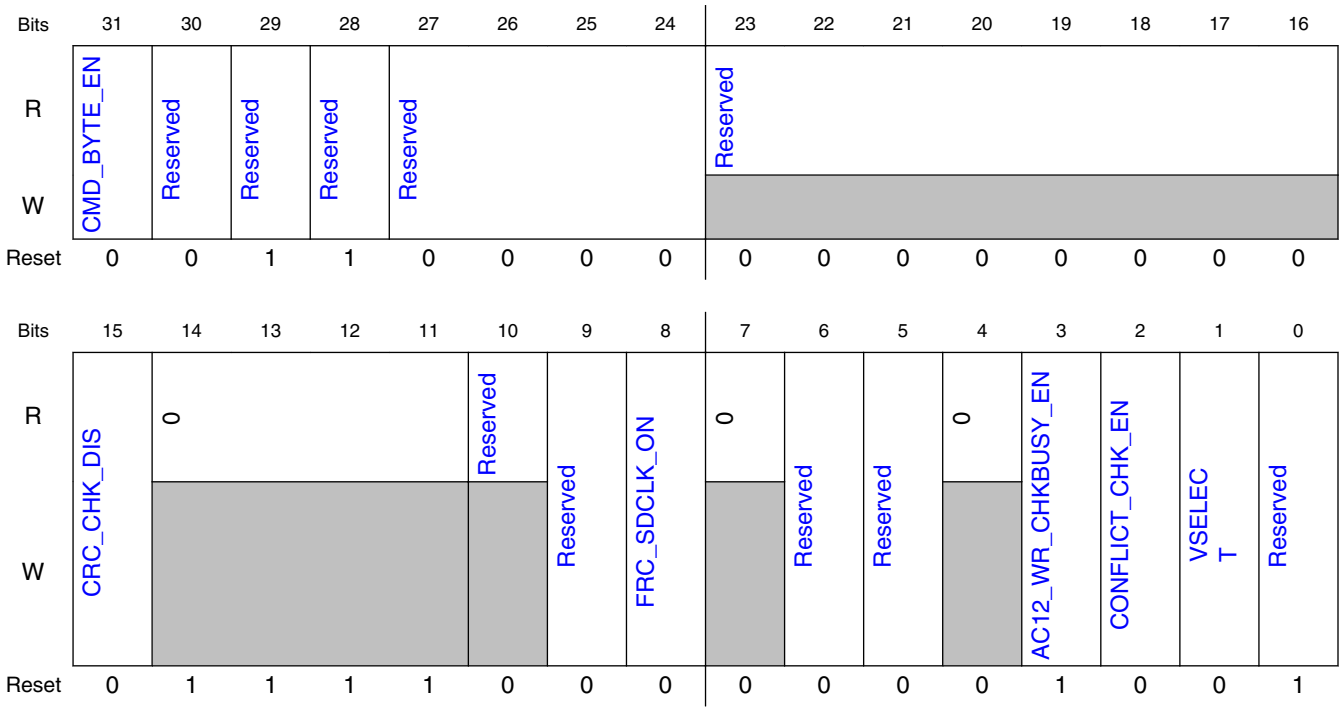
26.7.1.27.1 Offset

Register	Offset
VEND_SPEC	C0h

26.7.1.27.2 Function

This register contains the vendor specific control/status register.

26.7.1.27.3 Diagram



26.7.1.27.4 Fields

Field	Function
31 CMD_BYTE_EN	Byte access This field controls the byte access. 0b - Disable 1b - Enable
30 —	Reserved Always write as 0.
29 —	Reserved Always write as 1
28 —	Reserved Always write as 0.
27-24 —	Reserved Always write as 4'b0000.
23-16 —	Reserved Always write as 8'h00.
15 CRC_CHK_DIS	CRC Check Disable 0b - Check CRC16 for every read data packet and check CRC fields for every write data packet 1b - Ignore CRC16 check for every read data packet and ignore CRC fields check for every write data packet

Table continues on the next page...

Field	Function
14-11 —	Reserved Reserved
10 —	Reserved Always write as 0.
9 —	Reserved Always write as 0.
8 FRC_SDCLK_ON	Force CLK Force CLK output active Do not set this bit to 1 unless it is necessary. Also, make sure that this bit is cleared when uSDHC's clock is about to be changed (frequency change, clock source change, or delay chain tuning). 0b - CLK active or inactive is fully controlled by the hardware. 1b - Force CLK active
7 —	Reserved
6 —	Reserved
5 —	Reserved
4 —	Reserved
3 AC12_WR_CHKBUSY_EN	Check busy enable Check busy enable after auto CMD12 for write data packet 0b - Do not check busy after auto CMD12 for write data packet 1b - Check busy after auto CMD12 for write data packet
2 CONFLICT_CHK_EN	Conflict check enable It is not implemented in uSDHC IP. 0b - Conflict check disable 1b - Conflict check enable
1 VSELECT	Voltage selection Change the value of output signal VSELECT to control the voltage on pads for external card. There must be a control circuit out of uSDHC to change the voltage on pads. 0b - Change the voltage to high voltage range, around 3.0 V 1b - Change the voltage to low voltage range, around 1.8 V
0 —	Reserved Always write as 0.

26.7.1.28 MMC Boot (MMC_BOOT)

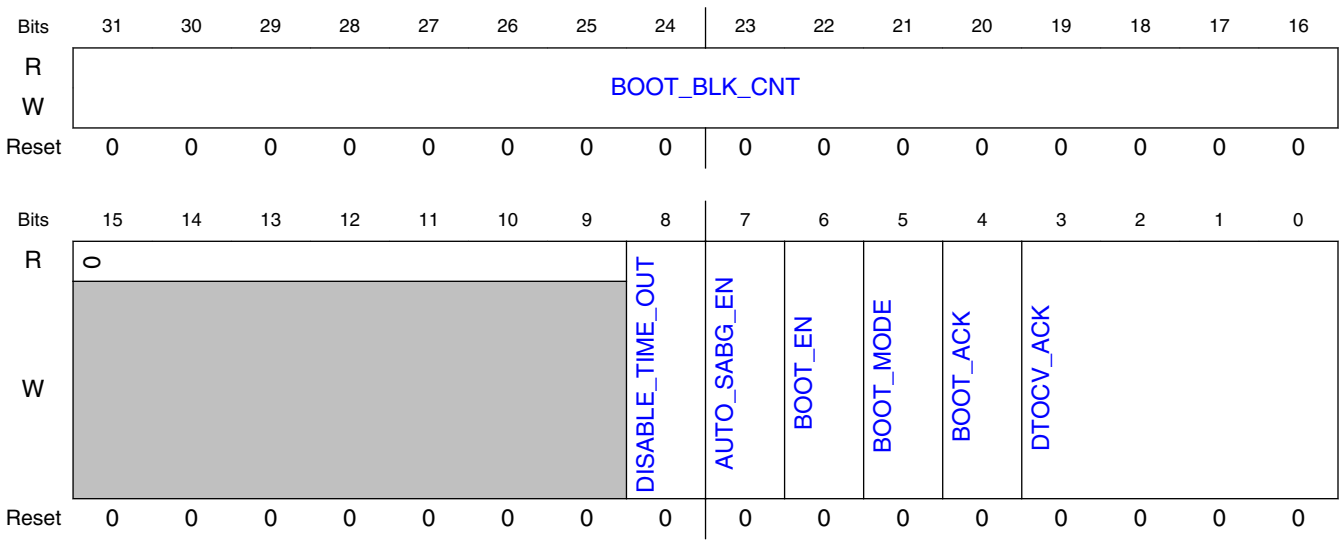
26.7.1.28.1 Offset

Register	Offset
MMC_BOOT	C4h

26.7.1.28.2 Function

This register contains the MMC Fast Boot control register.

26.7.1.28.3 Diagram



26.7.1.28.4 Fields

Field	Function
31-16 BOOT_BLK_CNT	Stop At Block Gap value of automatic mode The value defines the Stop At Block Gap value of automatic mode. When received, card block cnt is equal to (BLK_CNT - BOOT_BLK_CNT) and AUTO_SABG_EN is 1, then Stop At Block Gap. Here, BLK_CNT is defined in the Block Attributes Register, field 31 - 16 of 0x04.
15-9 —	Reserved
8 DISABLE_TIME_OUT	Time out **NOTE:** When this field is set, there is no timeout check no matter whether BOOT_EN is set or not. 0b - Enable time out 1b - Disable time out
7	Auto stop at block gap

Table continues on the next page...

Field	Function
AUTO_SABG_EN	During boot, enable auto stop at block gap function. This function is triggered, and host stops at block gap when received card block cnt is equal to (BLK_CNT - BOOT_BLK_CNT).
6 BOOT_EN	Boot enable Boot mode enable 0b - Fast boot disable 1b - Fast boot enable
5 BOOT_MODE	Boot mode Boot mode select 0b - Normal boot 1b - Alternative boot
4 BOOT_ACK	BOOT ACK Boot ACK mode select 0b - No ack 1b - Ack
3-0 DTOCV_ACK	Boot ACK time out Boot ACK time out counter value. 0000b - SDCLK x 2^14 0001b - SDCLK x 2^15 0010b - SDCLK x 2^16 0011b - SDCLK x 2^17 0100b - SDCLK x 2^18 0101b - SDCLK x 2^19 0110b - SDCLK x 2^20 0111b - SDCLK x 2^21 1110b - SDCLK x 2^28 1111b - SDCLK x 2^29

26.7.1.29 Vendor Specific 2 Register (VEND_SPEC2)

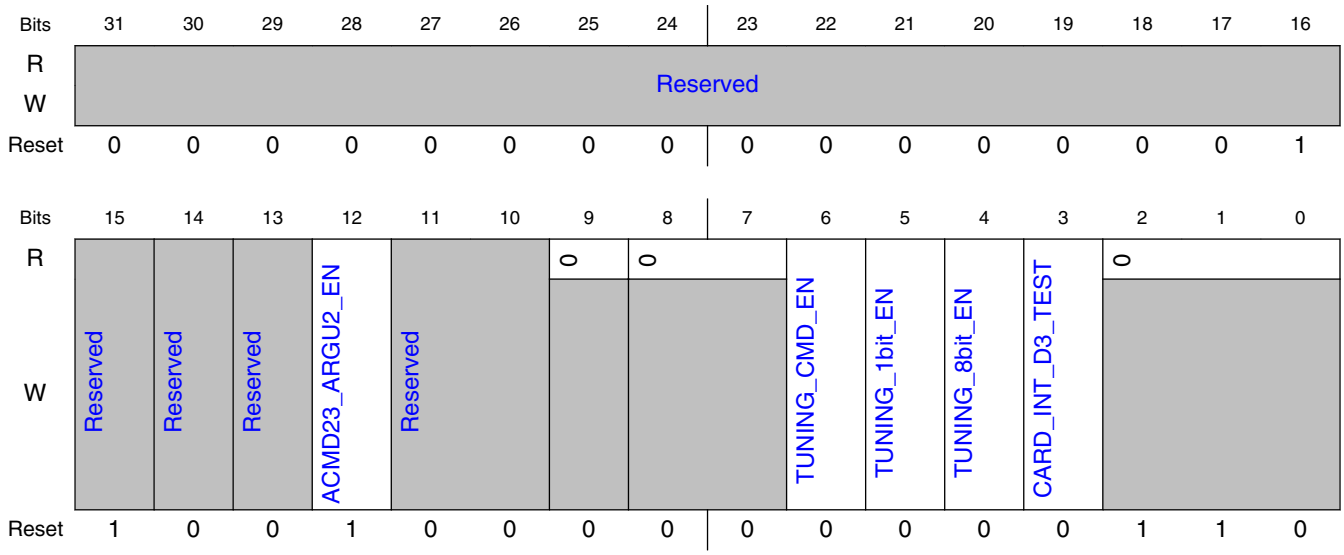
26.7.1.29.1 Offset

Register	Offset
VEND_SPEC2	C8h

26.7.1.29.2 Function

This register contains the vendor specific control 2 register.

26.7.1.29.3 Diagram



26.7.1.29.4 Fields

Field	Function
31-15 —	Reserved
14 —	Reserved
13 —	Reserved
12 ACMD23_ARGU2_EN	Argument2 register enable for ACMD23 0b - Disable 1b - Argument2 register enable for ACMD23 sharing with SDMA system address register. Default is enabled.
11-10 —	Reserved
9 —	Reserved
8-7 —	Reserved
6 TUNING_CMD_EN	Tuning command enable Enable the auto tuning circuit to check the CMD line. 0b - Auto tuning circuit does not check the CMD line. 1b - Auto tuning circuit checks the CMD line.
5	Tuning 1bit enable Enable the auto tuning circuit to check the DATA0 only. It is used with the TUNING_8bit_EN together.

Table continues on the next page...

Field	Function
TUNING_1bit_EN	
4 TUNING_8bit_EN	Tuning 8bit enable Enable the auto tuning circuit to check the DATA[7:0]. It is used with the TUNING_1bit_EN together. 0b00 - Tuning circuit only checks the DATA[3:0] 0b01 - Tuning circuit only checks the DATA0 0b10 - Tuning circuit checks the whole DATA[7:0] 0b11 - Invalid **NOTE:** The format of these two fields are [TUNNING_8bit_EN:TUNNING_1bit_EN].
3 CARD_INT_D3_TEST	Card interrupt detection test This field is used only for debugging. 0b - Check the card interrupt only when DATA3 is high. 1b - Check the card interrupt by ignoring the status of DATA3.
2-0 —	Reserved

26.7.1.30 Tuning Control (TUNING_CTRL)

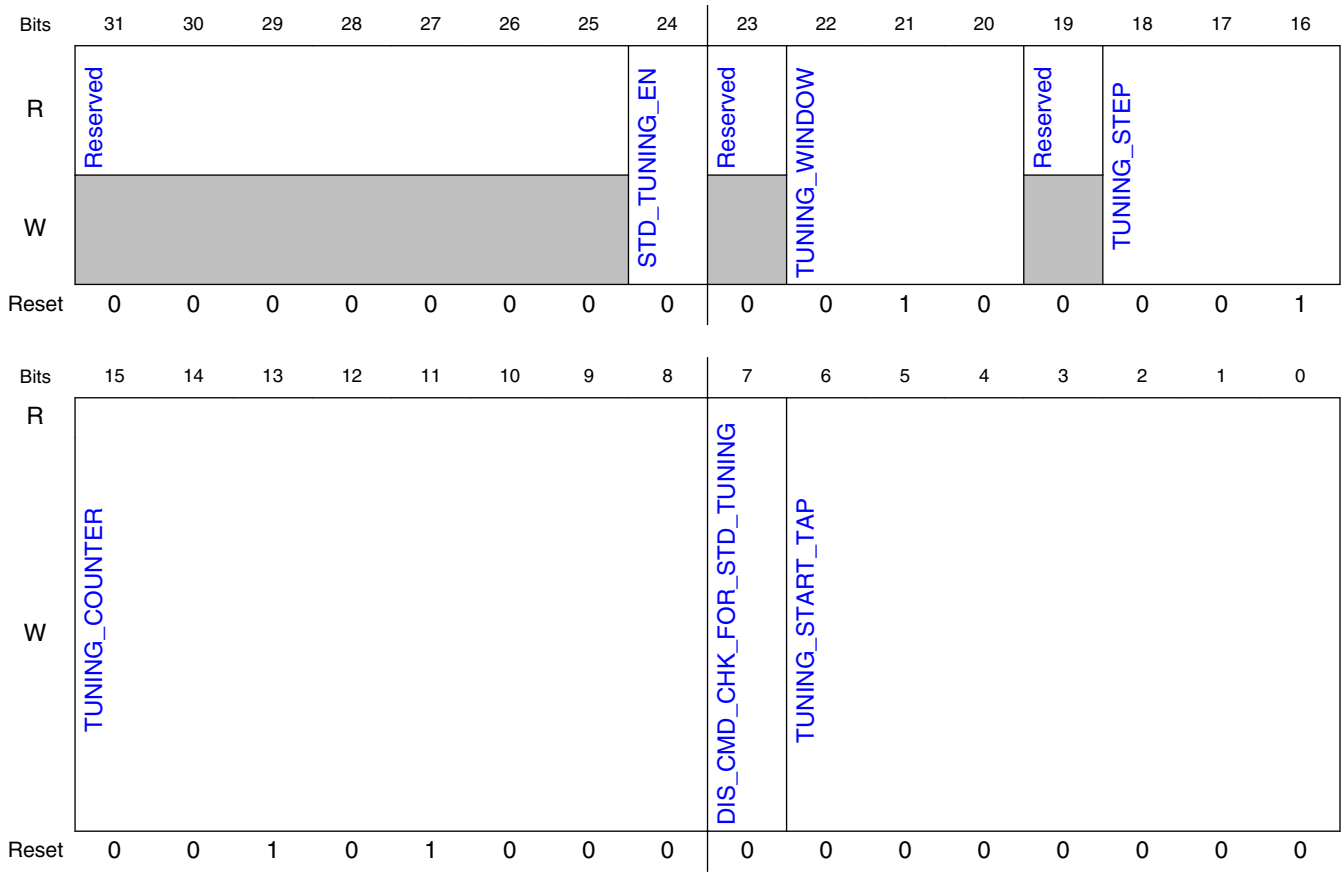
26.7.1.30.1 Offset

Register	Offset
TUNING_CTRL	CCh

26.7.1.30.2 Function

The register contains configuration of tuning circuit.

26.7.1.30.3 Diagram



26.7.1.30.4 Fields

Field	Function
31-25 —	Reserved
24 STD_TUNING_EN	Standard tuning circuit and procedure enable This field is used to enable standard tuning circuit and procedure.
23 —	Reserved
22-20 TUNING_WINDOW	Data window Select data window value for auto tuning
19 —	Reserved
18-16 TUNING_STEP	TUNING_STEP The increasing delay cell steps in tuning procedure.

Table continues on the next page...

Field	Function
15-8 TUNING_COUNTER	Tuning counter The MAX repeat CMD19 times in tuning procedure.
7 DIS_CMD_CHK_FOR_STD_TUNING	Disable command check for standard tuning Writing 1 to this field disables command check (command CRC, CMD end bit error, and CMD index error or CMD timeout error) for standard tuning flow after each tuning command is sent.
6-0 TUNING_START_TAP	Tuning start The start delay cell point when send first CMD19 in tuning procedure.

Chapter 27
FlexSPI Controller

27.1 Chip-specific FlexSPI information

Table 27-1. Reference links to related information

Topic	Related module(s)	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
Interrupts, DMA Events and XBAR Assignments	-	Interrupts, DMA Events and XBAR Assignments

NOTE

The soft reset bit (MCR_SWRESET) should be set, to enable the inverted clock, after setting the SCKBDIFFOPT bit.

NOTE

Octal mode is supported by combining SIOA[3:0] and SIOB[3:0], on this device.

27.1.1 Master ID allocation

This table summarizes the master IDs for all system master modules:

Table 27-2. Master IDs

Module	Master ID (MSTRID)
Core Platform	000b

Table continues on the next page...

Table 27-2. Master IDs (continued)

Module	Master ID (MSTRID)
eDMA	001b
DCP	010b
All others	011b

27.2 Overview

Flexible Serial Peripheral Interface (FlexSPI) host controller supports two SPI channels and up to 4 external devices. Each channel supports Single/Dual/Quad/Octal mode data transfer (1/2/4/8 bidirectional data lines).

NOTE

FlexSPI configuration is dependent on the chip configuration. Please refer to the system-level sections for boot and pinmux for chip specific information regarding the modes and number of devices supported.

27.2.1 Block diagram

The block diagram of FlexSPI is indicated in following figure:

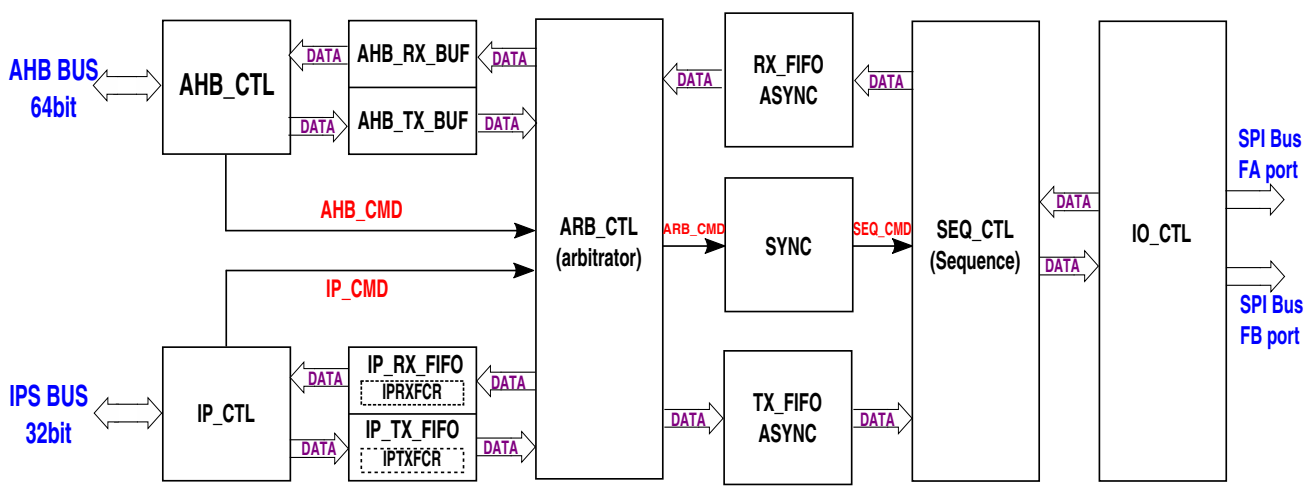


Figure 27-1. FlexSPI block diagram

27.2.2 Features

FlexSPI block supports following features:

- Flexible sequence engine (LUT table) to support various vendor devices
  - Serial NOR Flash or other device with similar SPI protocol as Serial NOR Flash
  - FPGA device
- Flash access mode
  - Single/Dual/Quad/Octal mode
  - SDR/DDR mode
  - Individual/Parallel mode
- Support sampling clock mode:
  - Internal dummy read strobe loopbacked internally
  - Internal dummy read strobe loopbacked from pad
  - Flash provided read strobe
- Memory mapped read/write access by AHB Bus
  - AHB RX Buffer implemented to reduce read latency. See the chip-specific section for AHB buffer size.
  - 16 AHB masters supported with priority for read access
  - 4 flexible and configurable buffers in AHB RX Buffer
  - AHB TX Buffer implemented to buffer all write data from one AHB burst. See the chip-specific section for AHB buffer size.
  - All AHB masters share this AHB TX Buffer. No AHB master number limitation for Write Access.
- Software triggered Flash read/write access by IP Bus
  - IP RX FIFO implemented to buffer all read data from External device. FIFO size: 128 Bytes
  - IP TX FIFO implemented to buffer all Write data to External device. FIFO size: 128 Bytes
  - DMA support to fill IP TX FIFO
  - DMA support to read IP RX FIFO
  - SCLK stopped when reading flash data and IP RX FIFO is full
  - SCLK stopped when writing flash data and IP TX FIFO is empty

27.3 Functional description

The following sections describe functional details of the FlexSPI module.

27.3.1 Flash Connection

There are two FlexSPI interface ports (A port and B port). Each port supports 2 flash devices by providing 2 chip select outputs.

NOTE

FlexSPI configuration is dependent on the chip configuration. See the chip-specific FlexSPI information regarding the number of devices supported.

The connection diagram with 4 devices is as following:

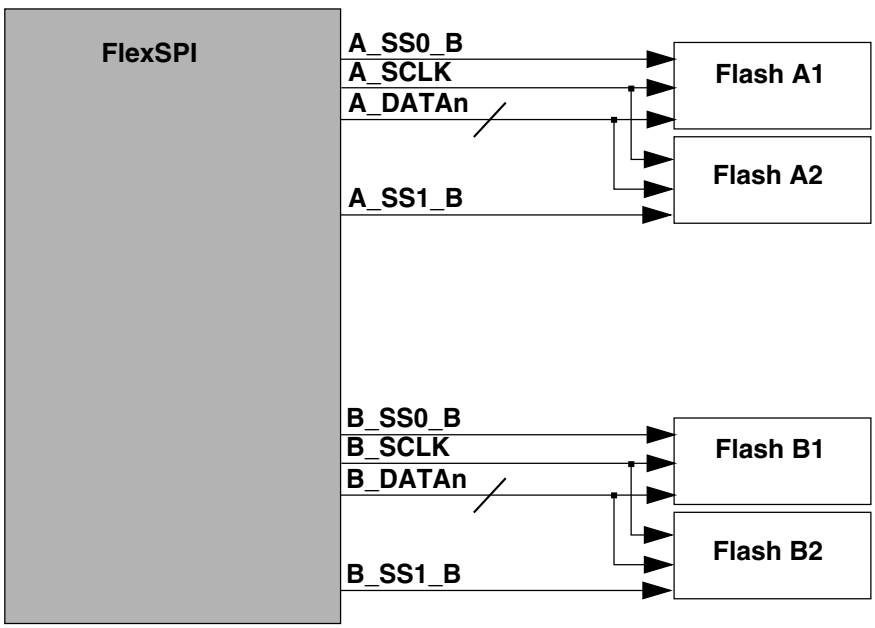


Figure 27-2. Flash connection diagram with four devices

NOTE

- Flash A1 and A2 could be two flash chip package or two flash die on the same package. There is no difference to FlexSPI. Same for Flash B1 and B2.
- Flash A1 and B1 could be accessed in parallel, using parallel mode. FlexSPI will merge/split the flash read/program data automatically. Same for A2 and B2.
- In parallel mode, A1 and A2 could not be accessed at the same time. Same for B1 and B2.
- In individual mode, A1, A2, B1 and B2 could not be accessed at the same time. But these four device could be accessed separately.

There is a combination mode to provide octal flash support by combining A port (A_DATA[3:0]) and B port (B_DATA[3:0]) together. Octal mode can also be supported by normal mode but the I/O width of A port and B port should be 8. The connection diagram for the combination mode is as following:

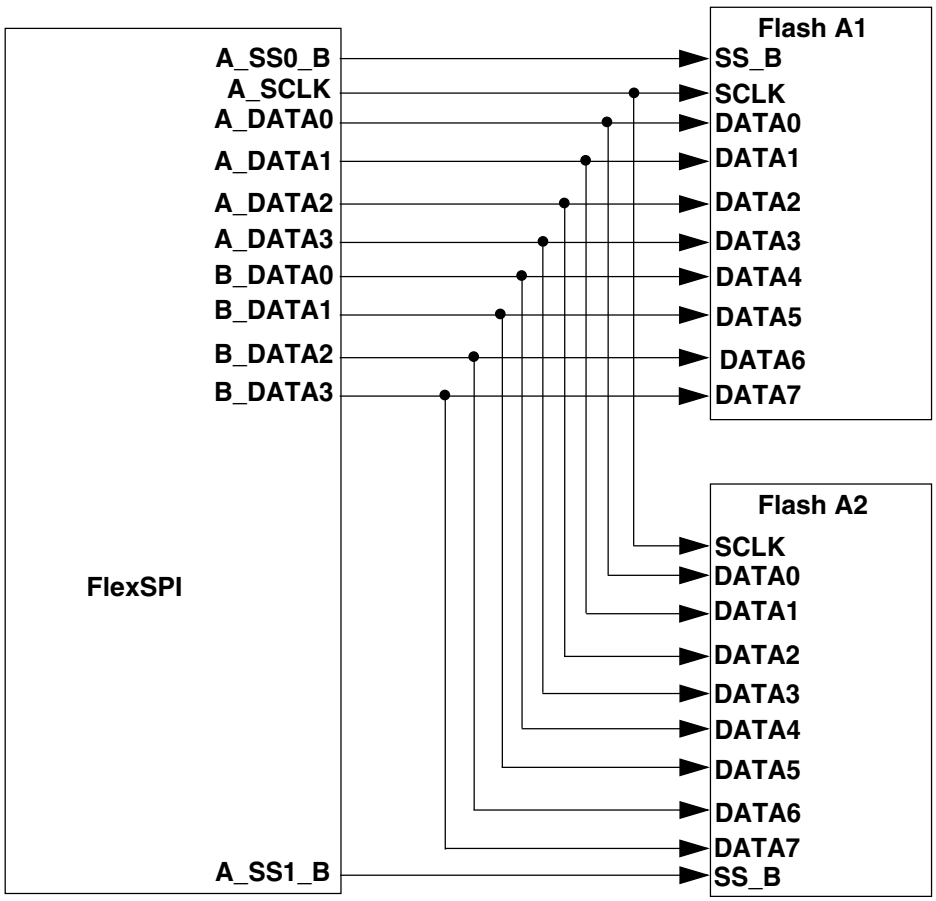


Figure 27-3. Flash connection diagram with combination mode

27.3.2 Flash Access mode

This section describes flash access mode.

27.3.2.1 SPI clock mode

FlexSPI supports only SPI clock mode 0: Clock polarity (CPOL)=0 and Clock Phase (CPHA)=0. SCLK will stay at logic low state when SPI bus is idle.

27.3.2.2 Flash Individual mode and Parallel mode

In individual mode, Flash read/write data is received/transmit on port A or port B.

In parallel mode, Flash read/write data is received/transmit on port A and B port parallely. FlexSPI will merge/split the flash read/program data automatically. Please note that only read/program data is merged/split (READ/WRITE instruction). For other instructions (such as Command/Address/Mode/Data size), same command code/address/mode bits/data size information will be transmit to port A and port B device. For more detail, please refer to Instruction execution on SPI interface.

Individual and Parallel mode is determined statically by register field IPCR1[IPAREN] (for IP command) or AHBCR[APAREN] (for AHB command).

27.3.2.3 SDR mode and DDR mode

In SDR (Single Data transfer Rate) mode, Flash receives data on SCLK rise edge and transmit data on SCLK fall edge.

In DDR (Dual Data transfer Rate) mode, Flash receives data on both SCLK rise and fall edges and transmit data on both SCLK rise and fall edges.

SDR and DDR mode is determined by instruction (opcode) in LUT sequence dynamically. There is no static configuration register field setting for SDR and DDR mode. For more details about input and output timing, please refer to FlexSPI Input Timing and FlexSPI Output Timing.

27.3.2.4 Single, Dual, Quad, and Octal mode

In Single mode, flash transmit/receive data on 1 Data pin (DATA0 for transmitting, DATA1 for receiving).

In Dual mode, flash transmit/receive data on 2 Data pin (DATA0~DATA1 for both transmitting and receiving).

In Quad mode, flash transmit/receive data on 4 Data pin (DATA0~DATA3 for both transmitting and receiving).

In Octal mode, flash transmit/receive data on 8 Data pin (DATA0~DATA7 for both transmitting and receiving).

Single, Dual, Quad and Octal mode is determined by instruction (num_pads) in LUT sequence dynamically. There is no static configuration register field setting for Single, Dual, Quad and Octal mode.

27.3.3 Operation Modes

This section provides information about the modes FlexSPI could be used.

- Module Disable mode

  This mode is low power mode in FlexSPI module.

  In module disable mode, AHB clock and serial clock domain will be gated off internally, but IPS Bus clock is not gated off. Control and status register read/write access is available except LUT/IP RX FIFO/IP TX FIFO read/write access. Serial Flash Memory access is also not available.

  This mode is entered by setting MCR0[MDIS], and exited by clearing MCR0[MDIS].

- Doze mode

  This mode is low power mode in SOC system.

  When the system requires FlexSPI to enter doze mode and MCR0[DOZEEN] is set, the FlexSPI will wait for all transactions to complete (STS0[ARBIDLE]=0x1) and enter doze mode. In doze mode, the AHB clock and serial clock domain will be gated off internally, but IPS Bus clock is not gated off. Control and status register read/write access is available except LUT/IP RX FIFO/IP TX FIFO read/write access. Serial Flash Memory access is also not available.

  This mode is entered by system request, and exited by deasserting this system request.

- Stop mode

  This mode is low power mode in SOC system.

  When the system requires FlexSPI to enter stop mode, FlexSPI will wait for all transactions to complete (STS0[ARBIDLE]=0x1) and return ACK handshake to system. After ACK handshake returned, IP will gate off the AHB clock and serial clock domain internally, the system can gate the AHB Bus clock, IPS Bus clock and serial clock in system level.

  This mode is entered by system stop mode request. When all command sequences finish, FlexSPI enters stop mode and returns ACK handshake. This mode is exited by deasserting system stop mode request, the ACK handshake is also deasserted immediately.

- Normal mode

In normal mode, all clocks are not gated internally. Normal register access and serial flash memory access is available.

27.3.4 Flash memory map

Flash memory map in individual and parallel mode is as following:

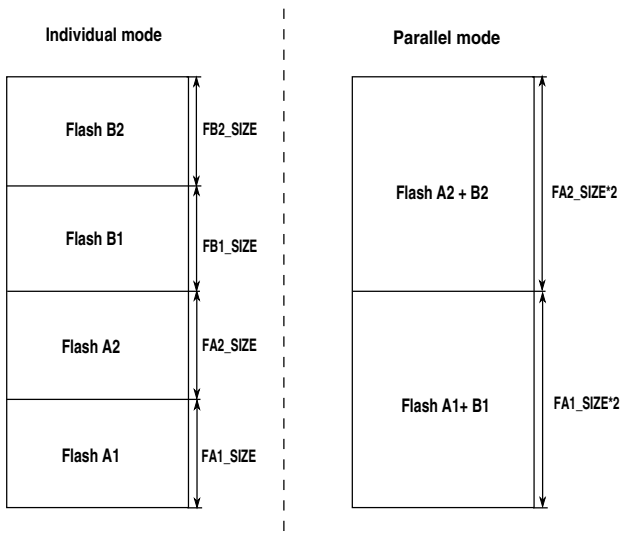


Figure 27-4. Flash memory map in individual and parallel mode

Flash memory map in individual mode:

- Flash A1 address range: 0x00000000 ~ FA1_SIZE
- Flash A2 address range: FA1_SIZE ~ (FA1_SIZE + FA2_SIZE)
- Flash B1 address range: (FA1_SIZE + FA2_SIZE) ~ (FA1_SIZE + FA2_SIZE + FB1_SIZE)
- Flash B2 address range: FA1_SIZE + FA2_SIZE + FB1_SIZE) ~ (FA1_SIZE + FA2_SIZE + FB1_SIZE + FB2_SIZE)

Flash memory map in parallel mode:

- Flash A1+B1 address range: 0x00000000 ~ FA1_SIZE*2
- Flash A2+B2 address range: FA1_SIZE*2 ~ (FA1_SIZE*2 + FA2_SIZE*2)

NOTE

- When MCR2[SAMEDEVICEEN] is set to 0x1, FA1_SIZE/FA2_SIZE/FB1_SIZE/FB2_SIZE = FLSHA1CR0[FLSHSZ] * 1KByte
- When MCR2[SAMEDEVICEEN] is set to 0x0, FA1_SIZE = FLSHA1CR0[FLSHSZ] * 1KByte; FA2_SIZE = FLSHA2CR0[FLSHSZ] * 1KByte; FB1_SIZE =

FLSHB1CR0[FLSHSZ] * 1KByte; FB2_SIZE = FLSHB2CR0[FLSHSZ] * 1KByte
- Flash B1/B2 size setting are ignored in parallel mode (FLSHB1CR0[FLSHSZ], FLSHB2CR0[FLSHSZ]). To support parallel mode application, Flash B1 should be same device as A1 and Flash B2 should be same device as A2.

27.3.5 Flash address sent to Device

Flash access start address is determined by AHB address (AHB command) or IPCR0[SFAR] register (IP command). Refer Flash access by AHB Command and Flash access by IP Command for more details.

For AHB command, FlexSPI controller will remove flash base address automatically when sending flash address to devices. For IP command, the address in IPCR0[SFAR] should be the flash device's address without base address. Flash address is sent to devices in two part: Row Address and Column Address. For flash devices not supportting Column address, please set register field FLSHxCR1[CAS] to 0. Then all flash address bits will be sent to Flash device as Row address. For flash device supporting word-addressable feature, the last bit of flash address is not needed because flash is read/programmed in terms of 2 bytes. For parallel mode, Flash A1/B1 (or A2/B2) is accessed parallelly, so the flash address sent to flash device should be divided by 2. Following table indicates the relationship of Row/Column Address and flash address (FA)

Table 27-3. Flash Row/Column Address

Parallel mode	Word-addressable	Row Address	Column Address	Comment
0	0	FA[31:CAS]	FA[CAS-1:0]	There is no limitation on FA and data size alignment.
0	1	FA[31:CAS+1]	FA[CAS:1]	FA and data size should be 2 byte aligned.
1	0	FA[31:CAS+1]	FA[CAS:1]	FA and data size should be 2 byte aligned.
1	1	FA[31:CAS+2]	FA[CAS+1:2]	FA and data size should be 4 byte aligned.

NOTE

- FA is the flash address with flash base address removed.
- If the Row/Column Flash Address bit number to be sent to Flash device defined in Instructions is more than valid

Row/Column Flash Address bit number, high posistion bits will be supplimented with zero. Refer Programmable Sequence Engine for more details about Row/Column Address instruction.

When parallel mode enabled or word-addressable flash used, there is limitation on flash start address and data size. This requirement could be meet by aligning AHB bus access address (For AHB command) or IP command address IPCR0[SFAR] (For IP command) in software. There are two ways to avoid these limitation in specified case.

1. For **AHB Read Command** only:

   When AHBCR[READADDROPT] and AHBCR[PREFETCHEN] are both set to 1, FlexSPI will guarantee flash access start address and data size are 64 bit aligned by hardware.

   When AHBCR[READADDROPT] is set to 1, FlexSPI will fetch redundant data to guarantee flash start address aligned with 8 bytes. When prefetch enabled (AHBCR[PREFETCHEN] is set to 1), flash read data size is determined by AHB RX Buffer size which is 64 bit aligned.

2. For **AHB Write Command and Individual mode** only:

   By default, FlexSPI will guarantee flash write access start address and data size are 16 bit aligned by using DQS as write mask.

   This feature is not applied in parallel mode and should be used only if external device supports write mask feature.

27.3.6 Look Up Table

The LUT (Look Up Table) is an internal memory to preserve a number of pre-programmed sequences. Each sequence consists of up to 8 instructions which are executed sequentially. When a flash access is triggered by an IP command or an AHB command, FlexSPI controller will fetch the sequence from LUT according to sequence index/number and execute it to generate a valid flash transaction on SPI interface.

Following figure indicates the structure of LUT, sequence and instruction.

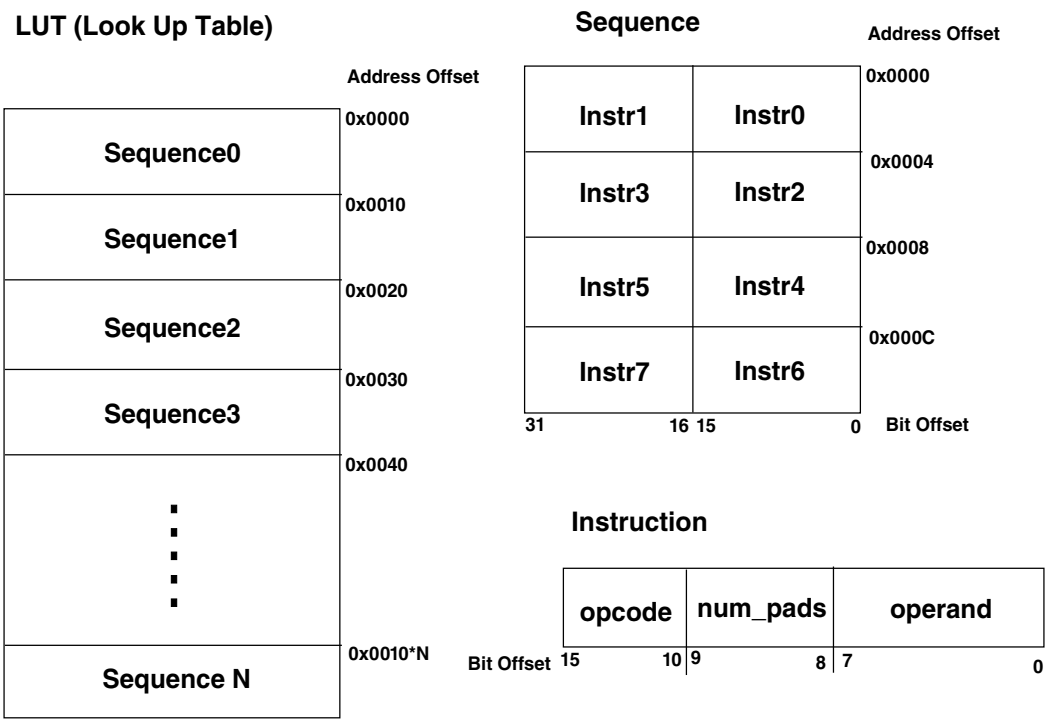


Figure 27-5. LUT and sequence structure

NOTE

If the instruction number needed is less than 8 for a flash transaction, STOP instruction should be programmed for the unneeded instructions (instruction code 8'h00).

For IP command and AHB write command, FlexSPI controller always executes from instruction pointer 0. For AHB read command, FlexSPI controller executes from a saved instruction start pointer. FlexSPI controller saves the instruction start pointers separately for each flash device. All these saved instruction pointer are zero before JMP_ON_CS instruction is executed,. When JMP_ON_CS instruction is executed, the operand in JMP_ON_CS instruction will be saved as instruction start pointer. Refer XIP Enhanced Mode for more details.

The reset value of LUT is unknown because it is implemented as internal memory. LUT should be programmed according to the device connected on board. In order to protect its contents during a code runover, the LUT could be locked/unlocked to avoid change by mistake after programmed. The key for locking or unlocking the LUT is **0x5AF05AF0**. The process for locking and unlocking the LUT is as follows:

Locking the LUT

1. Write the key **(0x5AF05AF0)** in to the LUT Key Register LUT Key Register (LUTKEY).

2. Write 1b1 to LUTCR[LOCK] and 1b0 to LUTCR[UNLOCK] fields of the LUT Control Register, immediately after the above KEY register writing. LUT is not successfully locked if there is any other register write access to FlexSPI between these two write accesses.

Unlocking the LUT

1. Write the key **(0x5AF05AF0)** in to the LUT Key Register LUT Key Register (LUTKEY).
2. Write 1b0 to LUTCR[LOCK] and 1b1 to LUTCR[UNLOCK] fields of the LUT Control Register, immediately after the above KEY register writing. LUT is not successfully unlocked if there is any other register write access to FlexSPI between these two write accesses.

The lock status of the LUT can be read from register field LUTCR[LOCK] and LUTCR[UNLOCK].

27.3.7 Programmable Sequence Engine

FlexSPI controller implements a programmable sequence engine that executes the sequence from LUT. FlexSPI controller executes the instructions sequentially and generates flash transaction on the SPI interface accordingly. The following table is a complete list of the supported instructions.

Table 27-4. Instruction set

Name	Opcode	Num_pads	Action on SPI interface	Transmit Data	Bits/Bytes/Cycle Number
CMD_SDR/ CMD_DDR	6'h01/ 6'h21	2'h0 - one pad (Single mode) 2'h1 - two pad (Dual mode) 2'h2 - four pad (Quad mode) 2'h3 - eight pad (Octal mode)	Transmit Command code to Flash	Command code: Operand[7:0]	Bit number: 8
RADDR_SDR/ RADDR_DDR	6'h02/ 6'h22		Transmit Row Address to Flash	Row_Address[31:0]	Bit number: operand[7:0]
CADDR_SDR/ CADDR_DDR	6'h03/ 6'h23		Transmit Column Address to Flash	Column_Address[31:0]	Bit number: operand[7:0]
MODE1_SDR/ MODE1_DDR	6'h04/ 6'h24		Transmit Mode bits to Flash	Mode bits: Operand[0]	Bit number: 1
MODE2_SDR/ MODE2_DDR	6'h05/ 6'25			Mode bits: Operand[1:0]	Bit number: 2
MODE4_SDR/ MODE4_DDR	6'h06/ 6'h26			Mode bits: Operand[3:0]	Bit number: 4
MODE8_SDR/ MODE8_DDR	6'h07/ 6'h27			Mode bits: Operand[7:0]	Bit number: 8

Table continues on the next page...

Table 27-4. Instruction set (continued)

Name	Opcode	Num_pads	Action on SPI interface	Transmit Data	Bits/Bytes/Cycle Number
WRITE_SDR/ WRITE_DDR	6'h08/ 6'h28	DLPR register is undocumented	Transmit Programming Data to Flash	Programming Data in IP TX FIFO or AHB_TX_BUF	Byte number (data size) is determined by AHB burst size and burst type (AHB Command) or IPCR1[IDATSZ] (IP command). For more detail about flash read/ program data size, refer Flash access by AHB Command and Flash access by IP Command.
READ_SDR/ READ_DDR	6'h09/ 6'h29		Receive Read Data from Flash Read Data is put into AHB_RX_BUF or IP_RX_FIFO.	-	
LEARN_SDR/ LEARN_DDR	6'h0A/ 6'h2A		Receive Read Data or Preamble bit from Flash device FlexSPI Controller will compare the data line bits with DLPR register to determine a correct sampling clock phase.	-	Learning pattern bit number: operand[7:0] Never set operand to zero for LEARN instruction. It indicates length of data learning pattern based on bits number. For example, 8bits pattern 0x5A on each data line needs to set operand to 8.
DATSZ_SDR/ DATSZ_DDR	6'h0B/ 6'h2B		Transmit Read/ Program Data size (byte number) to Flash	Internal Logic Read/Program data size for current command sequence.	Bit number: operand[7:0] Please never set operand to zero or larger than 64 for DATSZ instruction.
DUMMY_SDR/ DUMMY_DDR	6'h0C/ 6'h2C		Leave data lines undriven by FlexSPI controller. Provide turnaround cycles from host driving to device driving. num_pads will determine the number of pads in input mode.	-	Dummy cycle number (in serial root clock): Operand[7:0] **Dummy cycle (N), described in the Flash device datasheet is in number of SCLK cycles and this number may be configurable.** **In SDR mode, SCLK cycle is same as serial root clock. The operand value should be set as N.** **In DDR mode, SCLK cycle is double the serial root clock cycle. The operand value should be set as 2*N, 2*N-1 or 2*N+1 depending on how dummy cycle defined in device datasheet. Please refer to Flash access sequence example and dummy cycle definition on device datasheet.**

Table continues on the next page...

Table 27-4. Instruction set (continued)

Name	Opcode	Num_pads	Action on SPI interface	Transmit Data	Bits/Bytes/Cycle Number
DUMMY_RWDS_SDR/ DUMMY_RWDS_DDR	6'h0D/ 6'h2D		This instruction is similar as DUMMY_SDR/ DUMMY_DDR instruction. But the dummy cycle number is different.	-	For read command, dummy cycle number (in serial root clock): (operand[7:0]*4-1) if RWDS (DQS pin) is high; (operand[7:0]*2-1) if RWDS (DQS pin) is low; For write command, dummy cycle number (in serial root clock): (operand[7:0]*4-2) if RWDS (DQS pin) is high; (operand[7:0]*2-2) if RWDS (DQS pin) is low;
JMP_ON_CS	6'h1F	Num_pads setting will be ignored. Always set num_pads to 2'h0.	Stop execution, deassert CS and save operand[7:0] as the instruction start pointer for next sequence. Normally this instruction is used to support Execute-In-Place enhance mode. Refer XIP Enhanced Mode for more details. This instruction is only allowed for AHB read command. If this instruction is used in IP command or AHB write command, there will be interrupt status bit set (INTR[IPCMDERR] or INTR[AHBCMDERR]).	-	No transaction on SPI interface.
STOP	6'h00		Stop execution, deassert CS. Next command sequence (to the same flash device) will started from instruction pointer 0.	-	

The programmable sequence engine allows the software to configure the FlexSPI LUT according to external serial device connected on board. The flexible structure is easily adaptable to new command/protocol changes from different vendors.

DDR sequence is a flash access sequence that contain DDR instruction which is not DUMMY, it may contain SDR instructions optionally. SDR sequence is a flash access sequence that don't contain any DDR instruction. FlexSPI controller determines instruction SDR or DDR mode by decoding bit 5 of instruction opcode. The output/input timing on FlexSPI is different for SDR and DDR sequences. Especially note that SDR instruction in SDR sequence and DDR sequence is executed differently. Refer FlexSPI Input Timing and FlexSPI Output Timing for more details.

27.3.7.1 Instruction execution on SPI interface

This section describes the detail of instruction execution on SPI interface. For all instructions transmitting/receiving data bits to/from flash, the bit order in one byte is higher on DATA3 than DATA0, and higher on B port than A port.

1. **Command Instruction**

   Command Instruction (CMD_SDR/CMD_DDR) is normally used to transmit command code to external flash device. Command code is the 8 bits operand in instruction. Command code will be send to both B port and A port in parallel mode. Refer Flash access sequence example for examples.

2. **Address Instruction**

   Address Instructions (RADDR_SDR/RADDR_DDR/CADDR_SDR/CADDR_DDR) are normally used to send Flash access start address (Row/Column Address) to external flash device. Row/Column Address bits are determined by FlexSPI according to AHB access address or IP command address. Refer Flash address sent to Device for more details. The bit number is the operand value in instruction code. Row/Column address bits will be send to both B port and A port in parallel mode. For memories that read the flash address on 4 SCK edges, the sum of CADDR +RADDR must be 32. Otherwise, the FlexSPI will not generate 4 SCK edges of address phase. Refer Flash access sequence example for examples.

3. **Mode Instruction**

   Mode Instructions (MODEx_SDR/MODEx_DDR) are normally used to send mode bits to external flash device. Mode bits are the (lower) bits value of the instruction operand. The transition bit number is 1 for MODE1_SDR/MODE1_DDR, 2 for MODE2_SDR/MODE2_DDR, 4 for MODE4_SDR/MODE4_DDR and 8 for MODE8_SDR/MODE8_DDR. Note that pad number should be no more than mode

bit number. For example, it is not allowed to set num_pads to 2'b11 (Octal mode) for MODE4_* instructions. Mode bits will be send to both B port and A port in parallel mode. Refer Flash access sequence example for examples.

4. **Data Size Instruction**

   Data Size Instructions (DATSZ_SDR/DATSZ_DDR) are used to send program/read data size (byte number) to external device. This instruction is normally used in FPGA application when the memory space in external device acts similar as a FIFO. The device needs data size information to determine how much data will be popped from or pushed into internal FIFO. The bit number is the operation value in the instruction. Data size bits will be send to both B port and A port in parallel mode. Refer Flash access sequence example for examples.

5. **Write Instruction**

   Write Instructions (WRITE_SDR/WRITE_DDR) are normally used to send program data to external device. Programming data are fetched from IP TX FIFO (IP Command) or AHB TX Buffer (AHB command). For more details about flash program data size, refer Flash access by AHB Command and Flash access by IP Command. The byte order for programming data is always from low to high. Odd bytes are send on A port and Even bytes are send on B port in parallel mode. Refer Flash access sequence example for examples.

6. **Read Instruction**

   Read Instructions (READ_SDR/READ_DDR) are normally used to receive flash data from external device. Received data will be put into IP RX FIFO (IP Command) or AHB RX Buffer (AHB command). For more detail about flash read data size, refer Flash access by AHB Command and Flash access by IP Command. The byte order for reading data is always from low to high. Odd bytes are received from A port and Even bytes are received from B port in parallel mode. Refer Flash access sequence example for examples.

7. **Dummy Instruction**

   Dummy Instructions (DUMMY_SDR/DUMMY_DDR/DUMMY_RWDS_SDR/DUMMY_RWDS_DDR) are used to provide turnaround cycles on SPI interface. During dummy instruction, neither FlexSPI controller nor external device drives SPI interface. Refer Programmable Sequence Engine for more details about dummy cycle number.

   DUMMY_RWDS_SDR is reserved for future. FlexSPI controller checks DQS pin input level at the 4th cycle after SCLK output toggling is enabled. Refer Flash access sequence example for examples.

NOTE

The FlexSPI releases the bus after at least one cycle. Therefore, to avoid data contention, number of dummy lines should be programmed more than 1 if data is transferred on more than 1 line.

27.3.7.2 Flash access sequence example

Following is an example for SDR single I/O Read sequence (Cypress Serial Nor Flash S25FS512S) in individual mode.

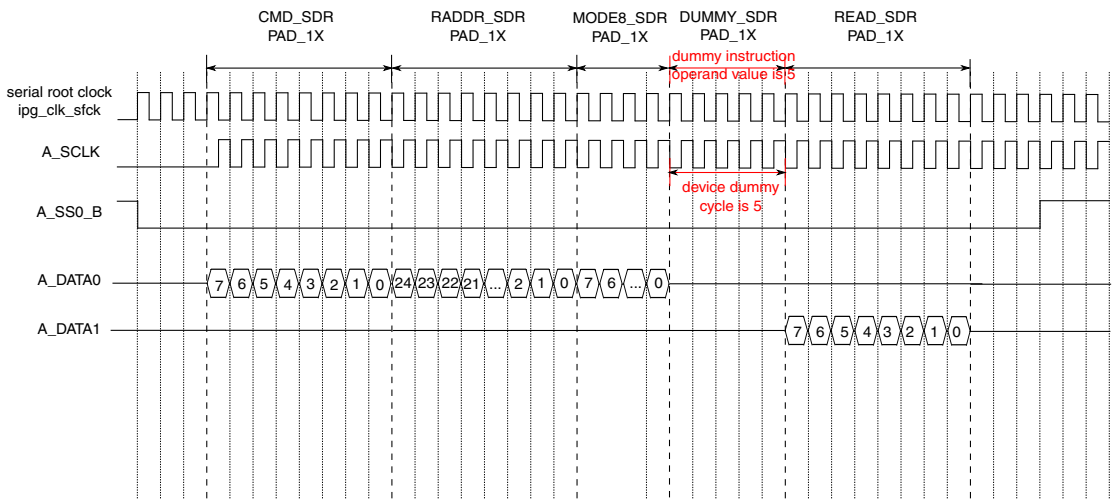


Figure 27-6. Flash access sequence example (SDR Single I/O Read sequence)

NOTE

- FlexSPI dummy instruction starts and ends at serial root clock rise edge.
- Device dummy cycle starts and ends at SCLK fall edge (on Cypress S25FS512S datasheet).

Following is an example for SDR Quad I/O Read sequence (Cypress Serial Nor Flash S25FS512S) in individual mode.

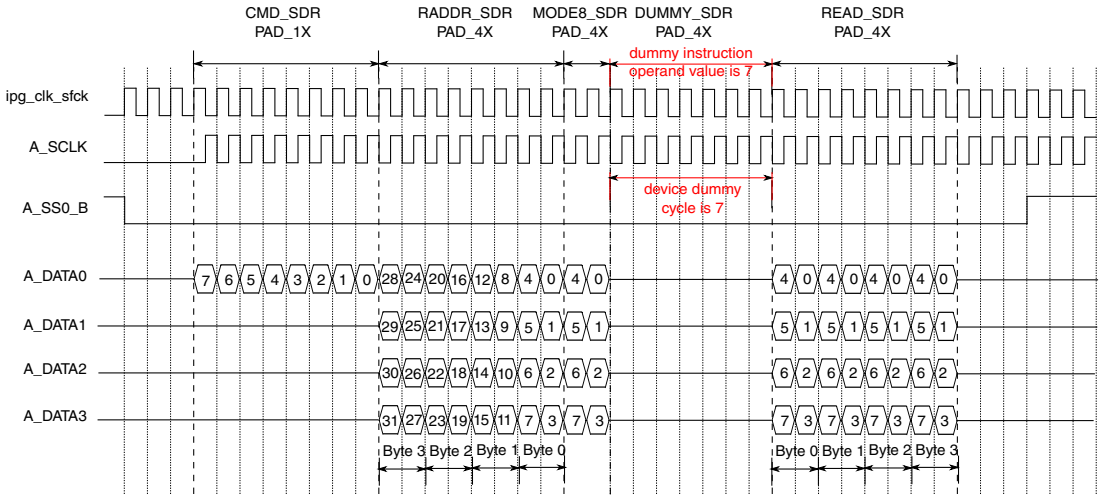


Figure 27-7. Flash access sequence example (SDR Quad I/O Read sequence)

NOTE

- FlexSPI dummy instruction starts and ends at serial root clock rise edge.
- Device dummy cycle starts and ends at SCLK fall edge (on Cypress S25FS512S datasheet).

Following is an example for DDR Quad I/O Read sequence (Cypress Serial Nor Flash S25FS512S) in parallel mode.

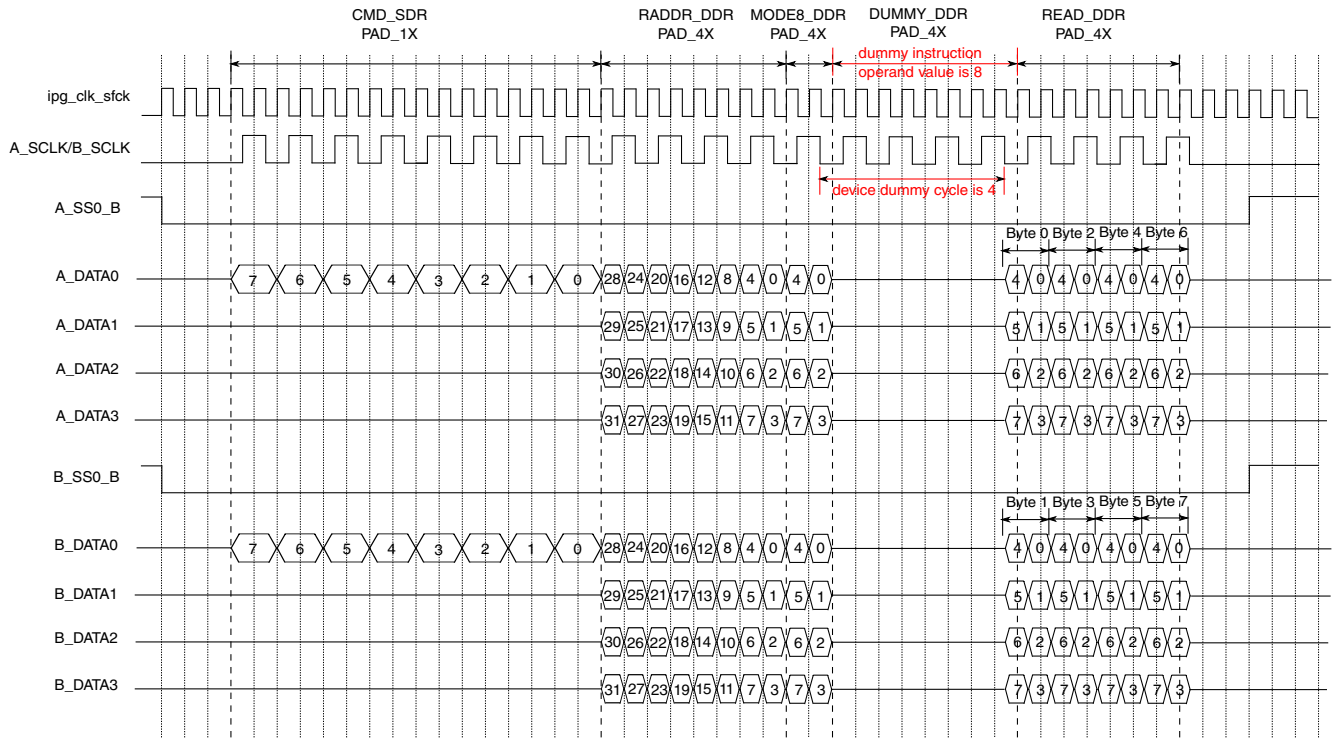


Figure 27-8. Flash access sequence example (DDR Quad I/O Read sequence)

27.3.8 Clocks

This section describes clocks and special clocking requirements of the FlexSPI module.

Table 27-5. Clock Usage

Clock Name	Description	Comment
serial clock root (ipg_clk_sfck)	Root clock for Serial domain	-
ahb clock (hclk)	AHB Bus clock	-
ipg clock (ipg_clk)	IPS Bus clock	-
SCLK	FlexSPI serial clock. Output clock on SCLK pin	Half clock frequency of serial clock root in DDR mode, and same frequency as serial clock root in SDR mode. Clock output toggles during the whole flash access sequence.
DQS_OUT	Dummy Read Strobe output	Same frequency as SCLK. Clock output toggles during READ/LEARN instructions.
DQS_IN	Sample clock for RX Data	Same frequency as SCLK. Sample clock comes from loopbacked dummy read strobe, loopbacked SCLK or Flash provided read strobe.

NOTE

- hclk should be **synchronous** with ipg_clk, but ipg_clk can be integer times of slower than hclk, for example hclk frequency = 4/3/2 X ipg_clk frequency.
- ipg_clk_sfck is **asynchronous** with hclk and ipg_clk.
- ipg_clk_sfck should be **synchronous** with its all divided version of generated clock.
- DQS is **asynchronous** with ipg_clk_sfck or hclk or ipg_clk.
- Please refer flexspi_mix_wrapper/contraints/ flexspi_mix_wrapper.sdc for clock domain information.

27.3.9 Interrupts

This section describes all the interrupts that the FlexSPI module generates.

- **IP command done interrupt**

  When IP command is finished, there will be interrupt generated if INTEN[IPCMDDONEEN] is set to 0x1.

- **IP command grant error interrupt**

  When IP command grant timeout (not grant after MCR0[IPGRANTWAIT] * 1024 ahb clock cycles), there will be interrupt generated if INTEN[IPCMDGEEN] is set to 0x1.

- **AHB command grant error interrupt**

  When AHB command grant timeout (not grant after MCR0[AHBGRANTWAIT] * 1024 ahb clock cycles), there will be interrupt generated if INTEN[AHBCMDGEEN] is set to 0x1.

- **IP command error interrupt**

  When there is command check error or command execution error for IP command, there will be interrupt generated if INTEN[IPCMDERREN] is set to 0x1. Refer Overview of Error Flags for more details.

- **AHB command error interrupt**

  When there is command check error or command execution error for AHB command, there will be interrupt generated if INTEN[AHBCMDERREN] is set to 0x1. Refer Overview of Error Flags for more details.

- **IP RX FIFO watermark available interrupt**

  When the fill level of IP RX FIFO is no less than watermark level (IPRXFCR[RXWMRK]), there will be interrupt generated if INTEN[IPRXWAEN] is set to 0x1.

- **IP TX FIFO watermark exceed interrupt**

  When the empty level of IP TX FIFO is no less than watermark level (IPTXFCR[TXWMRK]), there will be interrupt generated if INTEN[IPRXWAEN] is set to 0x1.

- **Sequence execution timeout interrupt**

  When a sequence execution time exceeds the timeout wait time (MCR1[SEQWAIT]), an interrupt will be generated if INTEN[SEQTIMEOUTEN] is set to 0x1. For example, the following flash read command sequence will lasts about 8000000 cycle (ipg_clk_sfck). If SEQWAIT is set 0xFFFF, there will be sequence timeout interrupt generated.

  - Triggered by IP command
  - Flash read data size is 0x1000000 bytes
  - Flash accessed in Single mode and SDR mode,

- **AHB Bus timeout interrupt**

  When AHB bus response timeout, there will be interrupt generated if INTEN[AHBBUSTIMEOUTEN] is set to 0x1. A typical case is AHB read sequence is not configured properly in LUT (such as without READ instruction). There will never be data read from external device, then FlexSPI will never hit the read data in AHB RX Buffers for AHB Read command.

- **SCLK stopped by write command interrupt**

  When IP TX FIFO is empty during write command sequence execution, FlexSPI will stop SCLK output clock toggling and wait for write data filling. At this time, this interrupt is generated if enabled by INTEN register.

- **SCLK stopped by read command interrupt**

  When IP RX FIFO is full during read command sequence execution, FlexSPI will stop SCLK output clock toggling and wait for read data read out from IP RX FIFO. At this time, this interrupt is generated if enabled by INTEN register.

27.3.10 Flash access by IP Command

Flash access could be triggered by IP command in following steps.

- Fill IP TX FIFO with programming data if this is a programing command (programming flash data, flash status registers etc.)
- Set flash access start address (IPCR0[SFAR]), read/program data size, sequence index in LUT and sequence number (IPCR1[ISEQNUM]).
- Trigger flash access command by writing 1 to register bit IPCMD[TRG]
- Polling register bit INTR[IPCMDDONE] to wait for this IP command to finish on FlexSPI interface.

NOTE

- IP TX FIFO could be filled before or after writing IPCR0/ IPCR1/IPCMD register. If SFM command is started with IP TX FIFO empty, FlexSPI will stop SCLK toggling to wait for TX data ready automatically.
- IPCMD register must be written after writing IPCR0/ IPCR1 register.
- Multiple Command sequences (8 at most) could be issued by one IP command.
- It is not allowed to issue another IP command before the previous IP command is finished. The behaviour is unknown in this case.

If this is a Reading command to Serial Flash Memory, all reading data from Flash will be put into IP RX FIFO. Software will need to read out data from IP RX FIFO by AHB bus or IP Bus. When IP RX FIFO is full and there is more data to be read from Flash device, FlexSPI will stop SCLK output clock toggling until there is free space in IP RX FIFO. See SCLK stop feature for more details.

The detail of triggered Serial Flash Command is as following:

- Flash access start address:

  Determined by register field IPCR0[SFAR]

- Flash Chip Select:

  Determined by flash access address and Flash size setting (FLSHxCR0[FLSHSZ]).

- Flash command Sequence Index and Sequence Number:

  The sequences indexed from IPCR1[ISEQID] to (IPCR1[ISEQID] + IPCR1[ISEQNUM]) in LUT will be executed by FlexSPI sequentially.

- Flash Individual/Parallel access mode

Determined by IPCR1[IPAREN].

- Flash Read/Program Data Size:
  - If IPCR1[IDATSZ] value is non-zero, flash read/program data size (in byte) is IPCR1[IDATSZ].
  - If IPCR1[IDATSZ] value is zero, flash read/program data size (in byte) will the operand value in the READ/WRITE instruction.

NOTE

- Software should make sure the last sequence index never exceeds the LUT sequence number (IPCR1[ISEQID] + IPCR1[ISEQNUM] < 16).
- Data size is applied to every command sequence if sequence number is more than one.
- Data size is ignored if there is no WRITE/READ instruction in the command sequence .

IP command request is sent to arbitrator after triggered by software. It is not executed on FlexSPI Interface until granted by arbitrator. Please refer to Command Arbitration for more details.

27.3.10.1 Reading Data from IP RX FIFO

FlexSPI put the read data from external device into IP RX FIFO for IP command.

These data could be read out by following two memory space access.

- By IP Bus
- By AHB Bus

If MCR0[ARDFEN] is set to 0x1, read data in IP RX FIFO could only be read out by AHB Bus, IP Bus read access to IP RX FIFO will always return with data zero and no bus error occur.

If MCR0[ARDFEN] is set to 0x0, read data in IP RX FIFO could only be read out by IPS Bus, AHB Bus read access to IP RX FIFO will trigger bus error.

FlexSPI push read data into IP RX FIFO in terms of 64 bits every time it receives 64 bits data from external device. When read data bits number is not 64 bits aligned, FlexSPI will push additional zero bits into IP RX FIFO for the last push.

IP RX FIFO could be read by processor or DMA. Following is the detail flow for processor and DMA reading:

1. **Reading by processor**

To read by processor, following register settings are needed:

- Set register field IPRXFCR[RXDMAEN] to 0.
- Set watermark level by IPRXFCR[RXWMRK], watermark level is (IPRXFCR[RXWMRK]+1)*8 bytes.
- Set register field INTEN[IPRXWAEN] to enable IP RX FIFO watermark available interrupt (optional).

Processor needs to poll register INTR[IPRXWA] or wait for IP RX FIFO Watermark Available interrupt before reading IP RX FIFO. This is to make sure there is a watermark level Data filled in IP RX FIFO before reading.

After reading a watermark level data from IP RX FIFO, software need to set register bit INTR[IPRXWA]. This set action will pop out a watermark level data from IP RX FIFO.

Following diagram indicates the reading flow from IP RX FIFO by processor.

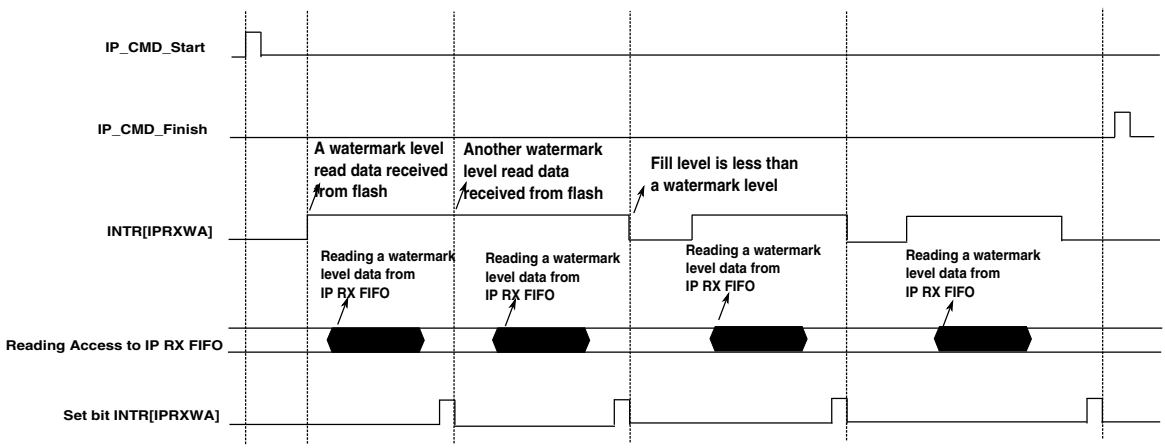


Figure 27-9. Reading IP RX FIFO by processor

NOTE

- Processor need to read out a watermark level data from IP RX FIFO each time before set register bit INTR[IPRXWA].
- It's supported that the total flash read/program data size is not multiple of watermark level. In this case, the reading data size from IP RX FIFO will be less than a watermark level for the last time, software should poll IPRXSTS[FILL] field instead of polling INTR[IPRXWA]. After copying all the data from IP RX FIFO and all command sequences to Flash are finished (INTR[IPCMDDONE]=1), software should

clear IP RX FIFO by setting IPRXFCR[CLRIPRXF]. Otherwise, the reading data will be wrong for the next reading command to Flash.

- IP RX FIFO data is not popped out by each reading access to IP RX FIFO, but popped by writing 0x1 to register INTR[IPRXWA] bit.

2. **Reading by DMA**

To read IP RX FIFO by DMA, following setting is needed:

- Set register field IPRXFCR[RXDMAEN] to 1.
- Set watermark level by IPRXFCR[RXWMRK], watermark level is (IPRXFCR[RXWMRK]+1)*8 bytes.
- Set DMA transfer Minor loop size to same watermark level.

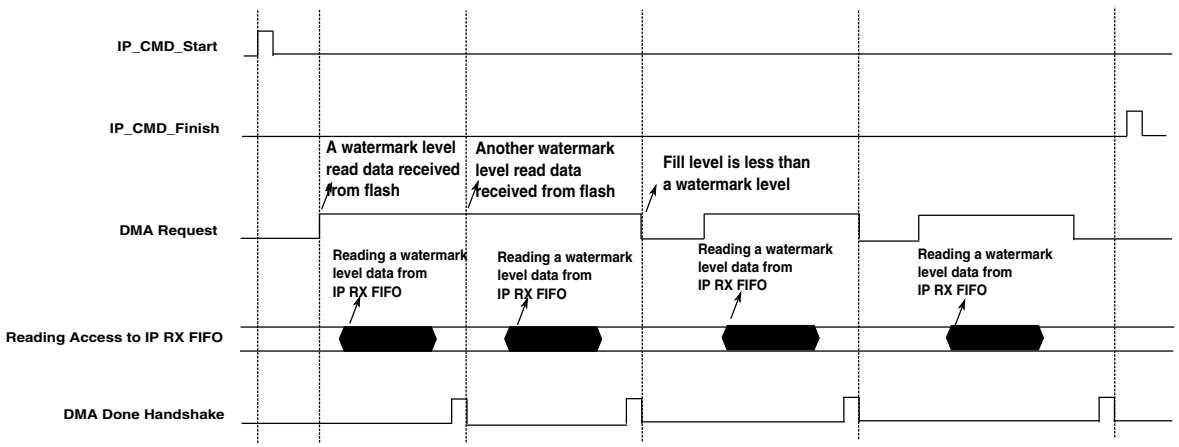


Figure 27-10. Reading from IP RX FIFO by DMA

NOTE

- DMA request is generated when the fill level of IP RX FIFO is higher than (or equal) watermark level. This request is not pulse valid but level valid.
- DMA should read out watermark level data from IP RX FIFO each time (set minor loop size with the same value as watermark level.
- DMA need to return a Done handshake (pulse valid) to FlexSPI each time it finished reading a watermark level data.
- IP RX FIFO data is not popped out by each reading access, but popped by DMA done handshake.
- It is not supported that the total read/write data size (Major loop size) is not multiple of watermark level.

Because the DMA does not know when the data is ready for the last reading. It makes the DMA driver too complex to poll IPRXFSTS [FILL] field.

27.3.10.2 Filling Data to IP TX FIFO

The programming should be put into IP TX FIFO and then transmit to Flash by FlexSPI. It could be filled by two memory space

- By IP Bus
- By AHB Bus

To fill by AHB Bus, need to set MCR0[ATDFEN] to 1, IP Bus write access to IP TX FIFO will be ignored, but no bus error.

To fill by IP Bus, need to set MCR0[ATDFEN] to 0, AHB Bus write access to IP TX FIFO will return Bus Error.

IP TX FIFO is popped with 64 bits data every time FlexSPI fetch data for transmitting. If the programming data size is not multiple of 64 bits, last popped valid bits will be less than 64 bits. But there is no problem because these invalid bits are not transmitted to Flash at all.

IP TX FIFO could be filled by processor or DMA.

To fill by processor, need following setting:

- Set register field IPTXFCR[TXDMAEN] to 0.
- Set watermark level by IPTXFCR[TXWMRK], watermark level is (IPTXFCR[TXWMRK]+1)*8 bytes.
- Set register field INTEN[IPTXWEEN] to enable IP TX FIFO watermark empty interrupt (optional).

Processor needs to poll register INTR[IPTXWE] or wait for IP TX FIFO Watermark empty interrupt before filling IP TX FIFO. This is to make sure there is enough space for a watermark level Data filling before filling.

After filling a watermark level data to IP TX FIFO, need to set register bit INTR[IPTXWE]. This will push a watermark level data into IP TX FIFO (write pointer is incremented).

NOTE

IP TX FIFO data is not pushed by each write access, only pushed by set INTR[IPTXWE] bit.

Following digram indicates the filling flow to IP TX FIFO by processor.

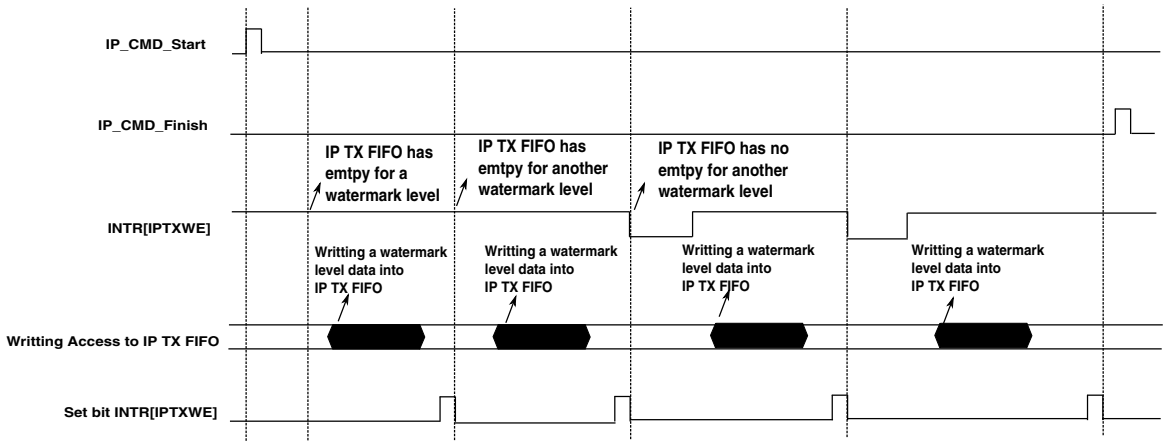


Figure 27-11. Filling IP TX FIFO by processor

NOTE

- Processor will need to fill a watermark level data to IP TX FIFO each time.
- It's allowed that total write data size is not multiple of watermark level. In this case, the writing size will be less than a watermark level for the last time. After filling all data into IP TX FIFO and all Command Sequences to Flash are finished (INTR[IPCMDDONE]=1), processor should clear IP TX FIFO by setting IPTXFCR[CLRIPTXF]. Otherwise, the programming data will be wrong for the next programming Command to Flash.

To fill IP TX FIFO by DMA, need following setting:

- Set register field IPTXFCR[TXDMAEN] to 1.
- Set watermark level by IPTXFCR[TXWMRK], watermark level is (IPTXFCR[TXWMRK]+1)*8 bytes.
- Set DMA transfer Minor loop size to same watermark level.

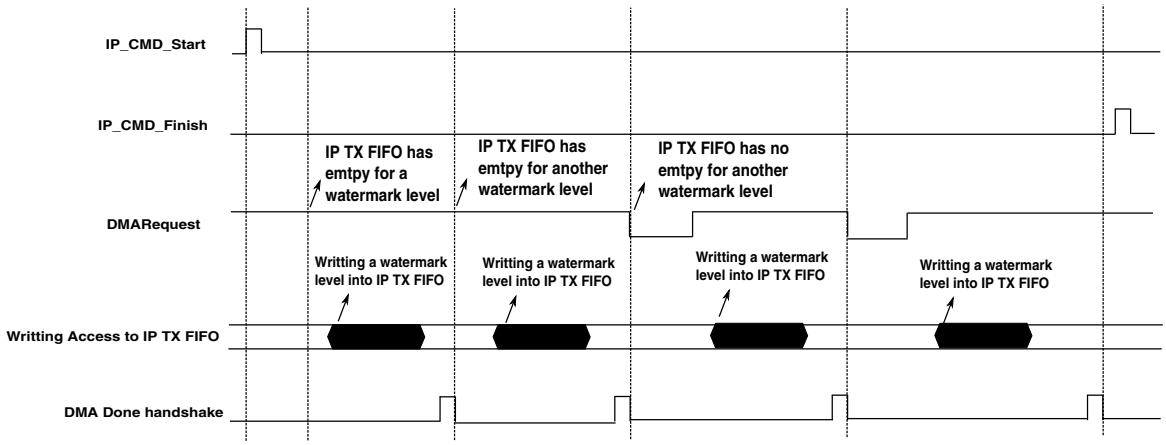


Figure 27-12. Filling from IP TX FIFO by DMA

NOTE

- DMA request is generated when there is empty space more than watermark level in IP TX FIFO. This request is not pulse valid but level valid.
- DMA should fill watermark level data into IP TX FIFO each time (set minor loop size with the same value as watermark level.
- DMA need to return a Done handshake (pulse valid) to FlexSPI each time it finished filling a watermark level data.
- IP TX FIFO data is not pushed in by each reading access, but pushed by DMA done handshake.
- It's allowed that total program data size (Major loop size) is not multiple of watermark level. After filling all data into IP TXFIFO and all command sequences to Flash are finished (INTR[IPCMDDONE]=1), need to clear IP TX FIFO by setting IPTXFCR[CLRIPTXF]. Otherwise, the programming data will be wrong for the next programming Command to Flash.

27.3.11 Flash access by AHB Command

Flash can be accessed by AHB bus directly on AHB address space. This address space is mapped to Serial Flash Memory in FlexSPI. AHB bus accesses to this address space triggers Flash access command sequences as needed.

For AHB read access to Serial Flash Memory, FlexSPI will fetch data from flash into AHB RX Buffers and then return the data on AHB Bus. For AHB write access to Serial Flash Memory, FlexSPI will buffer AHB Bus write data into AHB TX Buffer and then transmit to Serial Flash memory.

There is no software configuration or polling need for AHB command except FlexSPI initialization. AHB master access external flash device transparently similar as normal AHB slave.

AHB command is normally used to access serial Flash memory space. IP command should be used to access the control and status registers or other spaces such as OTP in external flash device.

The following section describes the AHB commands for read and write in more detail.

NOTE

When FlexSPI controller return AHB bus error for SFM access, AHB master should stop following access beats in current burst.

27.3.11.1 AHB write access to Flash

For AHB write access to Flash, FlexSPI will buffer the write data from AHB bus into internal AHB TX Buffer and then transmit them to Flash. FlexSPI only buffers write data for one AHB burst. Following diagram indicates the hardware operation in response to AHB write access to Flash.

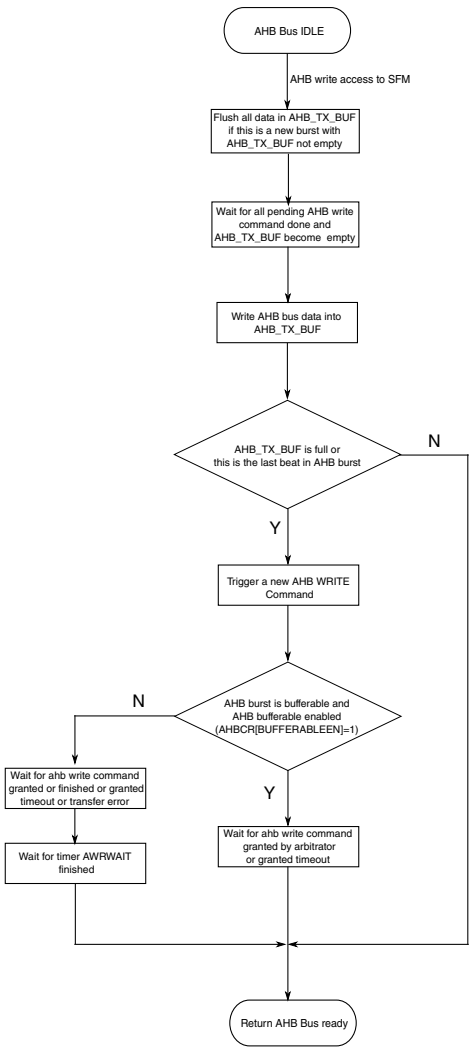


Figure 27-13. Hardware operation in response to AHB write access to Flash

FlexSPI triggers new AHB write command in following cases:

- This beat is the last one in current AHB burst (any burst type except INCR).
- AHB TX Buffer is full after buffering current beat data.
- AHB bus becomes IDLE or a new burst comes with AHB TX Buffer not emtpy.

The detail information about the triggered AHB write command:

- Flash Access Start Address:

  Determined by AHB burst address. FlexSPI will record the start address for the data in AHB TX Buffer and this address will be used as flash access start address

- Flash Chip Select:

FlexSPI determined the chip selection by flash access start address and flash size setting.

- Flash Command Sequence Index:

  Determined by FLSHxCR2[AWRSEQID].

- Flash Command Sequence Number:

  Determined by FLSHxCR2[AWRSEQNUM]. If AWRSEQNUM is not zero, multiple flash access command sequences will be triggered every time for AHB write command. The sequences indexed from AWRSEQID to (AWRSEQID + AWRSEQNUM) in LUT will be executed sequentially.

- Flash access mode (Individual/Parallel)

  Determined by AHBCR[APAREN].

- Flash Data Size:

  Determined by byte number of buffered data in AHB TX Buffer.

Following examples indicates internal logic for AHB write access to Flash. In these examples, AHB_TX_BUF is 64 Bytes (8*64bits) .

- AHB INCR/64bit/Bufferable burst with address sequence 0x8, 0x10, 0x18, 0x20, ..., 0x50 (10 beat totally):

  Two AHB write command will be triggered: the first command with flash start address 0x8 and data size 0x40; the second command with flash start address 0x48 and data size 0x10. See Figure 27-14.

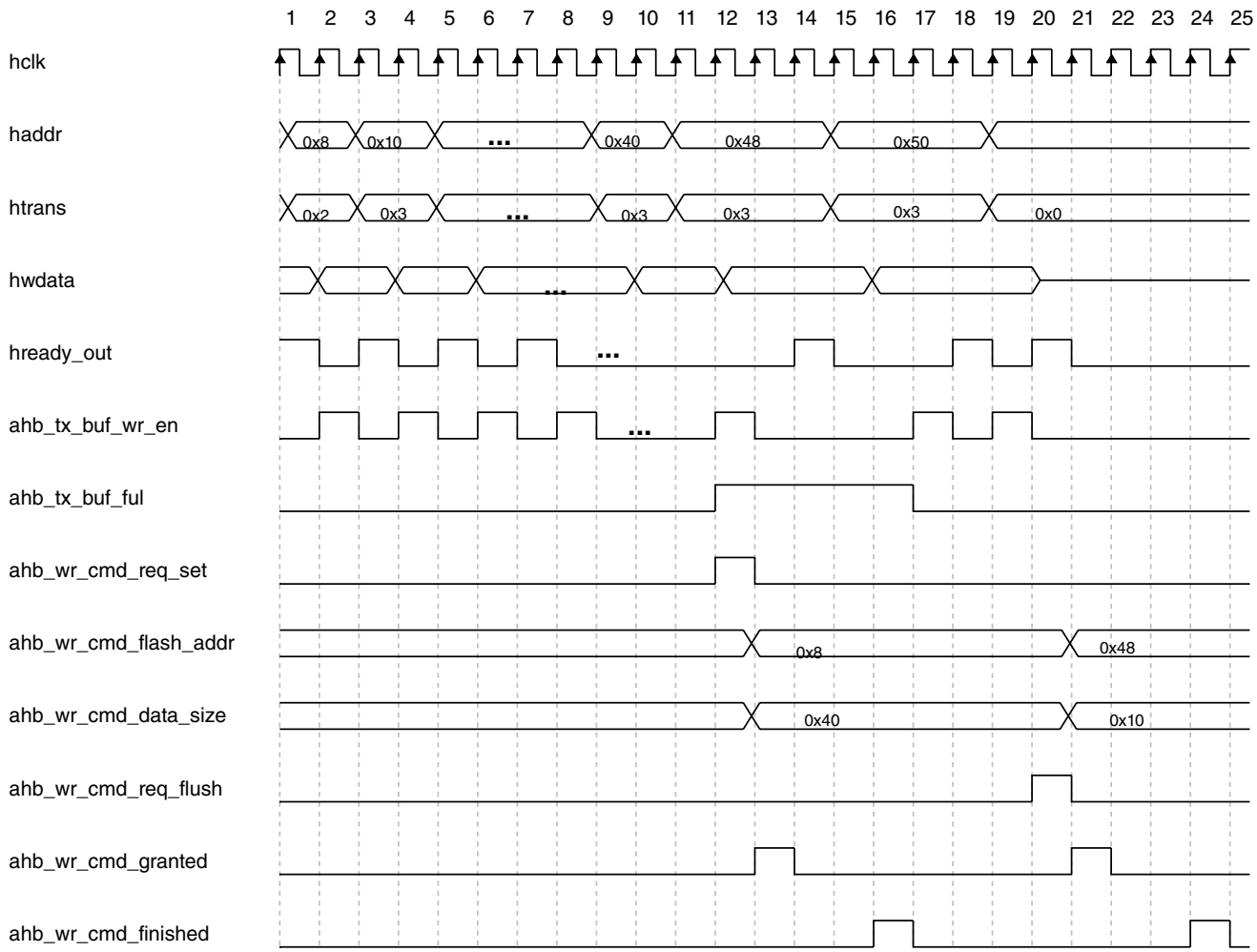


Figure 27-14. AHB write access (INCR/64bit/Bufferable)

- AHB WRAP8/64bit/Bufferable burst with address sequence 0x28, 0x30, 0x38, 0x0, 0x8, 0x10, 0x18, 0x20:

  One AHB write command will be triggered with flash start address 0x0 and data size 0x40. See Figure 27-15.

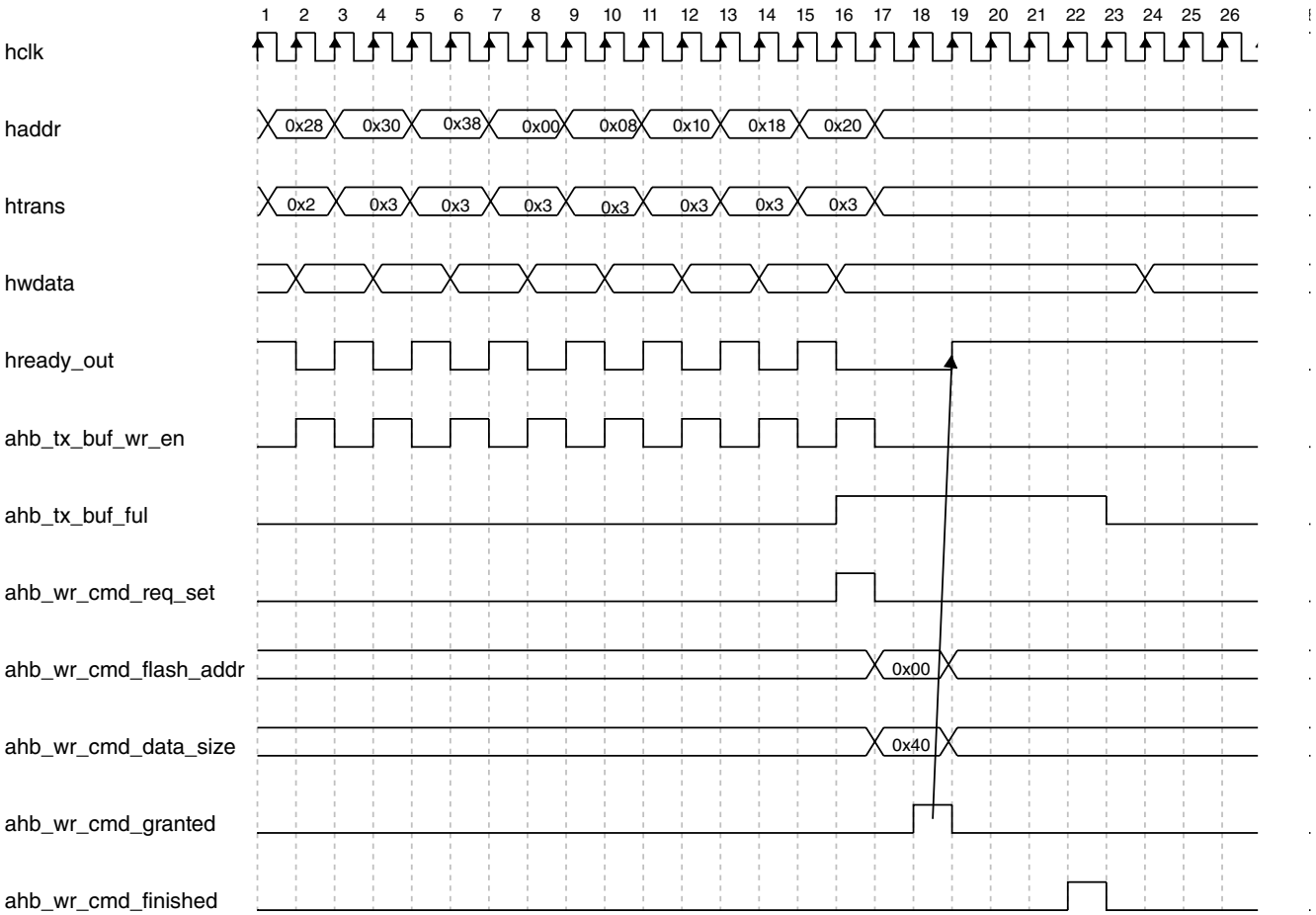


Figure 27-15. AHB write access (WRAP8/64bit/Bufferable)

- AHB WRAP8/64bit/Non-bufferable burst with address sequence 0x28, 0x30, 0x38, 0x0, 0x8, 0x10, 0x18, 0x20:

  One AHB write command will be triggered with flash start address 0x0 and data size 0x40;

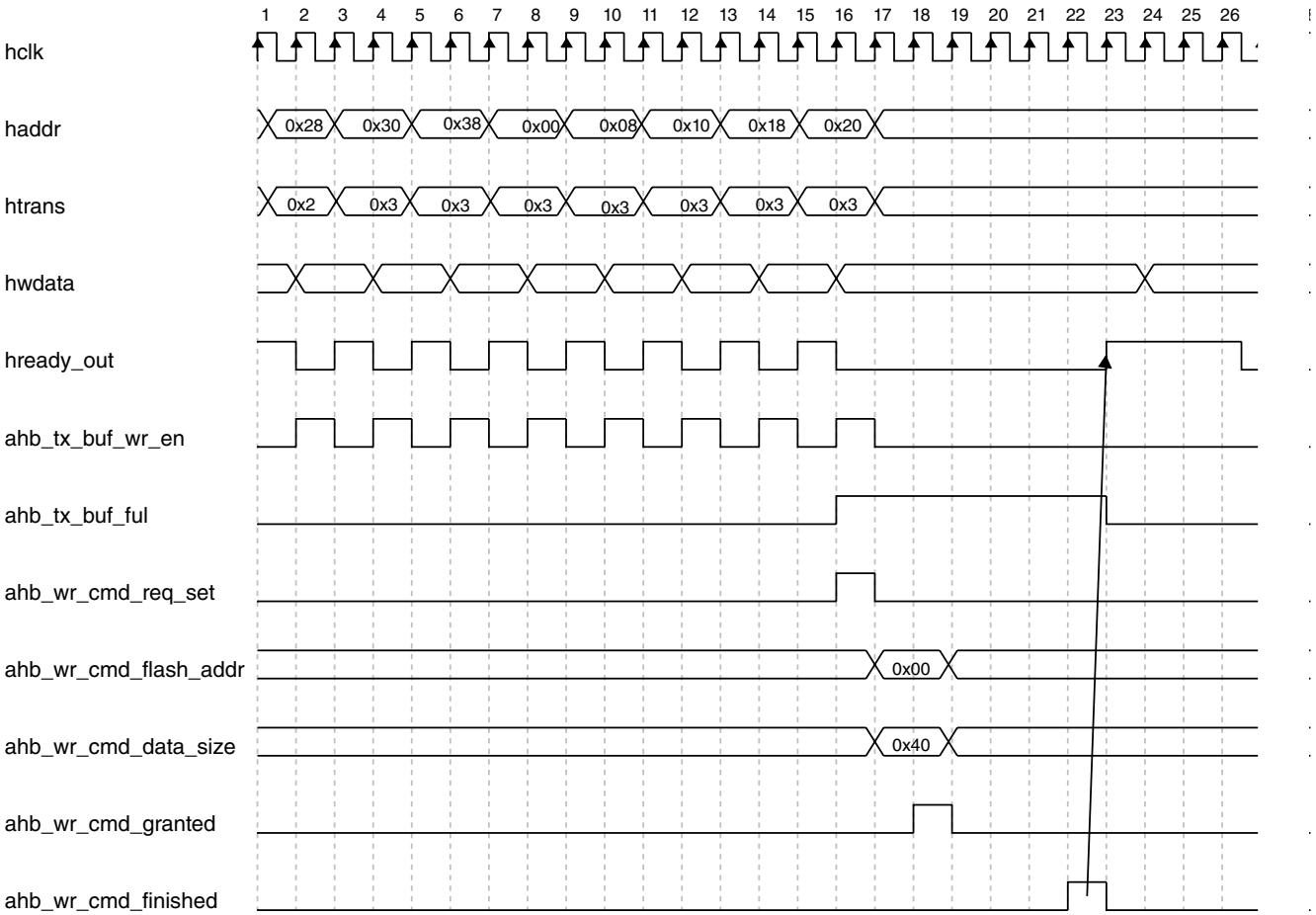


Figure 27-16. AHB write access (WRAP8/64bit/Non-Bufferable)

NOTE

The wrapper burst is not supported if burst data size (in byte) is larger than AHB TX Buffer size (in byte). For example, if AHB_TX_BUF is 64 Bytes (8*64bits), AHB WRAP16 * 64bit writet burst access is not supported.

27.3.11.2 AHB read access to Flash

For AHB read access to Flash, FlexSPI will check whether hit AHB TX Buffer/AHB RX Buffer/pending AHB read command according to the burst access type and register setting. If all these miss, FlexSPI trigger a new AHB read command to fetch data from Flash. Following diagram indicates the hardware operation in response to AHB read access to Flash.

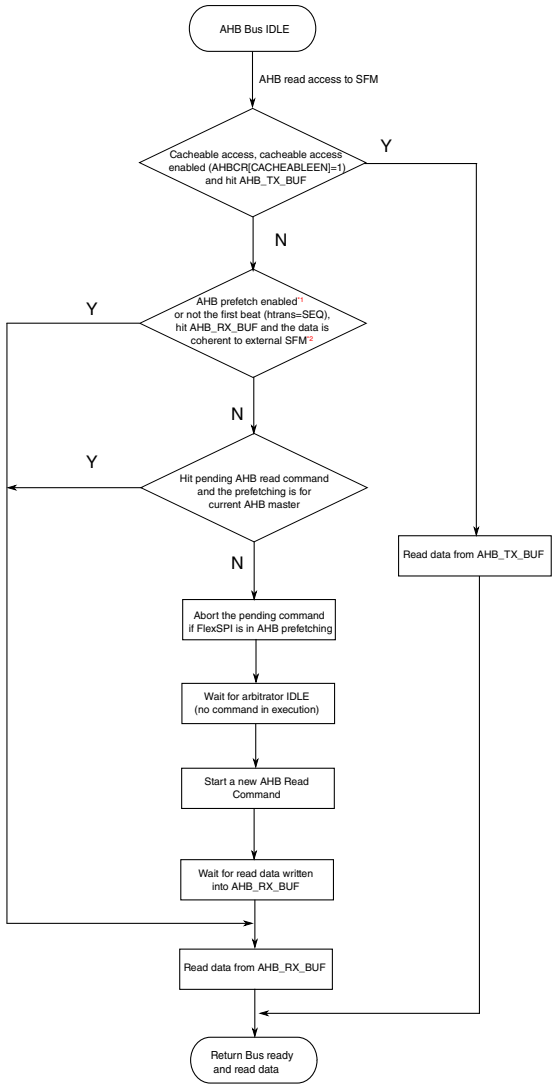


Figure 27-17. Hardware operation in response to AHB read access to Flash

NOTE

1. AHB prefetch is enabled when both AHBCR[PREFETCHEN]=0x1 and AHBRXBUFxCR0[PREFETCHEN]=0x1 (x is the ahb rx buffer ID for current AHB master).
2. AHB RX Buffer holds the data read from Flash. This data may become incoherent if the AHB writes to the same flash address.

The detail information about the triggered AHB read command:

- Flash Access Start Address and Data Size:

Determined by AHB address, burst type and burst size. FlexSPI will fetch read data from the start address for current burst or beat.

Table 27-6. AHB read command flash start address and data size

Prefetch Enable	Cross flash boundary	Burst Type	Flash start address [28:0]	Data Size
0	Never cross flash boundary because AHB burst never cross 1K Byte boundary.	SINGLE/INCR4/ INCR8/INCR16	hbeat_start_address	(hburst_end_address- hbeat_start_address)
		INCR	hbeat_start_address	byte size of current beat For INCR burst with prefetch disabled, each beat is handled same as SINGLE burst.
		WRAP4/ WRAP8/ WRAP16	hburst_start_address	(hburst_end_address- hburst_start_address)
1	No	INCR/SINGLE/ INCR4/INCR8/ INCR16	hbeat_start_address	ahb_rx_buf_sz
	No	WRAP4/ WRAP8/ WRAP16	hburst_start_address	ahb_rx_buf_sz
	Yes	INCR/SINGLE/ INCR4/INCR8/ INCR16	hbeat_start_address	(flash_top_address- hbeat_start_address)
	Yes	WRAP4/ WRAP8/ WRAP16	hburst_start_address	(flash_top_address- hburst_start_address)

NOTE

- hbeat_start_address is HADDR input from AHB master for current beat.
- hburst_start_address (for e.g.0x00) is the lowest address for current burst; hburst_end_address (for e.g. 0x10) is the highest address in current burst plus 1. For example, WRAP4 burst with HADDR 0x8,0xC, 0x0, 0x4.

- ahb_rx_buf_sz is the buffer size in byte of AHB RX Buffer which is used by current master.
- flash_top_address is the top address of currently accessed flash.

- Flash Chip Select:

  FlexSPI determined the chip selection by flash access start address and flash size setting.

- Flash Command Sequence Index:

  Determined by FLSHxCR2[ARDSEQID].

- Flash Command Sequence Number:

  Determined by FLSHxCR2[ARDSEQNUM]. If ARDSEQNUM is not zero, multiple flash access command sequences will be triggered every time for AHB read command. The sequences indexed from ARDSEQID to (ARDSEQID + ARDSEQNUM) in LUT will be executed sequentially.

- Flash access mode (Individual/Parallel):

  Determined by AHBCR[APAREN].

NOTE

- FlexSPI determines which ARDSEQNUM/ARDSEQID fields will be used as sequence ID by flash device chip selection automatically. See MCR2[SAMEDEVICEEN] for more details.
- FlexSPI determines which AHB RX Buffer used for current AHB read access by master ID. For more details about the AHB RX buffer ID and AHB master ID mapping, see AHB RX Buffer Management.
- It is not allowed to allocate AHB RX Buffer less than AHB Burst size. The behaviour is unknown for this case.

27.3.11.3 AHB RX Buffer Management

There are 4 buffers (Buffer 0 - Buffer 3) in AHB RX Buffer, which are transparent to AHB masters. FlexSPI fetch flash data and return on AHB Bus automatically. There is no status register polling needed for AHB read access to Serial Flash Memory.

See the chip-specific section for AHB buffer size. The buffer size is flexibly configurable for each buffer in AHB RX Buffers by register fields AHBRXBUF0CR0[BUFSZ]~AHBRXBUF6CR0[BUFSZ]. The buffer size for Buffer 0 to Buffer 3 could be set 0. If the buffer size is set to 0x0, the related MSTRID field setting (in same AHBRXBUFxCR0 register) is ignored by FlexSPI. Buffer 3 is used for all AHB masters which is not assigned to Buffer 0 - Buffer 2. Buffer 3 size setting field (AHBRXBUF3CR0[BUFSZ]) is ignored by FlexSPI, its buffer size is: AHB RX Buffer total size - sum of (Buffer 0 - Buffer 2 size).

When there is AHB read access to Serial Flash Memory, FlexSPI determines which AHB RX Buffer to use as following:

1. If master ID equal AHBRXBUF0CR0[MSTRID] and AHBRXBUF0CR0[BUFSZ] is not zero, Buffer 0 will be used.
2. If master ID equal AHBRXBUF1CR0[MSTRID] and AHBRXBUF1CR0[BUFSZ] is not zero, Buffer 1 will be used.
3. If master ID equal AHBRXBUF2CR0[MSTRID] and AHBRXBUF2CR0[BUFSZ] is not zero, Buffer 2 will be used.
4. If all above case not meet, Buffer 3 will be used.

NOTE

- Software should make sure the buffer size of each buffer is no less than the max burst size of AHB Read access from the master using this buffer. Otherwise the behaviour is undefined. The minimal buffer size request depends on the SoC implementation, for example, if the SoC's capability to send out a maximal burst is 64 Bytes, then minimal AHB RX buffer is 64 Bytes.
- It is not supported to assign multiple buffers for single AHB master.
- When AHB read prefetch is enabled (AHBCR[PREFETCHEN] is set), the prefetch data size will be determined by buffer size. FlexSPI will fetch data from external Flash with buffer size if no flash boundary across.
- AHB master priority setting (register field AHBRXBUFxCR0[PRIORITY] is used only for the suspending control of AHB prefetching. See Command Abort and Suspend.

27.3.12 Command Arbitration

There are four Flash access command sources:

1. AHB Command (triggered by AHB Write access to SFM space)
2. AHB Command (triggered by AHB Read access to SFM space)
3. IP command (triggered by writing IPCMD[TRG])
4. Suspended command (AHB Read prefetch sequence which is suspended)

NOTE

- An AHB bus access never triggers a write command and a read command at the same time.
- AHB prefetch sequence is an AHB Command sequence triggered by AHB Read access when AHB prefetch is enabled. After all read data fetched for current AHB read burst, FlexSPI will prefetch more data to reduce the read latency for next AHB read access. AHB command for read is never aborted while fetching read data for current AHB read burst. But AHB read command could be aborted by any new IP command or AHB command request when it's prefetching data (not for current read burst).

The granted priority of these 4 command source is as following when Arbitrator is idle (STS0[ARBIDLE]=1):

1. AHB command (Read/Write)
2. IP Command
3. Suspended Command

NOTE

Suspended command is not granted immediately when arbitrator is idle and no AHB/IP command request. Arbitrator will wait for n AHB clock cycle idle state before resuming the suspended command (n is the register field value in MCR2[RESUMEWAIT]). This intend to avoid AHB prefetch sequence being resumed and suspended frequently.

All command request are not granted if Arbitrator is busy in executing AHB/IP command (not suspended command), and there will AHB/IP Command granted error if the grant is timeout.

If new AHB/IP command request comes while Arbitrator is executing AHB read prefetch sequence, AHB read prefetch sequence will be aborted (but not immediately). Arbitrator will grant AHB/IP command request after AHB read prefetch sequence is aborted on FlexSPI interface and saved all internal data pointers.

27.3.12.1 Command Abort and Suspend

This section describes command abort and suspend mechanism.

1. **Command Abort**

   As mentioned above, AHB read prefetch sequence could be aborted if new AHB/IP command request comes.

2. **Command Suspend**

   When AHB read prefetch sequence is aborted on FlexSPI interface, FlexSPI will save this suspended sequence in following cases and resume this sequence if arbitrator is idle for enough time:

   - There is no valid suspended command (Register field AHBSPNDSTS[ACTIVE] is 0x0). This is possible if there is no suspended command yet or suspended command is resumed.
   - Aborted AHB read prefetch sequence is higher priority than current active suspend sequence.

   NOTE

   Original suspended sequence will be ignored and never resumed by FlexSPI.

3. **Suspended Command**

   The suspended command (internal status) turns active when there is any AHB prefetch command aborted and suspended. It turns inactive in following cases:

   - Suspended command is resumed by Arbitrator.
   - There is a new AHB read command request and it's triggered by AHB master using the same AHB RX Buffer (Buffer ID).

   NOTE

   - MCR0[SWRESET] can be used to clear any suspended command. It means saved suspended command will be dropped and no more resume will happen. Clearing suspended command is needed after flash page program/erase or any other command may lead period of time that flash can not support read command.

Following is an example indicating command abort/suspend/resume flow:

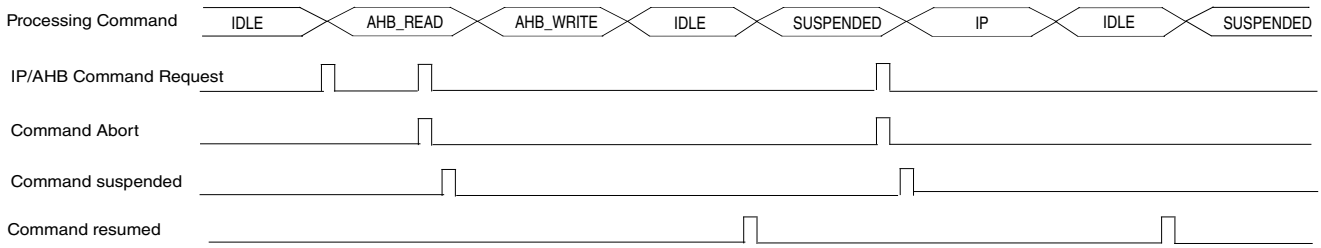


Figure 27-18. Command Instructions Execution on FlexSPI interface

27.3.13 SCLK stop feature

FlexSPI will stop SCLK output toggling when programming data is not ready for programming command sequence or there is no space (in internal FIFO) to receive data for reading command sequence.

There may be certain devices that do not support SCLK stopped during command sequence (chip select is valid). SCLK stopping could be avoid as following:

- For flash reading triggered by IP command
  - Never trigger a read command with data size larger than IP RX FIFO size.
  - Internal async FIFO for flash reading should never be full.

    FlexSPI pop data from this async FIFO in 64 bits per AHB clock cycle, and receiving data from FlexSPI interface in serial root clock. The receiving speed is determined by Flash access mode (Single/Dual/Quad/Octal mode and Individual/Parallel mode). For example, in Octal mode and Parallel mode, FlexSPI receives 16 bits per serial root clock cycle. This async FIFO is never full if AHB clock frequency is higher than 1/4 of serial root clock.

- For flash programming triggered by IP command
  - Never trigger a program command with data size larger than IP TX FIFO size.
  - Fill all programming data into IP TX FIFO before trigger the IP command.
  - Internal async FIFO for flash programming should never be empty

    FlexSPI fetch programming data into this async FIFO in 64 bits per AHB clock cycle, and transmitting data to FlexSPI interface in serial root clock. The transmitting speed is determined by Flash access mode (Single/Dual/Quad/Octal mode and Individual/Parallel mode). For example, in Octal mode and Parallel mode, FlexSPI transmits 16 bits per serial root clock cycle. This async FIFO is never empty if AHB clock frequency is higher than 1/4 of serial root clock.

- For flash reading/programming triggered by AHB command

- Internal async FIFO for flash reading/programming should never be full/empty. The frequency ratio limitation is same as flash reading/programming triggered by IP command.

NOTE

- FlexSPI never trigger a AHB read command with data size larger than internal AHB RX buffer size.
- FlexSPI never trigger a AHB program command with data size larger than internal AHB TX buffer size.
- All programming data is buffered into AHB TX Buffer before triggering AHB program command in FlexSPI.

27.3.14 FlexSPI Output Timing

This section describes the output timing in FlexSPI.

27.3.14.1 Output timing between Data and SCLK

This section describes the output timing relationship of data (on A_DATA/B_DATA) and SCLK. There are three cases for the data output timing:

- **SDR instruction in SDR sequence**
  SDR sequence is the sequence which contains only SDR instructions. In this case, all data bits last one serial root clock cycle on FlexSPI interface. Following diagram indicates the relationship of serial root clock, data and SCLK:

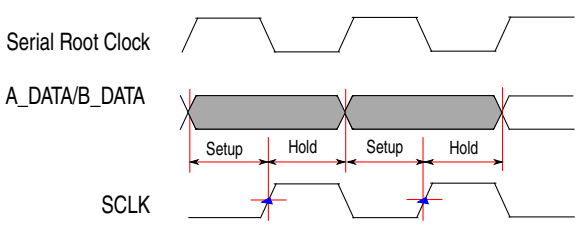


Figure 27-19. SDR instruction in SDR sequence

- **SDR instruction in DDR sequence**
  DDR sequence is a flash access command sequence that contain DDR instruction which is not DUMMY, it may contain SDR instructions optionally. In the case of SDR instruction in DDR sequence, all data bits last two serial root clock cycles on FlexSPI interface. Following diagram indicates the relationship of serial root clock, data and SCLK:

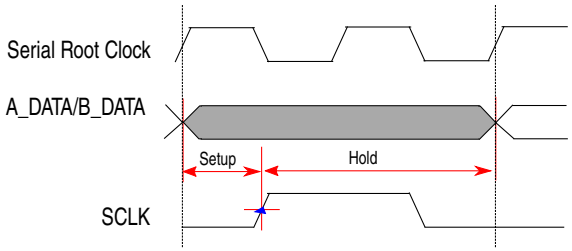


Figure 27-20. SDR instruction in DDR sequence

- **DDR instruction in DDR sequence**
  In the case of DDR instruction in DDR sequence, all data bits last one serial root clock cycle on FlexSPI interface. Following diagram indicates the relationship of serial root clock, data and SCLK:

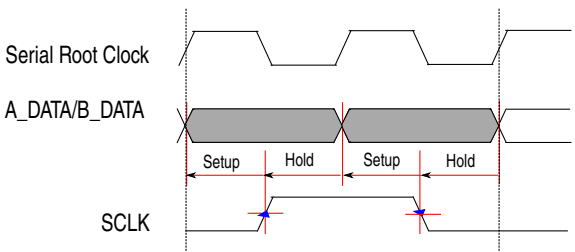


Figure 27-21. DDR instruction in DDR sequence

27.3.14.2 Output timing between Chip selection and SCLK

This section describes the output timing relationship of Chip select (on A_SS0_B/ A_SS1_B/B_SS0_B/B_SS1_B) and SCLK. The timing relationship is a little different for SDR sequence and DDR sequence.

- **Chip Select timing in SDR sequence**

  For SDR sequence, the delay from chip select assertion and the SCLK rising edge is **(FLSHxCR1[TCSS]+0.5)** cycles of serial root clock; The delay from SCLK falling edge and chip select deassertion is **FLSHxCR1[TCSH]** cycles of serial root clock. Following diagram indicates the timing relationship between chip selection and SCLK:

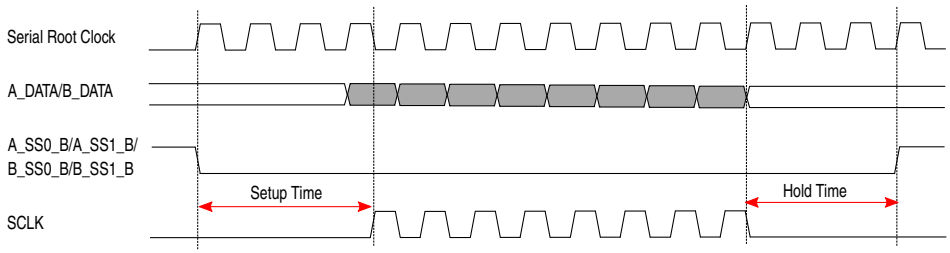


Figure 27-22. Chip selection output timing for SDR sequence

- **Chip Selection timing in DDR sequence**
  For DDR sequence, the delay from chip selection assertion and SCLK rise edge is **(TCSS+0.5)** cycles of serial root clock; The delay from SCLK fall edge and chip selection deassertion is **(TCSH+0.5)** cycles of serial root clock.

NOTE

When AHB RX prefetch is enabled, and prefetch can be aborted, set TCSH to be least 1 to guarantee positive chip select hold time after SCK falling edge.

Following diagram indicates the timing relationship between chip selection and SCLK:

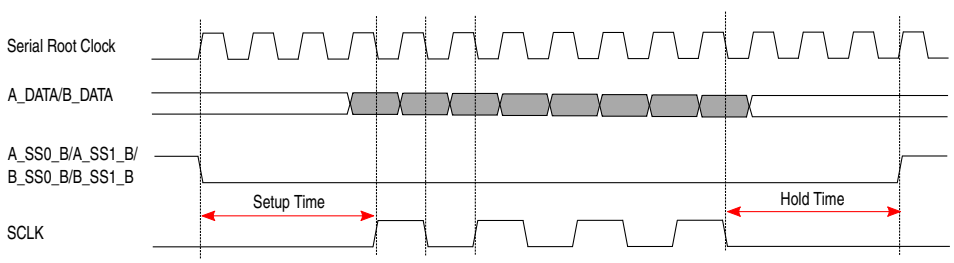


Figure 27-23. Chip selection output timing for DDR sequence

For certain device (such as FPGA device), there is limitation on the interval between Chip Selection valid. FlexSPI will ensure a delay time between chip selection valid if register field FLSHxCR1[CSINTERVAL] is set to non-zero value. The delay time is: CSINTERVAL*1024 cycle of serial root clock no matter SDR or DDR sequence. Please set this register field value to zero if there is no this limitation for external device. Following diagram indicates the timing of chip selection interval.

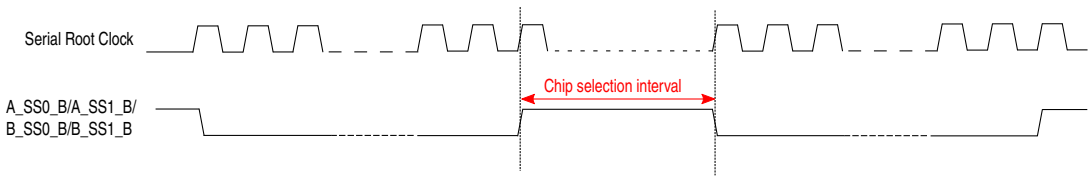


Figure 27-24. Chip Selection Valid interval

27.3.15 FlexSPI Input Timing

This section describes the input timing of FlexSPI.

27.3.15.1 Receiving Block diagram

This section describes the receiving block in FlexSPI.

FlexSPI determines SDR mode or DDR mode by decoding instruction code and determines whether both edge used or not for flash data sampling. FlexSPI samples the data line for Learn instruction and Read instruction.

There are four clock source for the sampling clock and the source selection is configured by register field MCR0[RXCLKSRC]. For more details about DLL configuration, please refer to DLL configuration for sampling

- Internal dummy read strobe and loopbacked internally
- Internal dummy read strobe and loopbacked from DQS pad
- SCK output and loopbacked from SCK pad
- Flash provided read strobe

Following diagram indicates the sampling block in FlexSPI:

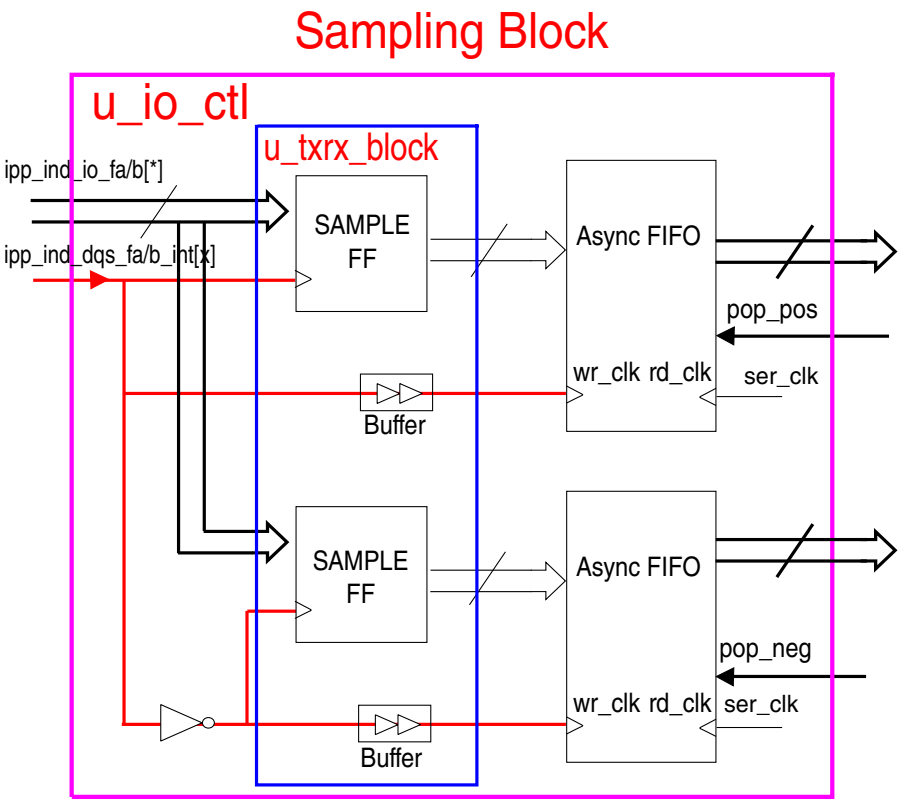


Figure 27-25. Sampling Block in FlexSPI

Following is the RX block diagram for data learning feature.

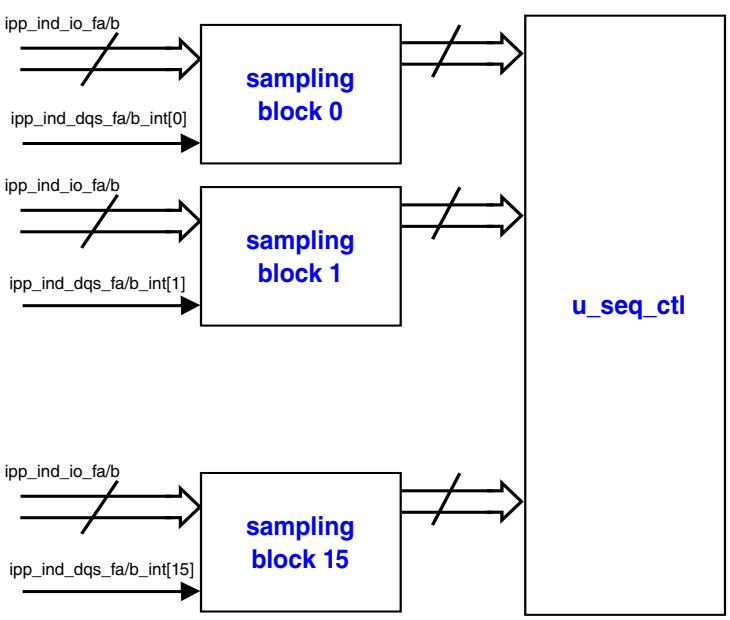


Figure 27-26. Sampling Blocks for Data Learning feature

NOTE

- FlexSPI don't support read with 0 turn around cycle for direction switching on IO pins. This kind of read often happens on QPI configuration register read. For example, in below wavefrom, "QuadSPI configuration register read", IO[3:0] is output for FlexSPI in CMD phase, but is input for FlexSPI in DATA phase. And there is no time left for FlexSPI to switch the direction. This kind of read is not supported. Moreover, FlexSPI and Flash will drive the IO pins at same time, this will cause problem. So, SPI mode or Read with dummy cycles is required to avoid described behavior on IO pins.
- Please notice in below example, cycle number is not accurate, for accurate cycle number, please refer to device's spec.

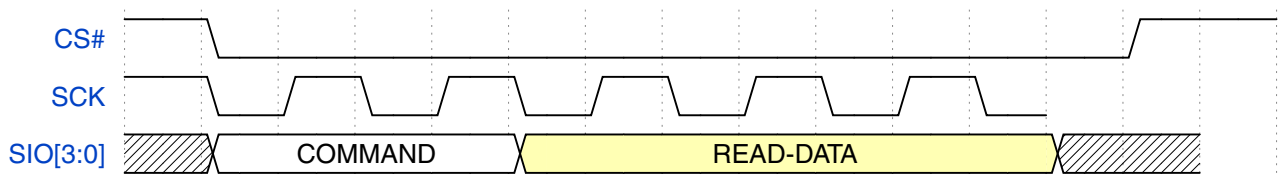


Figure 27-27. Unsupported Read Behavior

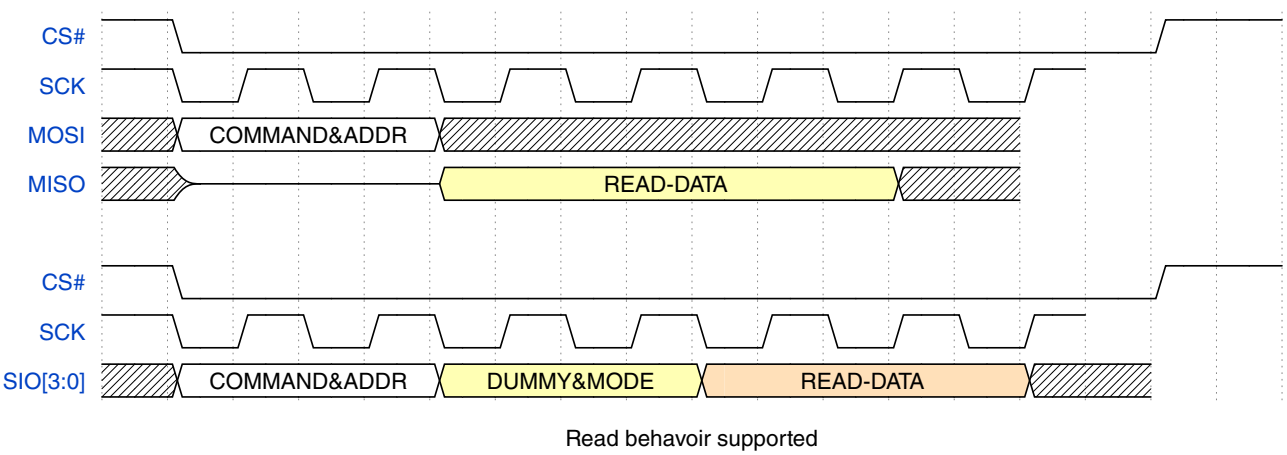


Figure 27-28. Supported Read Behavior

27.3.15.2 RX Clock Source Features

This section describes the features of each RX clock source.

- **Internal dummy read strobe and loopbacked internally(MCR0[RXCLKSRC]==0)**

Supporting legacy device with zero device output hold time.

Saving one pad(DQS pad).

Supporting low frequency clock for boot up usage.

- **Internal dummy read strobe and loopbacked from DQS pad(MCR0[RXCLKSRC]==1)**

  Supporting higher frequency than mode "MCR0[RXCLKSRC]==0".

  Supporting device doesn't provide read strobe.

- **Flash provided read strobe(MCR0[RXCLKSRC]==3)**

  Supporting the highest frequency.

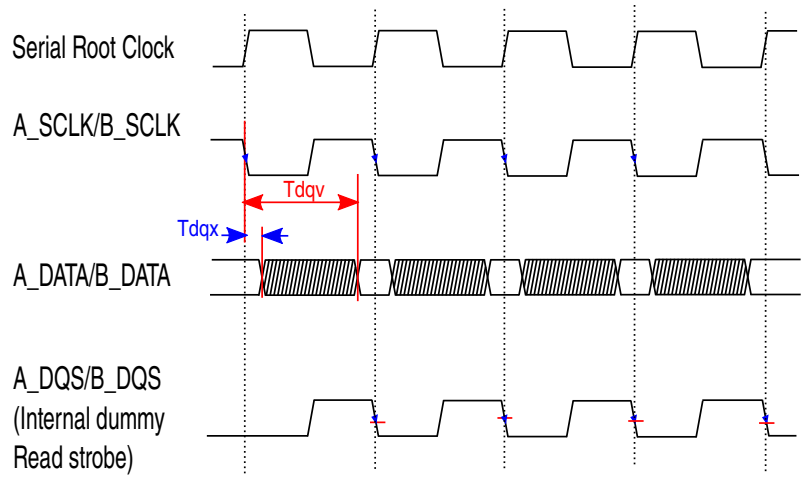
  Supporting device provides read strobe.

27.3.15.3 Input timing for sampling with dummy read strobe

This section describes the input timing when sampling with internal dummy read strobe (MCR0[RXCLKSRC] is set to 0x0 or 0x1). The timing is very similar for sampling with dummy read strobe loopback internally and loopback from pad. But it could achieve higher read frequency by sampling with dummy read strobe loopback from DQS pad because it will compensate the delay of SCLK output path and Data pin input path. The input timing is different for SDR mode and DDR mode.

- **Input timing for sampling with dummy read strobe in SDR mode**

  For SDR Read/Learn instruction, FlexSPI samples input data pins with the falling edge of dummy read strobe. Following diagram indicates the input timing for sampling with dummy read strobe in SDR mode:



Tdqv - setup time for the SCLK to the next valid data.
Tdqx - hold time for the SCLK to the current valid data.

Figure 27-29. Input Timing for sampling with dummy read strobe in SDR mode

- **Input timing for sampling with dummy read strobe in DDR mode**

  For DDR Read/Learn instruction, FlexSPI sample input data pins with both rise and fall edge of dummy read strobe. Following diagram indicates the input timing for sampling with dummy read strobe in DDR mode

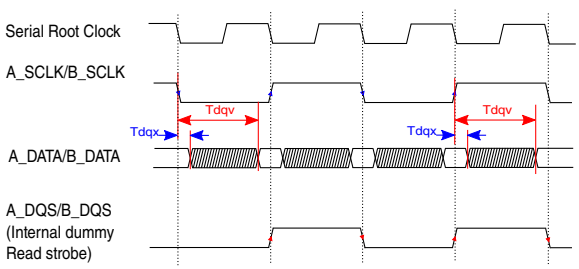


Figure 27-30. Input Timing for sampling with dummy read strobe in DDR mode

27.3.15.4 Input timing for sampling with SCK output loopbacked from SCK pad

This section describes the input timing when sampling with SCK output loopbacked from SCK pad (MCR0[RXCLKSRC] is set to 0x2). The input timing is similar as sampling with dummy read strobe. SCK output is toggling for all type instructions, but internal dummy strobe just toggles for Read and Learn instruction. So FlexSPI will got more data bits if sampling with SCK output. FlexSPI will ignore the redundant data bits sampled automatically.

27.3.15.5 Input timing for sampling with flash provided read strobe

This section describes the input timing when sampling with flash provided read strobe (MCR0[RXCLKSRC] is set to 0x3). The input timing is different for SDR mode and DDR mode.

NOTE

There are no known devices that provide read strobe and support SDR mode operation.

There are two kinds of Flash provided read strobe:

- **Flash provide read strobe with SCLK**

  For certain flash devices, it provides both read data and read strobes with SCLK. Then the read strobe edge is aligned with read data change. FlexSPI controller should delay read strobe by half cycle in serial root clock (with DLL) and then sample read data with delayed strobe. Refer DLL configuration for sampling for more details. Following diagrams indicates the input timing for sampling with flash read strobe in SDR mode and DDR mode:

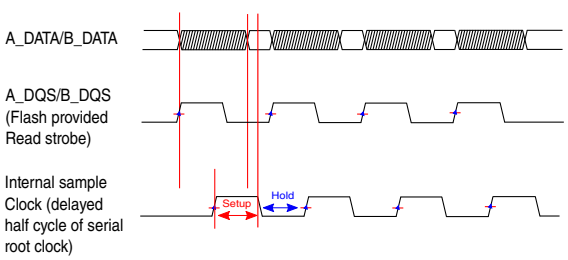


Figure 27-31. Input Timing 2 for Flash provided read strobe in SDR mode

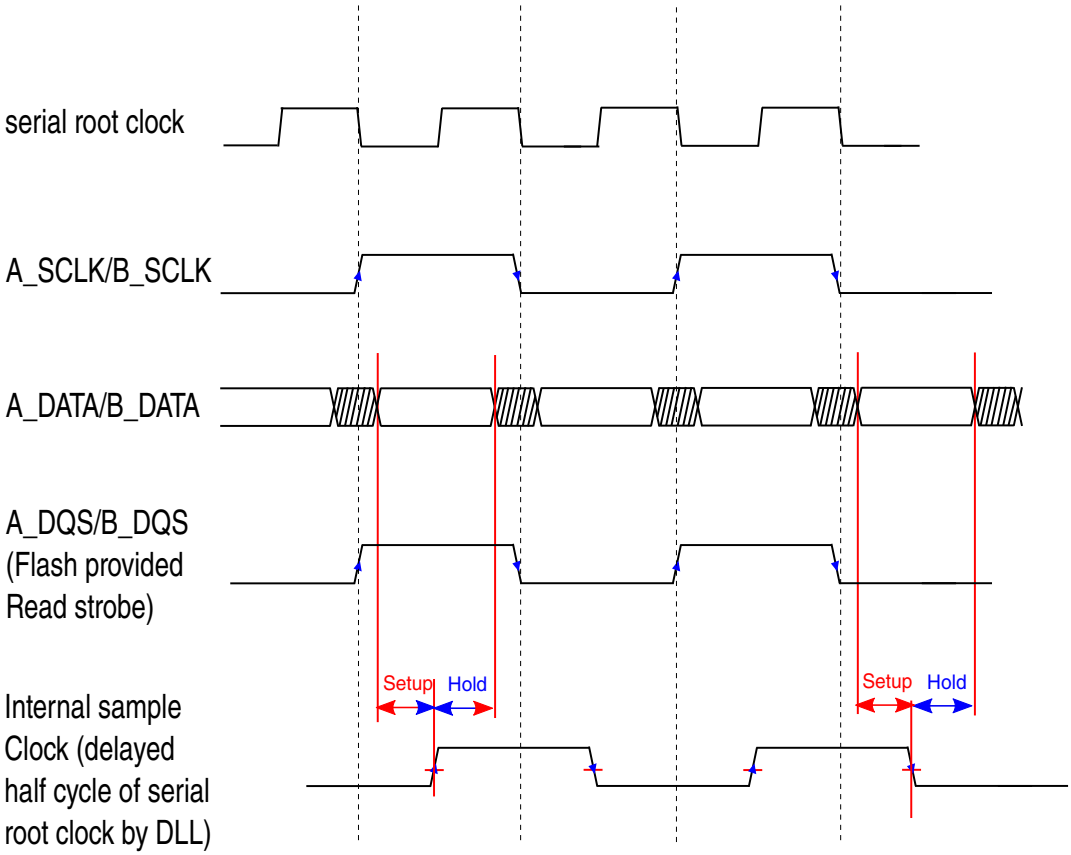


Figure 27-32. Input Timing 2 for Flash provided read strobe in DDR mode

27.3.15.6 DLL configuration for sampling

The input timing is different for those four sampling clock source. This is handled by setting register DLLxCR differently according to the sampling clock source mode. DLL is a delay line chain, which could be set to a fixed number of delay cells or auto-adjusted to lock on a certain phase delay to the reference clock.

- In following cases, DLLxCR should be set 0x00000100 (1 fixed delay cells in DLL delay chain) :
  - Sampling data with Dummy read strobe loopbacked internally(MCR0[RXCLKSRC]=0x0)
  - Sampling data with Dummy read strobe loopbacked from DQS pad(MCR0[RXCLKSRC]=0x1)
- When data is sampled with Flash provided read strobe (MCR0[RXCLKSRC]=0x3) and flash provides read strobe with SCLK, DLL should be set as following to lock on half cycle of the reference clock (serial root clock)
  - SLVDLYTARGET=0xF

- DLLEN=0x1
- OVRDEN=0x0
- Other fields in DLLxCR should be kept as reset value (all zero)

NOTE

If serial root clock is lower than 100 MHz, DLL is unable to lock on half cycle of serial root clock because the delay cell number is limited in delay chain. Then DLL should be configured as following instead:

- OVRDEN=0x1
- OVRDVAL=N; Each delay cell in DLL is about 75 ps~225 ps. The delay of DLL delay chain is (N * Delay_cell_delay), N should be set based on max. DDR frequency that current project supported, N = 21, please notice this is a recommended value. May need to adjust in real application if facing failure.
- Other fields in DLLxCR should be kept as reset value (all zero).

27.3.16 XIP Enhanced Mode

FlexSPI always supports Execute-In-Place (XIP), no matter external device provided XIP enhanced mode or not. Execute-In-Place is supported by putting program code on External device, then read/execute on external device directly by AHB read access to SFM space. There is no configuration or status polling needed during AHB read access to External device memory and AHB RX buffer is fully transparent to software.

Certain devices provide XIP enhanced mode to improve code execution. In this mode, there is no need to provide Command code for Read Sequence. It saves many cycles for Command Instructions and greatly improves code execution. This XIP enhanced mode is entered/exit by a special sequence which is device specified. Please refer to external device datasheet for more detail.

Normally, the XIP enhanced mode is entered by following sequence:

1. Enable XIP enhanced mode in External Flash by IP command
2. Send the first Read Sequence to External Flash device with correct Mode bits. Command code is needed in this Read sequence.
3. Send following Read sequences to External Flash device with correct Mode bits. Command code is not needed in these Read Sequences. But Mode bits should be sent according to Flash specified. Otherwise Flash will exit Execute-In-Place Enhanced mode.

The instruction JMP_ON_CS in FlexSPI should be used to support external device XIP enhanced mode. This instruction is only allowed in AHB Read command, otherwise there will be IPCMDERR or AHBCDMERR error interrupt generated if this interrupt is enabled. JMP_ON_CS should never be used if device doesn't support XIP enhanced mode. To support XIP enhanced mode, the first instruction in the Read Sequence should be Command instruction and the last valid instruction should be JMP_ON_CS (with operand 0x1).

For the first AHB read Command triggered, FlexSPI will execute the instructions from the instruction pointer 0 in the sequence (which is Command instruction). After this sequence executed FlexSPI will save operand in JMP_ON_CS instruction as start pointer for next Command to current device internally. For the following AHB read Command triggered, FlexSPI will execute from instruction pointer 0x1 and Command instruction is bypassed.

Once XIP enhanced mode has been started it is not possible to access flash by other commands like any program commands. Following diagram indicates XIP operation with Flash XIP Enhanced mode:

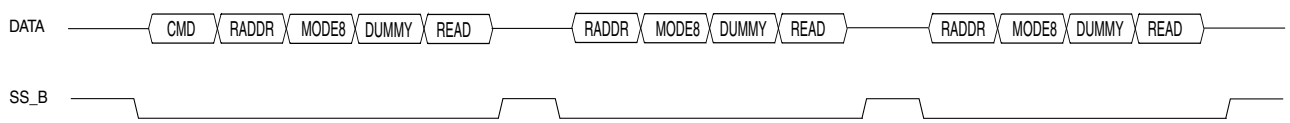


Figure 27-33. XIP Enhanced Mode operation

27.4 External Signals

This section provides the external signal information of the FlexSPI module.

Table 27-7. External Signal List

Signal Name	Function	Direction	Description
A_SS0_B	Peripheral Chip Select Flash A1	O	This signal is the chip select for the serial flash device A1. Port name: PCSA1
A_SS1_B	Peripheral Chip Select Flash A2	O	This signal is the chip select for the serial flash device A2. Port name: PCSA2
B_SS0_B	Peripheral Chip Select Flash B1	O	This signal is the chip select for the serial flash device B1. Port name: PCSB1
B_SS1_B	Peripheral Chip Select Flash B2	O	This signal is the chip select for the serial flash device B2.

Table continues on the next page...

Table 27-7. External Signal List (continued)

Signal Name	Function	Direction	Description
			Port name: PCSB2
A_SCLK	Serial Clock Flash A	O	This signal is the serial clock output to the serial flash device A. Half clock frequency of serial clock root in DDR mode, and same frequency as serial clock root in SDR mode. Clock output toggles during the whole flash access sequence. Port name: SCKA
B_SCLK	Serial Clock Flash B	O	This signal is the serial clock output to the serial flash device B. Half clock frequency of serial clock root in DDR mode, and same frequency as serial clock root in SDR mode. Clock output toggles during the whole flash access sequence. **NOTE:** • B_SCLK may be optionally used as A_SCLK's differential clock output by setting Register Field MCR2[SCKBDIFFOPT]=1'b1 Port name: SCKB
A_DATAn	Serial I/O Flash A	I/O	These signals are the data I/O lines to/from the serial flash device A. Port name: SIOAn
B_DATAn	Serial I/O Flash B	I/O	These signals are the data I/O lines to/from the serial flash device B. Port name: SIOBn
A_DQS	Data Strobe signal Flash A	I/O	Data strobe signal for port A. There are three functions for this signal: • Driven with Read Strobe by external device: Some flash devices provide the Read Strobe signal together with Read data. In this case, this pad may need a **pull down** resistor if external device drives this pad only when reading flash data. • Driven with Latency Information by external device: Certain devices use this pin to indicate the dummy cycles needed (before Program/ Read data transfer). • Loopback dummy read strobe: FlexSPI controller provides internal dummy read strobe for flash read data. Higher read frequency can be achieved by looping back this dummy read strobe from pad. This pin

Table continues on the next page...

Table 27-7. External Signal List (continued)

Signal Name	Function	Direction	Description
			can be floated or put some cap loads on board level to compensate DATA/SCLK pins load. Port name: DQSA
B_DQS	Data Strobe signal Flash B	I/O	Similar to A_DQS. Port name: DQSB

27.5 Initialization

FlexSPI controller initialization sequence is as following:

- Enable controller clocks (AHB clock/IP Bus clock/Serial root clock) in System level.
- Set MCR0[MDIS] to 0x1 (Make sure controller is configured in module stop mode)
- Configure module control registers: MCR0, MCR1, MCR2. (Don't change MCR0[MDIS])
- Configure AHB bus control register (AHBCR) and AHB RX Buffer control register (AHBRXBUFxCR0) optionally, if AHB command will be used
- Configure Flash control registers (FLSHxCR0,FLSHxCR1,FLSHxCR2) according to external device type
- Configure DLL control register (DLLxCR) according to sample clock source selection
- set MCR0[MDIS] to 0x0 (Exit module stop mode)
- Configure LUT as needed (For AHB command or IP command)
- Reset controller optionally (by set MCR0[SWRESET] to 0x1)

External device needs configuration by IP command normally after controller initialization. For example, the device configuration is done by WRITE STATUS command for most serial NOR Flash.

27.6 Application information

This section describes applications supported by the FlexSPI module.

27.6.1 Overview of Error Flags

The following table gives an overview of error category, flags and triggered source.

Table 27-8. Error category and flags in FlexSPI

Error Category	Triggered Source	Description	Error Flags
Command grant error	AHB write command	Command grant timeout	INTR[AHBCMDGE] will be set AHB bus error response
	AHB read command		INTR[AHBCMDGE] will be set AHB bus error response
	IP command		INTR[IPCMDGE] will be set
Command check error	AHB write command	• AHB write command with JMP_ON_CS instruction used in the sequence • There is unknown instruction opcode in the sequence. • Instruction DUMMY_SDR/ DUMMY_RWDS_SDR used in DDR sequence. • Instruction DUMMY_DDR/ DUMMY_RWDS_DDR used in SDR sequence.	INTR[AHBCMDERR] will be set Command is not executed when error detected in command check
	AHB read command	• There is unknown instruction opcode in the sequence. • Instruction DUMMY_SDR/ DUMMY_RWDS_SDR used in DDR sequence. • Instruction DUMMY_DDR/ DUMMY_RWDS_DDR used in SDR sequence.	INTR[AHBCMDERR] will be set Command is not executed when error detected in command check
	IP command	• IP command with JMP_ON_CS instruction used in the sequence • There is unknown instruction opcode in the sequence. • Instruction DUMMY_SDR/ DUMMY_RWDS_SDR used in DDR sequence. • Instruction DUMMY_DDR/ DUMMY_RWDS_DDR used in SDR sequence. • Flash boundary across.	INTR[IPCMDERR] will be set Command is not executed when error detected in command check
Command execution error	AHB write command	Command timeout during execution	INTR[AHBCMDERR] will be set INTR[SEQTIMEOUT] will be set There will be AHB bus error response except following case: • AHB write command is triggered by flush (INCR burst

Table continues on the next page...

Table 27-8. Error category and flags in FlexSPI (continued)

Error Category	Triggered Source	Description	Error Flags
			ended with AHB_TX_BUF not empty) • AHB bufferable write access and bufferable enabled (AHBCR[BUFFERABLEEN]=0x1)
	AHB read command		INTR[AHBCMDERR] will be set INTR[SEQTIMEOUT] will be set There will be AHB bus error response
	IP command		INTR[IPCMDERR] will be set INTR[SEQTIMEOUT] will be set
AHB Bus timeout	AHB write command	AHB bus timeout (no bus ready return)	INTR[AHBBUSTIMEOUT] will be set There will be AHB bus error response
	AHB read command		

NOTE

- flash_top_address is the top address of currently accessed flash

27.6.2 Application on Serial NOR Flash device

This section provides the example sequences for serial NOR flash device (Cypress Flash S25FS128S).

27.6.2.1 Write Enable command

The following table shows WRITE ENABLE command sequence.

Table 27-9. WRITE ENABLE command

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0x0	0x06	command name:WREN
1~7	STOP (0x00)	0x0	0x00	

27.6.2.2 Write Registers command

The following table shows Write Registers command sequence.

Table 27-10. Write Registers command

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0x0 (Single)	0x01	command name:WRR
1	WRITE_SDR	0x0 (Single)	0x1 or 0x2	Data size is 1 or 2byte. Byte 0 write data for Status Register 1; Byte 1 write data for Configuration Register 1 This value could be overridden by IPCR1[IDATSZ].
2~7	STOP (0x00)	0x0	0x00	

27.6.2.3 Page Program command

The following table shows Page Program command sequence.

Table 27-11. PAGE PROGRAM command (Cypress serial NOR flash)

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0x0 (Single)	0x02 or 0x12	command name: PP or 4PP PP is 3-Byte address mode. 4PP is 4-Byte address mode
1	ADDR_SDR	0x0 (Single)	0x18 or 0x20	Address bit number (operand value) should be 24 in 3-Byte address mode and 32 in 4-Byte address mode.
2	WRITE_SDR	0x0 (Single)	Any non-zero value	This operand value could be used as default programming data size if IPCR1[IDATSZ] is zero. This value is ignored for AHB command.
3~7	STOP (0x00)	0x0	0x00	

The following table shows Page Program command sequence (QPI mode).

Table 27-12. PAGE PROGRAM (QPI mode) command (Cypress serial NOR flash)

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0x2 (Quad)	0x02 or 0x12	command name: PP or 4PP PP is 3-Byte address mode. 4PP is 4-Byte address mode
1	ADDR_SDR	0x2 (Quad)	0x18 or 0x20	Address bit number (operand value) should be 24 in 3-Byte address mode and 32 in 4-Byte address mode.
2	WRITE_SDR	0x2 (Quad)	Any non-zero value	This operand value could be used as default programming data size if IPCR1[IDATSZ] is zero. This value is ignored for AHB command.
3~7	STOP (0x00)	0x0	0x00	

27.6.2.4 Read Status 1 command

The following table shows READ STATUS 1 command sequence.

Table 27-13. READ STATUS 1 command

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0x0 (Single)	0x05	command name:RDSR1
1	READ_SDR	0x0 (Single)	0x1	1 Byte for Status register 1
2~7	STOP (0x00)	0x0	0x00	

27.6.2.5 Read command

The following table shows READ command sequence.

Table 27-14. READ command

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0x0 (Single)	0x03 or 0x13	command name:READ or 4READ READ is 3-Byte address mode. 4READ is 4-Byte address mode

Table continues on the next page...

Table 27-14. READ command (continued)

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
1	ADDR_SDR	0x0 (Single)	0x18 or 0x20	Address bit number (operand value) should be 24 in 3-Byte address mode and 32 in 4-Byte address mode.
2	READ_SDR	0x0 (Single)	Any non-zero value	This operand value could be used as default reading data size if IPCR1[IDATSZ] is zero. This value is ignored for AHB command.
3~7	STOP (0x00)	0x0	0x00	

27.6.2.6 Fast Read command

The following table shows Fast Read command sequence (In the case of Cypress SPI Configuration Register bits CR2V[7]=0, CR2V[3:0]=0x8).

Table 27-15. FAST_READ command

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0x0 (Single)	0x0B or 0x0C	command name:FAST_READ or 4FAST_READ FAST_READ is 3-Byte address mode. 4FAST_READ is 4-Byte address mode
1	ADDR_SDR	0x0 (Single)	0x18 or 0x20	Address bit number (operand value) should be 24 in 3-Byte address mode and 32 in 4-Byte address mode.
2	DUMMY_SDR	0x0 (Single)	0x08	Dummy cycle is 8 (in serial root clock) as CR2V[3:0]=0x8.
3	READ_SDR	0x0 (Single)	Any non-zero value	This operand value could be used as default reading data size if IPCR1[IDATSZ] is zero. This value is ignored for AHB command.
4~7	STOP (0x00)	0x0	0x00	

27.6.2.7 Dual IO Fast Read command

The following table shows Dual IO FAST_READ command sequence (In the case of Cypress SPI Configuration Register bits CR2V[7]=0, CR2V[3:0]=0x8, Continuous Read mode).

Table 27-16. Dual IO FAST_READ command (Continuous Read mode)

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0x0 (Single)	0xBB or 0xBC	command name:DIOR or 4DIOR DIOR is 3-Byte address mode. 4DIOR is 4-Byte address mode
1	ADDR_SDR	0x1 (Dual)	0x18 or 0x20	Address bit number (operand value) should be 24 in 3-Byte address mode and 32 in 4-Byte address mode.
2	MODE8_SDR	0x1 (Dual)	0xAx	Enter continuous read mode or keep in continuous read mode.
3	DUMMY_SDR	0x1 (Dual)	0x08	Dummy cycle is 8 (in serial root clock) as CR2V[3:0]=0x8.
4	READ_SDR	0x1 (Dual)	Any non-zero value	This operand value could be used as default reading data size if IPCR1[IDATSZ] is zero. This value is ignored for AHB command.
5	JMP_ON_CS	0x0 (or Dont care)	0x01	The CMD instruction will be bypassed after the first read access.
6~7	STOP (0x00)	0x0	0x00	

The following table shows Dual IO FAST_READ command sequence (In the case of Cypress SPI Configuration Register bits CR2V[7]=0, CR2V[3:0]=0x8, Non-continuous Read mode).

Table 27-17. Dual IO FAST_READ command (Non-continuous Read mode)

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0x0 (Single)	0xBB or 0xBC	command name:DIOR or 4DIOR DIOR is 3-Byte address mode. 4DIOR is 4-Byte address mode
1	ADDR_SDR	0x1 (Dual)	0x18 or 0x20	Address bit number (operand value) should be 24 in 3-Byte address mode and 32 in 4-Byte address mode.
2	MODE8_SDR	0x1 (Dual)	Any value other than 0xAx	Exit continuous read mode or keep in non-continuous read mode.
3	DUMMY_SDR	0x1 (Dual)	0x08	Dummy cycle is 8 (in serial root clock) as CR2V[3:0]=0x8.
4	READ_SDR	0x1 (Dual)	Any non-zero value	This operand value could be used as default reading data size if IPCR1[IDATSZ] is zero. This value is ignored for AHB command.
5~7	STOP (0x00)	0x0	0x00	

27.6.2.8 Quad IO Fast Read command

The following table shows Quad IO FAST_READ command sequence (In the case of Cypress SPI Configuration Register bits CR2V[7]=0, CR2V[6]=0, CR2V[3:0]=0x8, Non-QPI mode, Non-continuous read mode).

Table 27-18. Quad IO FAST_READ command (Non-QPI mode, Non-continuous read mode)

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0x0 (Single)	0xEB or 0xEC	command name:QIOR or 4QIOR QIOR is 3-Byte address mode. 4QIOR is 4-Byte address mode
1	ADDR_SDR	0x2 (Quad)	0x18 or 0x20	Address bit number (operand value) should be 24 in 3-Byte address mode and 32 in 4-Byte address mode.
2	MODE8_SDR	0x2 (Quad)	Any value other than 0xAx	Exit continuous read mode or keep in non-continuous read mode.
3	DUMMY_SDR	0x2 (Quad)	0x08	Dummy cycle is 8 (in serial root clock) as CR2V[3:0]=0x8.
4	READ_SDR	0x2 (Quad)	Any non-zero value	This operand value could be used as default reading data size if IPCR1[IDATSZ] is zero. This value is ignored for AHB command.
5~7	STOP (0x00)	0x0	0x00	

The following table shows Quad IO FAST_READ command sequence (In the case of Cypress SPI Configuration Register bits CR2V[7]=0, CR2V[6]=0, CR2V[3:0]=0x8, Non-QPI mode, Continuous read mode).

Table 27-19. Quad IO FAST_READ command (Non-QPI mode, Continuous read mode)

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0x0 (Single)	0xEB or 0xEC	command name:QIOR or 4QIOR QIOR is 3-Byte address mode. 4QIOR is 4-Byte address mode
1	ADDR_SDR	0x2 (Quad)	0x18 or 0x20	Address bit number (operand value) should be 24 in 3-Byte address mode and 32 in 4-Byte address mode.
2	MODE8_SDR	0x2 (Quad)	0xA0	Enter continuous read mode or keep in continuous read mode.
3	DUMMY_SDR	0x2 (Quad)	0x08	Dummy cycle is 8 (in serial root clock) as CR2V[3:0]=0x8.
4	READ_SDR	0x2 (Quad)	Any non-zero value	This operand value could be used as default reading data size if IPCR1[IDATSZ] is zero. This value is ignored for AHB command.

Table continues on the next page...

Table 27-19. Quad IO FAST_READ command (Non-QPI mode, Continuous read mode) (continued)

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
5	JMP_ON_CS	0x0 (or Dont care)	0x01	The CMD instruction will be bypassed after the first read access.
6~7	STOP (0x00)	0x0	0x00	

The following table shows Quad IO FAST_READ command sequence (In the case of Cypress SPI Configuration Register bits CR2V[7]=0, CR2V[6]=1, CR2V[3:0]=0x8, QPI mode, Continuous read mode).

Table 27-20. Quad IO FAST_READ command (QPI mode, Continuous read mode)

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0x2 (Quad)	0xEB or 0xEC	command name:QIOR or 4QIOR QIOR is 3-Byte address mode. 4QIOR is 4-Byte address mode
1	ADDR_SDR	0x2 (Quad)	0x18 or 0x20	Address bit number (operand value) should be 24 in 3-Byte address mode and 32 in 4-Byte address mode.
2	MODE8_SDR	0x2 (Quad)	0xA0	Enter continuous read mode or keep in continuous read mode.
3	DUMMY_SDR	0x2 (Quad)	0x08	Dummy cycle is 8 (in serial root clock) as CR2V[3:0]=0x8.
4	READ_SDR	0x2 (Quad)	Any non-zero value	This operand value could be used as default reading data size if IPCR1[IDATSZ] is zero. This value is ignored for AHB command.
5	JMP_ON_CS	0x0 (or Dont care)	0x01	The CMD instruction will be bypassed after the first read access.
6~7	STOP (0x00)	0x0	0x00	

27.6.2.9 DDR Quad IO Fast Read command

The following table shows DDR Quad IO FAST_READ command sequence (In the case of Cypress SPI Configuration Register bits CR2V[7]=0, CR2V[6]=0, CR2V[3:0]=0x8, Non-QPI mode, Non-continuous read mode).

Table 27-21. DDR Quad IO FAST_READ command (Non-QPI mode, Non-continuous read mode)

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0x0 (Single)	0xED or 0xEE	command name:QIOR_DDR or 4QIOR_DDR QIOR_DDR is 3-Byte address mode. 4QIOR_DDR is 4-Byte address mode
1	ADDR_DDR	0x2 (Quad)	0x18 or 0x20	Address bit number (operand value) should be 24 in 3-Byte address mode and 32 in 4-Byte address mode.
2	MODE8_DDR	0x2 (Quad)	Any value other than 0xAx	Exit continuous read mode or keep in non-continuous read mode.
3	DUMMY_DDR	0x2 (Quad)	0x08	Dummy cycle is 8 (in serial root clock) as CR2V[3:0]=0x8.
4	LEARN_DDR	0x2 (Quad)	0x1	DLP (Data Learning Pattern is 8 bits).
5	READ_DDR	0x2 (Quad)	Any non-zero value	This operand value could be used as default reading data size if IPCR1[IDATSZ] is zero. This value is ignored for AHB command.
6~7	STOP (0x00)	0x0	0x00	

The following table shows DDR Quad IO FAST_READ command sequence (In the case of Cypress SPI Configuration Register bits CR2V[7]=0, CR2V[6]=0, CR2V[3:0]=0x8, Non-QPI mode, Continuous read mode).

Table 27-22. DDR Quad IO FAST_READ command (Non-QPI mode, Continuous read mode)

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0x0 (Single)	0xED or 0xEE	command name:QIOR_DDR or 4QIOR_DDR QIOR_DDR is 3-Byte address mode. 4QIOR_DDR is 4-Byte address mode
1	ADDR_DDR	0x2 (Quad)	0x18 or 0x20	Address bit number (operand value) should be 24 in 3-Byte address mode and 32 in 4-Byte address mode.
2	MODE8_DDR	0x2 (Quad)	0xA0	Enter continuous read mode or keep in continuous read mode.
3	DUMMY_DDR	0x2 (Quad)	0x08	Dummy cycle is 8 (in serial root clock) as CR2V[3:0]=0x8.
4	LEARN_DDR	0x2 (Quad)	0x1	DLP (Data Learning Pattern is 8 bits).
5	READ_DDR	0x2 (Quad)	Any non-zero value	This operand value could be used as default reading data size if IPCR1[IDATSZ] is zero. This value is ignored for AHB command.

Table continues on the next page...

Table 27-22. DDR Quad IO FAST_READ command (Non-QPI mode, Continuous read mode) (continued)

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
6	JMP_ON_CS	0x0 (or Dont care)	0x01	The CMD instruction will be bypassed after the first read access.
7	STOP (0x00)	0x0	0x00	

The following table shows DDR Quad IO FAST_READ command sequence (In the case of Cypress SPI Configuration Register bits CR2V[7]=0, CR2V[6]=1, CR2V[3:0]=0x8, QPI mode, Continuous read mode).

Table 27-23. DDR Quad IO FAST_READ command (QPI mode, Continuous read mode)

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0x2 (Quad)	0xED or 0xEE	command name:QIOR_DDR or 4QIOR_DDR QIOR_DDR is 3-Byte address mode. 4QIOR_DDR is 4-Byte address mode
1	ADDR_DDR	0x2 (Quad)	0x18 or 0x20	Address bit number (operand value) should be 24 in 3-Byte address mode and 32 in 4-Byte address mode.
2	MODE8_DDR	0x2 (Quad)	0xA0	Enter continuous read mode or keep in continuous read mode.
3	DUMMY_DDR	0x2 (Quad)	0x08	Dummy cycle is 8 (in serial root clock) as CR2V[3:0]=0x8.
4	LEARN_DDR	0x2 (Quad)	0x1	DLP (Data Learning Pattern is 8 bits).
5	READ_DDR	0x2 (Quad)	Any non-zero value	This operand value could be used as default reading data size if IPCR1[IDATSZ] is zero. This value is ignored for AHB command.
6	JMP_ON_CS	0x0 (or Dont care)	0x01	The CMD instruction will be bypassed after the first read access.
7	STOP (0x00)	0x0	0x00	

27.6.3 Application on FPGA device

FPGA device should be accessed by AHB command. All AHB accesses to FPGA will be transparent to SW driver (no SW intervention).There may be some special requirements from FPGA device.

1. **Device type may be different on A1/A2/B1/B2**

For this case, clear the MCR2[SAMEDEVICEEN] bit and configure FLSHxCR0 and FLSHxCR1 register separately for up to four external devices.

2. **Device needs different wait cycle for Programming.**

   The AHB write wait cycle number could be set separately for these four external devices (by register field FLSHxCR2[AWRWAIT]). Software could configure the sequences in LUT with different DUMMY instructions (operand will determine dummy cycle). Note that FlexSPI will hold AHB bus ready for this wait time, so AHB Bus performance may become very low when this wait time is very long.

3. **Device needs different wait cycle for Reading.**

   The AHB Read Sequence index and Sequence Number could be set separately for these four external devices (by register field FLSHxCR2[ARDSEQID] and FLSHxCR2[ARDSEQNUM]). Software could configure the sequences in LUT with different DUMMY instruction (operand will determine dummy cycle).

4. **Device may be sensitive to read instruction clock cycle number**

   Device will be sensitive to read instruction clock cycle number if its internal memory is implemented similar as FIFO. For this case, software could send the data size information to external device by DATSZ instruction. FPGA device should decode the data size information and determine how much data bytes should be popped.

5. **Device may needs interval time between Chip selection valid**

   This could be handled by register field FLSHxCR1[CSINTERVAL] setting.

6. **Device may use SCLK as reference clock for its internal PLL**

   In this case, SCLK should be free-running and clock frequency should be stable. This could be achieved by setting MCR0[SCKFREERUNEN] and use SDR sequence only.

27.7 Memory Map and register definition

This section includes the FlexSPI module memory map and detailed descriptions of all registers.

27.7.1 Register Access

All registers can be accessed with 8-bit, 16-bit, and 32-bit width operations. Never change the setting value of reserved fields in control registers. Changing the value of reserved fields may impact the normal functioning of the controller.

27.7.2 FlexSPI register descriptions

27.7.2.1 FlexSPI memory map

FlexSPI base address: 402A_8000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Module Control Register 0 (MCR0)	32	RW	FFFF_80C2h
4h	Module Control Register 1 (MCR1)	32	RW	FFFF_FFFFh
8h	Module Control Register 2 (MCR2)	32	RW	2000_81F7h
Ch	AHB Bus Control Register (AHBCR)	32	RW	0000_0018h
10h	Interrupt Enable Register (INTEN)	32	RW	0000_0000h
14h	Interrupt Register (INTR)	32	W1C	0000_0000h
18h	LUT Key Register (LUTKEY)	32	RW	5AF0_5AF0h
1Ch	LUT Control Register (LUTCR)	32	RW	0000_0002h
20h	AHB RX Buffer 0 Control Register 0 (AHBRXBUF0CR0)	32	RW	8000_0020h
24h	AHB RX Buffer 1 Control Register 0 (AHBRXBUF1CR0)	32	RW	8001_0020h
28h	AHB RX Buffer 2 Control Register 0 (AHBRXBUF2CR0)	32	RW	8002_0020h
2Ch	AHB RX Buffer 3 Control Register 0 (AHBRXBUF3CR0)	32	RW	8003_0020h
60h	Flash Control Register 0 (FLSHA1CR0)	32	RW	0001_0000h
64h	Flash Control Register 0 (FLSHA2CR0)	32	RW	0001_0000h
68h	Flash Control Register 0 (FLSHB1CR0)	32	RW	0001_0000h
6Ch	Flash Control Register 0 (FLSHB2CR0)	32	RW	0001_0000h
70h	Flash Control Register 1 (FLSHA1CR1)	32	RW	0000_0063h
74h	Flash Control Register 1 (FLSHA2CR1)	32	RW	0000_0063h
78h	Flash Control Register 1 (FLSHB1CR1)	32	RW	0000_0063h
7Ch	Flash Control Register 1 (FLSHB2CR1)	32	RW	0000_0063h
80h	Flash Control Register 2 (FLSHA1CR2)	32	RW	0000_0000h
84h	Flash Control Register 2 (FLSHA2CR2)	32	RW	0000_0000h
88h	Flash Control Register 2 (FLSHB1CR2)	32	RW	0000_0000h
8Ch	Flash Control Register 2 (FLSHB2CR2)	32	RW	0000_0000h
94h	Flash Control Register 4 (FLSHCR4)	32	RW	0000_0000h

Table continues on the next page...

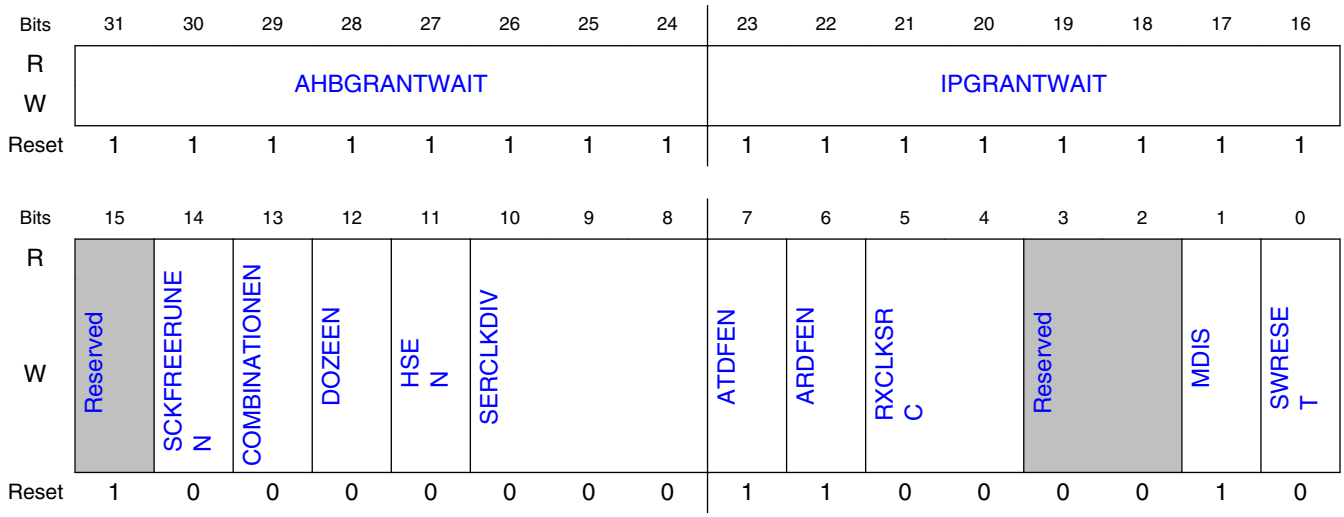
Offset	Register	Width (In bits)	Access	Reset value
A0h	IP Control Register 0 (IPCR0)	32	RW	0000_0000h
A4h	IP Control Register 1 (IPCR1)	32	RW	0000_0000h
B0h	IP Command Register (IPCMD)	32	RW	0000_0000h
B8h	IP RX FIFO Control Register (IPRXFCR)	32	RW	0000_0000h
BCh	IP TX FIFO Control Register (IPTXFCR)	32	RW	0000_0000h
C0h	DLL Control Register 0 (DLLACR)	32	RW	0000_0100h
C4h	DLL Control Register 0 (DLLBCR)	32	RW	0000_0100h
E0h	Status Register 0 (STS0)	32	RO	0000_0002h
E4h	Status Register 1 (STS1)	32	RO	0000_0000h
E8h	Status Register 2 (STS2)	32	RO	0100_0100h
ECh	AHB Suspend Status Register (AHBSPNDSTS)	32	RO	0000_0000h
F0h	IP RX FIFO Status Register (IPRXFSTS)	32	RO	0000_0000h
F4h	IP TX FIFO Status Register (IPTXFSTS)	32	RO	0000_0000h
100h - 17Ch	IP RX FIFO Data Register a (RFDR0 - RFDR31)	32	RO	0000_0000h
180h - 1FCh	IP TX FIFO Data Register a (TFDR0 - TFDR31)	32	WO	0000_0000h
200h - 2FCh	LUT a (LUT0 - LUT63)	32	RW	Table 27-23

27.7.2.2 Module Control Register 0 (MCR0)

27.7.2.2.1 Offset

Register	Offset
MCR0	0h

27.7.2.2.2 Diagram



27.7.2.2.3 Fields

Field	Function
31-24 AHBGRANTWAIT	Timeout wait cycle for AHB command grant. If AHB Triggered Command is not granted by arbitrator, it will timeout after AHBGRANTTIMEOUT * 1024 AHB Clock cycles. This grant timeout may occur when the pending command sequence is IP triggered and the read/write data size is too large. When an AHB command grant time out occurs, there will be an interrupt generated (INTR[AHBCMDGE]) if this interrupt is enabled (INTEN[AHBCMDGEEN] is set) and AHB command is ignored by arbitrator. **NOTE:** This field is for debug only, please keep default value! It is not allowed to set this field to value 0x0.
23-16 IPGRANTWAIT	Time out wait cycle for IP command grant. If IP Triggered Command is not granted by arbitrator, it will timeout after IPGRANTTIMEOUT * 1024 AHB Clock cycles. This grant timeout maybe occur when pending command sequence is AHB triggered and read/write data size is too large. When IP command grant time out occurs, there will be an interrupt generated (INTR[IPCMDGE]) if this interrupt is enabled (INTEN[IPCMDGEEN] is set 0x1) and IP command is ignored by arbitrator. **NOTE:** This field is for debug only, please keep default value! It is not allowed to set this field to value 0x0.
15 —	Reserved
14 SCKFREERUNEN	This bit is used to force SCLK output free-running. For FPGA applications, external device may use SCLK as reference clock to its internal PLL. If SCLK free-running is enabled, data sampling with loopback clock from SCLK pad is not supported (MCR0[RXCLKSRC]=2). 0b - Disable. 1b - Enable.
13 COMBINATIONEN	This bit is to support Flash Octal mode access by combining Port A and B Data pins (A_DATA[3:0] and B_DATA[3:0]), when Port A and Port B are of 4 bit data width.

Table continues on the next page...

Field	Function
	NOTE: Combination mode is not supported if Port A and Port B are 8 bit data width. This bit should be set to zero in this case. 0b - Disable. 1b - Enable.
12 DOZEEN	Doze mode enable bit 0b - Doze mode support disabled. AHB clock and serial clock will not be gated off when there is doze mode request from system. 1b - Doze mode support enabled. AHB clock and serial clock will be gated off when there is doze mode request from system.
11 HSEN	Half Speed Serial Flash access Enable. This bit enables the divide by 2 of the clock to external serial flash devices (A_SCLK/B_SCLK) for all commands (for both SDR and DDR mode). FlexSPI need to be set into MDIS mode before changing value of HSEN. Otherwise, it is possible to cause issue on internal logic/state machine. 0b - Disable divide by 2 of serial flash clock for half speed commands. 1b - Enable divide by 2 of serial flash clock for half speed commands.
10-8 SERCLKDIV	The serial root clock could be divided inside FlexSPI . See Clocks section for more details on clocking. **NOTE:** Don't change this field during IP's normal operation mode. Alter the value after putting IP into stop mode. 000b - Divided by 1 001b - Divided by 2 010b - Divided by 3 011b - Divided by 4 100b - Divided by 5 101b - Divided by 6 110b - Divided by 7 111b - Divided by 8
7 ATDFEN	Enable AHB bus Write Access to IP TX FIFO. 0b - IP TX FIFO should be written by IP Bus. AHB Bus write access to IP TX FIFO memory space will get bus error response. 1b - IP TX FIFO should be written by AHB Bus. IP Bus write access to IP TX FIFO memory space will be ignored but no bus error response.
6 ARDFEN	Enable AHB bus Read Access to IP RX FIFO. 0b - IP RX FIFO should be read by IP Bus. AHB Bus read access to IP RX FIFO memory space will get bus error response. 1b - IP RX FIFO should be read by AHB Bus. IP Bus read access to IP RX FIFO memory space will always return data zero but no bus error response.
5-4 RXCLKSRC	Sample Clock source selection for Flash Reading Refer RX Clock Source Features for more details 00b - Dummy Read strobe generated by FlexSPI Controller and loopback internally. 01b - Dummy Read strobe generated by FlexSPI Controller and loopback from DQS pad. 10b - Reserved 11b - Flash provided Read strobe and input from DQS pad
3-2 —	Reserved
1 MDIS	Module Disable When module disabled, AHB/serial clock will be gated off internally to save power. Only register access (except LUT/IP RX FIFO/IP TX FIFO) is allowed.
0 SWRESET	Software Reset This bit is auto-cleared by hardware after software reset done. Configuration registers will not be reset.

27.7.2.3 Module Control Register 1 (MCR1)

27.7.2.3.1 Offset

Register	Offset
MCR1	4h

27.7.2.3.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SEQWAIT															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	AHBBUSWAIT															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

27.7.2.3.3 Fields

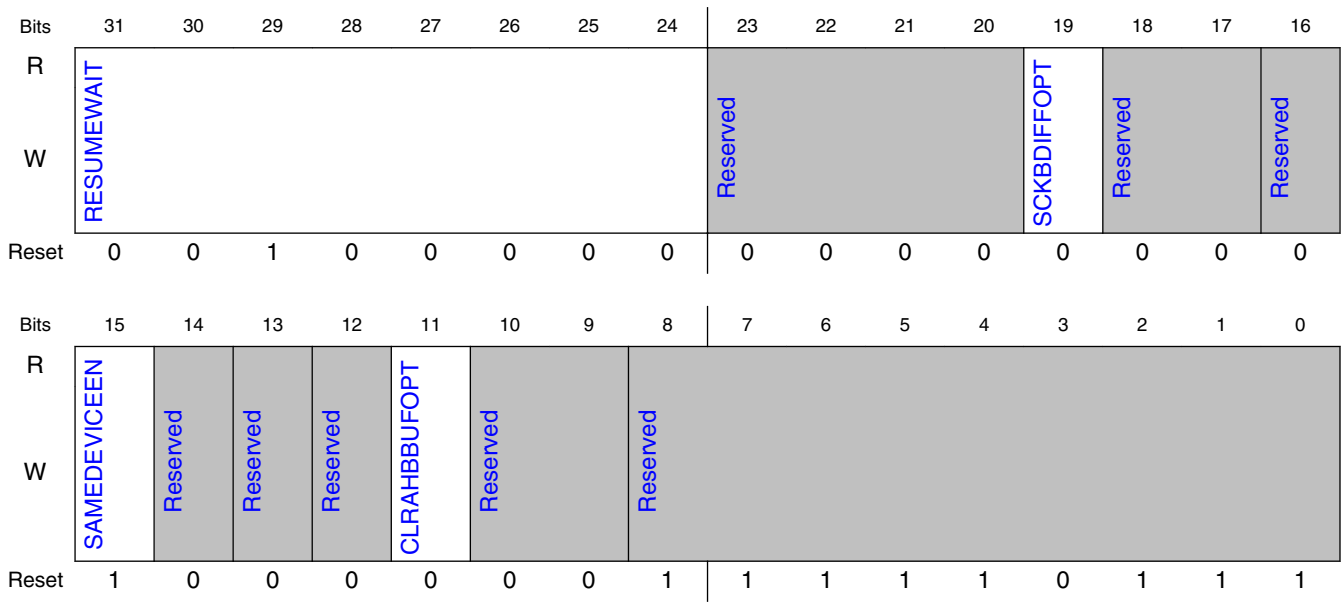
Field	Function
31-16 SEQWAIT	Command Sequence Execution will timeout and abort after SEQWAIT * 1024 Serial Root Clock cycles. When sequence execution time out occurs, there will be an interrupt generated (INTR[SEQTIMEOUT]) if this interrupt is enabled (INTEN[SEQTIMEOUTEN] is set 0x1) and AHB command is ignored by arbitrator. **NOTE:** It is not allowed to set this field to value 0x0.
15-0 AHBBUSWAIT	AHB Read/Write access to Serial Flash Memory space will timeout if not data received from Flash or data not transmitted after AHBBUSWAIT * 1024 ahb clock cycles, AHB Bus will get an error response. When AHB bus time out occurs, there will be an interrupt generated (INTR[AHBBUSTIMEOUT]) if this interrupt is enabled (INTR[AHBBUSTIMEOUT]) is set 0x1) and AHB command is ignored by arbitrator. **NOTE:** It is not allowed to set this field to value 0x0.

27.7.2.4 Module Control Register 2 (MCR2)

27.7.2.4.1 Offset

Register	Offset
MCR2	8h

27.7.2.4.2 Diagram



27.7.2.4.3 Fields

Field	Function
31-24 RESUMEWAIT	Wait cycle (in AHB clock cycle) for idle state before suspended command sequence resumed.
23-20 —	Reserved
19 SCKBDIFFOPT	B_SCLK pad can be used as A_SCLK differential clock output (inverted clock to A_SCLK). In this case, port B flash access is not available. After changing the value of this field, MCR0[SWRESET] should be set. 0b - B_SCLK pad is used as port B SCLK clock output. Port B flash access is available. 1b - B_SCLK pad is used as port A SCLK inverted clock output (Differential clock to A_SCLK). Port B flash access is not available.
18-17 —	Reserved
16 —	Reserved
15	All external devices are same devices (both in types and size) for A1/A2/B1/B2.

Table continues on the next page...

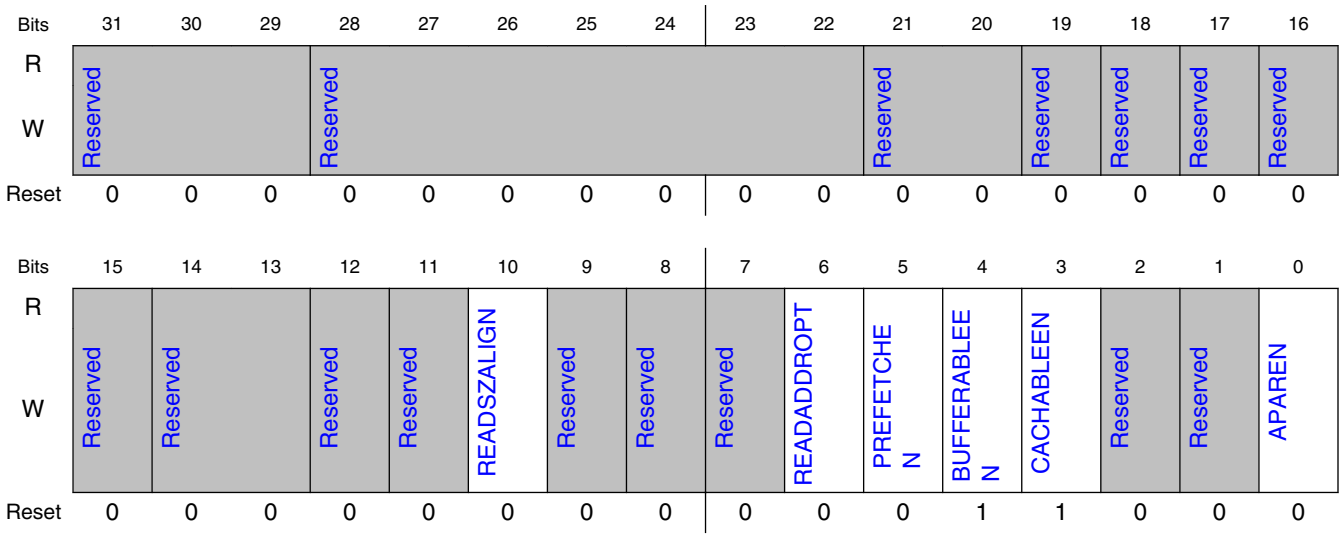
Field	Function
SAMEDEVICEEN	0b - In Individual mode, FLSHA1CRx/FLSHA2CRx/FLSHB1CRx/FLSHB2CRx register setting will be applied to Flash A1/A2/B1/B2 separately. In Parallel mode, FLSHA1CRx register setting will be applied to Flash A1 and B1, FLSHA2CRx register setting will be applied to Flash A2 and B2. FLSHB1CRx/FLSHB2CRx register settings will be ignored. 1b - FLSHA1CR0/FLSHA1CR1/FLSHA1CR2 register settings will be applied to Flash A1/A2/B1/B2. FLSHA2CRx/FLSHB1CRx/FLSHB2CRx will be ignored.
14 —	Reserved
13 —	Reserved
12 —	Reserved
11 CLRAHBBUFOPT	This bit determines whether AHB RX Buffer and AHB TX Buffer will be cleaned automatically when FlexSPI returns STOP mode ACK. Software should set this bit if AHB RX Buffer or AHB TX Buffer will be powered off in STOP mode. Otherwise AHB read access after exiting STOP mode may hit AHB RX Buffer or AHB TX Buffer but their data entries are invalid. 0b - AHB RX/TX Buffer will not be cleaned automatically when FlexSPI return Stop mode ACK. 1b - AHB RX/TX Buffer will be cleaned automatically when FlexSPI return Stop mode ACK.
10-9 —	Reserved
8-0 —	Reserved

27.7.2.5 AHB Bus Control Register (AHBCR)

27.7.2.5.1 Offset

Register	Offset
AHBCR	Ch

27.7.2.5.2 Diagram



27.7.2.5.3 Fields

Field	Function
31-29 —	Reserved
28-22 —	Reserved
21-20 —	Reserved
19 —	Reserved
18 —	Reserved
17 —	Reserved
16 —	Reserved
15 —	Reserved
14-13 —	Reserved
12 —	Reserved
11	Reserved

Table continues on the next page...

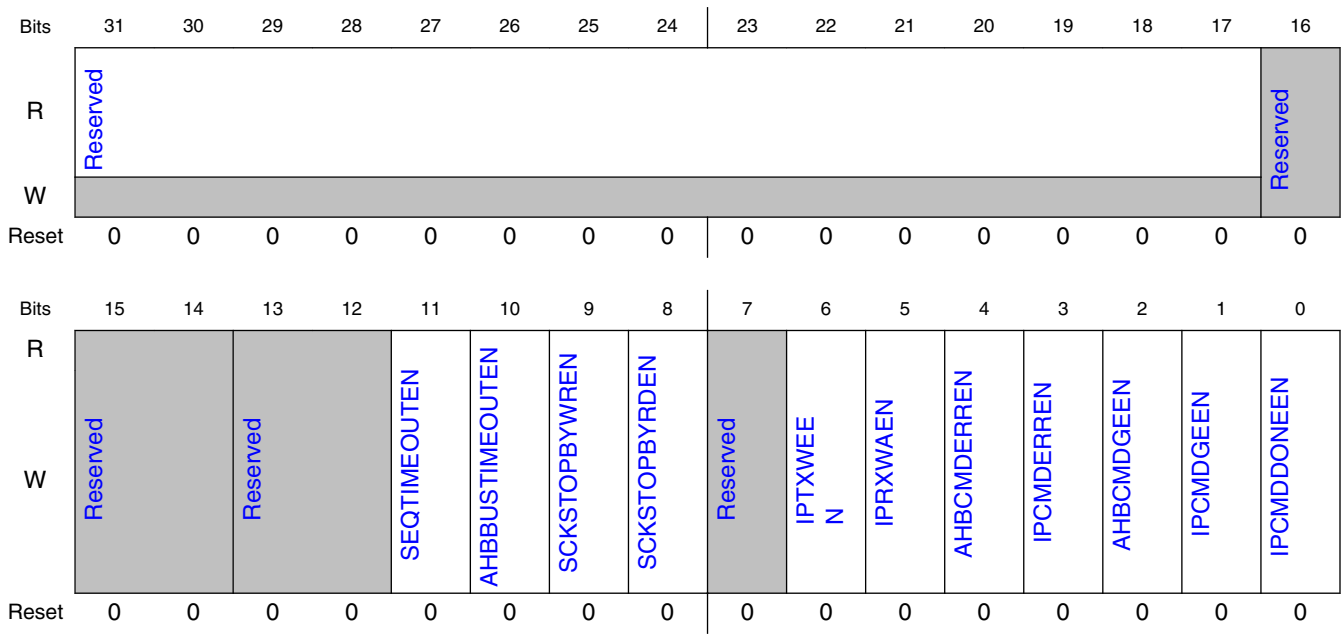
Field	Function
—	
10 READSZALIGN	AHB Read Size Alignment 0b - AHB read size will be decided by other register setting like PREFETCH_EN 1b - AHB read size to up size to 8 bytes aligned, no prefetching
9 —	Reserved
8 —	Reserved
7 —	Reserved
6 READADDROPT	AHB Read Address option bit. This option bit is intend to remove AHB burst start address alignment limitation. When FlexSPI controller is used for FPGA application, there may be requirement that FlexSPI fetch exactly the byte number as AHB burst. In this case, FPGA device should be designed as non-wordaddressable and this option bit should be set 0. 0b - There is AHB read burst start address alignment limitation when flash is accessed in parallel mode or flash is word-addressable. 1b - There is no AHB read burst start address alignment limitation. FlexSPI will fetch more data than AHB burst required to meet the alignment requirement.
5 PREFETCHEN	AHB Read Prefetch Enable. When AHB read prefetch is enabled, FlexSPI will fetch more flash read data than current AHB burst needed so that the read latency for next AHB read access will be reduced.
4 BUFFERABLEEN	Enable AHB bus bufferable write access support. This field affects the last beat of AHB write access, refer for more details about AHB bufferable write. 0b - Disabled. For all AHB write access (no matter bufferable or non-bufferable), FlexSPI will return AHB Bus ready after all data is transmitted to External device and AHB command finished. 1b - Enabled. For AHB bufferable write access, FlexSPI will return AHB Bus ready when the AHB command is granted by arbitrator and will not wait for AHB command finished.
3 CACHABLEEN	Enable AHB bus cachable read access support. 0b - Disabled. When there is AHB bus cachable read access, FlexSPI will not check whether it hit AHB TX Buffer. 1b - Enabled. When there is AHB bus cachable read access, FlexSPI will check whether it hit AHB TX Buffer first.
2 —	Reserved
1 —	Reserved
0 APAREN	Parallel mode enabled for AHB triggered Command (both read and write) . 0b - Flash will be accessed in Individual mode. 1b - Flash will be accessed in Parallel mode.

27.7.2.6 Interrupt Enable Register (INTEN)

27.7.2.6.1 Offset

Register	Offset
INTEN	10h

27.7.2.6.2 Diagram



27.7.2.6.3 Fields

Field	Function
31-17 —	Reserved
16 —	Reserved
15-14 —	Reserved
13-12 —	Reserved
11 SEQTIMEOUTEN	Sequence execution timeout interrupt enable.Refer Interrupts chapter for more details.
10	AHB Bus timeout interrupt.Refer Interrupts chapter for more details.

Table continues on the next page...

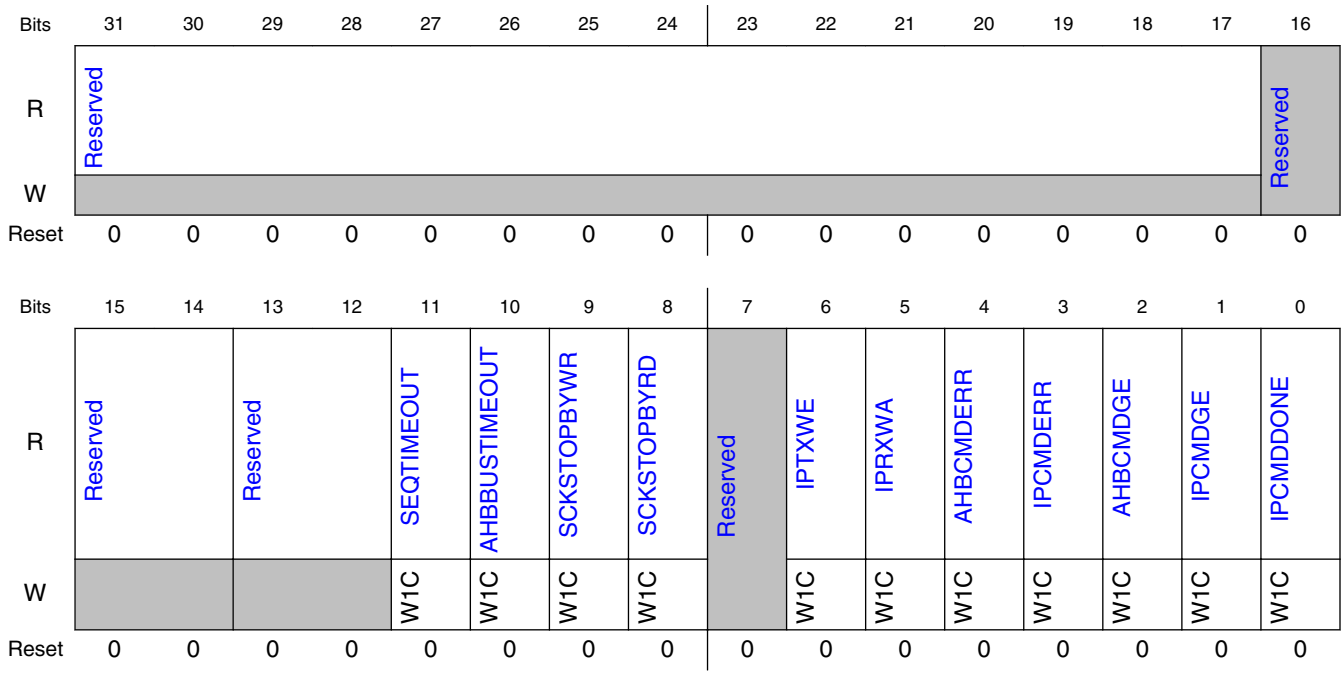
Field	Function
AHBBUSTIMEOUTEN	
9 SCKSTOPBYWREN	SCLK is stopped during command sequence because Async TX FIFO empty interrupt enable.
8 SCKSTOPBYRDEN	SCLK is stopped during command sequence because Async RX FIFO full interrupt enable.
7 —	Reserved
6 IPTXWEEN	IP TX FIFO WaterMark empty interrupt enable. IP TX FIFO has no less empty space than WaterMark level interrupt enable.
5 IPRXWAEN	IP RX FIFO WaterMark available interrupt enable. IP RX FIFO has no less valid data than WaterMark level interrupt enable.
4 AHBCMDERREN	AHB triggered Command Sequences Error Detected interrupt enable.
3 IPCMDERREN	IP triggered Command Sequences Error Detected interrupt enable.
2 AHBCMDGEEN	AHB triggered Command Sequences Grant Timeout interrupt enable.
1 IPCMDGEEN	IP triggered Command Sequences Grant Timeout interrupt enable.
0 IPCMDDONEEN	IP triggered Command Sequences Execution finished interrupt enable.

27.7.2.7 Interrupt Register (INTR)

27.7.2.7.1 Offset

Register	Offset
INTR	14h

27.7.2.7.2 Diagram



27.7.2.7.3 Fields

Field	Function
31-17 —	Reserved
16 —	Reserved
15-14 —	Reserved
13-12 —	Reserved
11 SEQTIMEOUT	Sequence execution timeout interrupt.
10 AHBBUSTIMEOUT	AHB Bus timeout interrupt.Refer Interrupts chapter for more details.
9 SCKSTOPBYWR	SCLK is stopped during command sequence because Async TX FIFO empty interrupt.
8 SCKSTOPBYRD	SCLK is stopped during command sequence because Async RX FIFO full interrupt.

Table continues on the next page...

Field	Function
7 —	Reserved
6 IPTXWE	IP TX FIFO watermark empty interrupt. IP TX FIFO has no less empty space than WaterMark level interrupt.
5 IPRXWA	IP RX FIFO watermark available interrupt. IP RX FIFO has no less valid data than WaterMark level interrupt.
4 AHBCMDERR	AHB triggered Command Sequences Error Detected interrupt. When an error detected for AHB command, this command will be ignored and not executed at all.
3 IPCMDERR	IP triggered Command Sequences Error Detected interrupt. When an error detected for IP command, this command will be ignored and not executed at all.
2 AHBCMDGE	AHB triggered Command Sequences Grant Timeout interrupt.
1 IPCMDGE	IP triggered Command Sequences Grant Timeout interrupt.
0 IPCMDDONE	IP triggered Command Sequences Execution finished interrupt. This interrupt is also generated when there is IPCMDGE or IPCMDERR interrupt generated.

27.7.2.8 LUT Key Register (LUTKEY)

27.7.2.8.1 Offset

Register	Offset
LUTKEY	18h

27.7.2.8.2 Function

The LUT Key Register contains the key to lock and unlock LUT. Refer to Look Up Table for details.

27.7.2.8.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	KEY															
Reset	0	1	0	1	1	0	1	0	1	1	1	1	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	KEY															
Reset	0	1	0	1	1	0	1	0	1	1	1	1	0	0	0	0

27.7.2.8.4 Fields

Field	Function
31-0 KEY	The Key to lock or unlock LUT. The key is 0x5AF05AF0. Read value is always 0x5AF05AF0.

27.7.2.9 LUT Control Register (LUTCR)

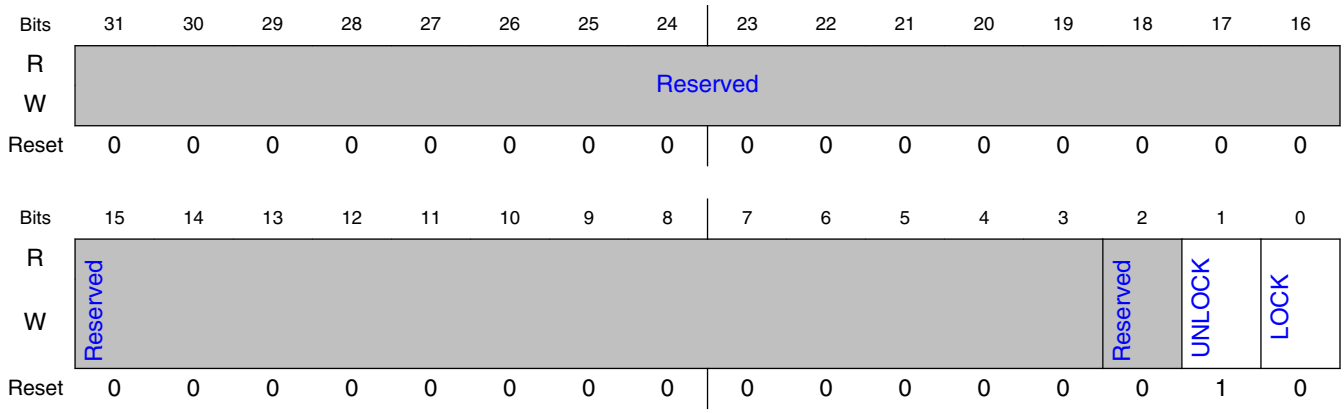
27.7.2.9.1 Offset

Register	Offset
LUTCR	1Ch

27.7.2.9.2 Function

The LUT control register is used along with LUTKEY register to lock or unlock LUT. This register has to be written immediately after writing 0x5AF05AF0 to LUTKEY register for the lock or unlock operation to be successful. Refer Look Up Table for details on locking/unlocking LUT. Setting both the LOCK and UNLOCK bits as "00" or "11" is not allowed.

27.7.2.9.3 Diagram



27.7.2.9.4 Fields

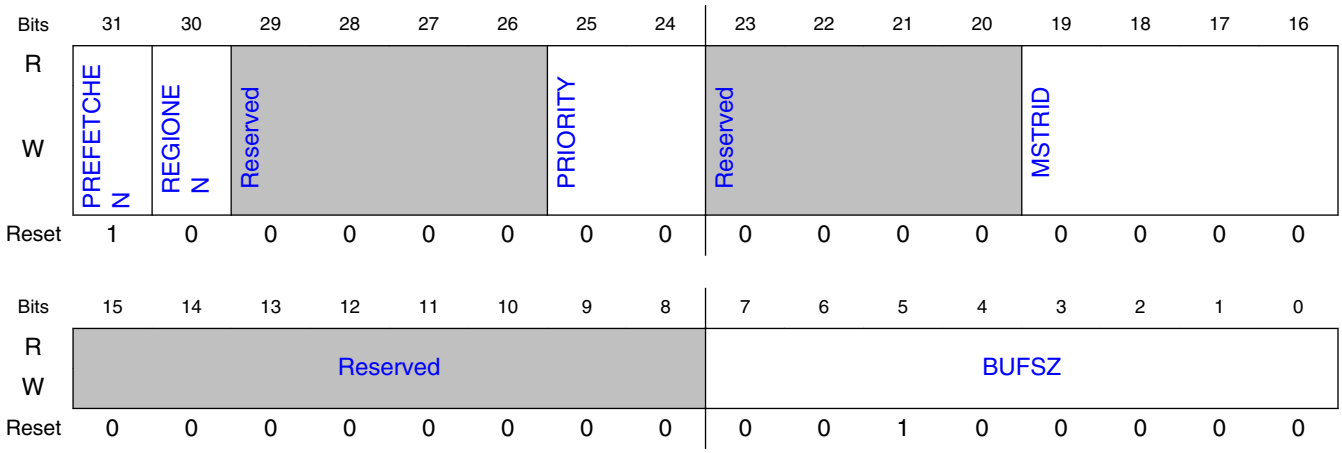
Field	Function
31-3 —	Reserved
2 —	Reserved
1 UNLOCK	Unlock LUT
0 LOCK	Lock LUT

27.7.2.10 AHB RX Buffer 0 Control Register 0 (AHBRXBUF0CR0)

27.7.2.10.1 Offset

Register	Offset
AHBRXBUF0CR0	20h

27.7.2.10.2 Diagram



27.7.2.10.3 Fields

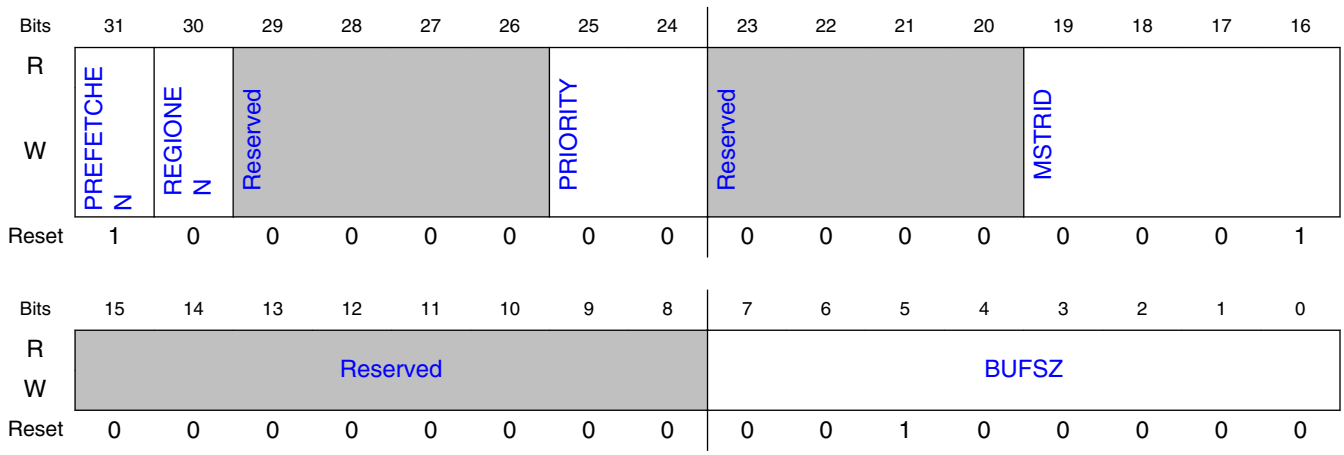
Field	Function
31 PREFETCHEN	AHB Read Prefetch Enable for current AHB RX Buffer corresponding Master. The prefetch feature is disabled when AHBCR[PREFETCHEN] is set 0. This field allows prefetch disable/ enable separately for each master.
30 REGIONEN	AHB RX Buffer address region funciton enable REGIONEN=1, hit current AHB buffer when ahb address is inside of range defined by AHBBUFREGIONiSTART and AHBBUFREGIONiEND. **NOTE:** Only AHBRXBUF0 TO 2 have this function
29-26 —	Reserved
25-24 PRIORITY	This priority for AHB Master Read which this AHB RX Buffer is assigned. 7 is the highest priority, 0 the lowest. Please refer to Command Abort and Suspend for more details.
23-20 —	Reserved
19-16 MSTRID	This AHB RX Buffer is assigned according to AHB Master with ID (MSTR_ID). Please refer to AHB RX Buffer Management for AHB RX Buffer allocation.
15-8 —	Reserved
7-0 BUFSZ	AHB RX Buffer Size in 64 bits. Please refer to AHB RX Buffer Management for more details.

27.7.2.11 AHB RX Buffer 1 Control Register 0 (AHBRXBUF1CR0)

27.7.2.11.1 Offset

Register	Offset
AHBRXBUF1CR0	24h

27.7.2.11.2 Diagram



27.7.2.11.3 Fields

Field	Function
31 PREFETCHEN	AHB Read Prefetch Enable for current AHB RX Buffer corresponding Master. The prefetch feature is disabled when AHBCR[PREFETCHEN] is set 0. This field allows prefetch disable/ enable separately for each master.
30 REGIONEN	AHB RX Buffer address region funciton enable REGIONEN=1, hit current AHB buffer when ahb address is inside of range defined by AHBBUFREGIONiSTART and AHBBUFREGIONiEND. **NOTE:** Only AHBRXBUF0 TO 2 have this function
29-26 —	Reserved
25-24 PRIORITY	This priority for AHB Master Read which this AHB RX Buffer is assigned. 7 is the highest priority, 0 the lowest. Please refer to Command Abort and Suspend for more details.
23-20 —	Reserved
19-16	This AHB RX Buffer is assigned according to AHB Master with ID (MSTR_ID).

Table continues on the next page...

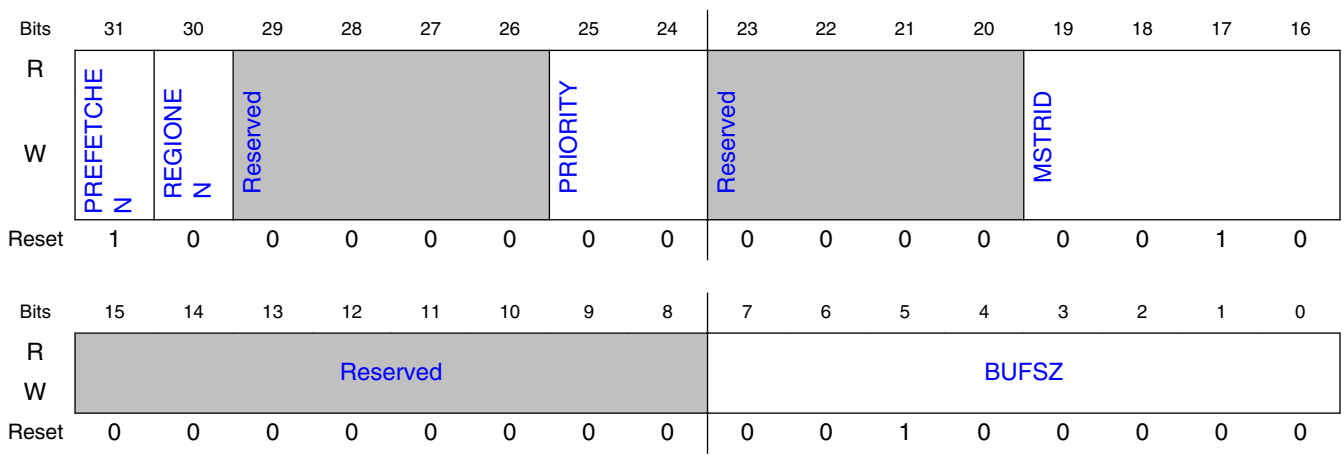
Field	Function
MSTRID	Please refer to AHB RX Buffer Management for AHB RX Buffer allocation.
15-8 —	Reserved
7-0 BUFSZ	AHB RX Buffer Size in 64 bits. Please refer to AHB RX Buffer Management for more details.

27.7.2.12 AHB RX Buffer 2 Control Register 0 (AHBRXBUF2CR0)

27.7.2.12.1 Offset

Register	Offset
AHBRXBUF2CR0	28h

27.7.2.12.2 Diagram



27.7.2.12.3 Fields

Field	Function
31 PREFETCHEN	AHB Read Prefetch Enable for current AHB RX Buffer corresponding Master. The prefetch feature is disabled when AHBCR[PREFETCHEN] is set 0. This field allows prefetch disable/enable separately for each master.
30 REGIONEN	AHB RX Buffer address region funciton enable REGIONEN=1, hit current AHB buffer when ahb address is inside of range defined by AHBBUFREGIONiSTART and AHBBUFREGIONiEND.

Table continues on the next page...

Field	Function
	NOTE: Only AHBRXBUF0 TO 2 have this function
29-26 —	Reserved
25-24 PRIORITY	This priority for AHB Master Read which this AHB RX Buffer is assigned. 7 is the highest priority, 0 the lowest. Please refer to Command Abort and Suspend for more details.
23-20 —	Reserved
19-16 MSTRID	This AHB RX Buffer is assigned according to AHB Master with ID (MSTR_ID). Please refer to AHB RX Buffer Management for AHB RX Buffer allocation.
15-8 —	Reserved
7-0 BUFSZ	AHB RX Buffer Size in 64 bits. Please refer to AHB RX Buffer Management for more details.

27.7.2.13 AHB RX Buffer 3 Control Register 0 (AHBRXBUF3CR0)

27.7.2.13.1 Offset

Register	Offset
AHBRXBUF3CR0	2Ch

27.7.2.13.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PREFETCHEN	REGIONEN	Reserved				PRIORITY		Reserved				MSTRID			
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								BUFSZ							
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

27.7.2.13.3 Fields

Field	Function
31 PREFETCHEN	AHB Read Prefetch Enable for current AHB RX Buffer corresponding Master. The prefetch feature is disabled when AHBCR[PREFETCHEN] is set 0. This field allows prefetch disable/ enable separately for each master.
30 REGIONEN	AHB RX Buffer address region funciton enable REGIONEN=1, hit current AHB buffer when ahb address is inside of range defined by AHBBUFREGIONiSTART and AHBBUFREGIONiEND. **NOTE:** Only AHBRXBUF0 TO 2 have this function
29-26 —	Reserved
25-24 PRIORITY	This priority for AHB Master Read which this AHB RX Buffer is assigned. 7 is the highest priority, 0 the lowest. Please refer to Command Abort and Suspend for more details.
23-20 —	Reserved
19-16 MSTRID	This AHB RX Buffer is assigned according to AHB Master with ID (MSTR_ID). Please refer to AHB RX Buffer Management for AHB RX Buffer allocation.
15-8 —	Reserved
7-0 BUFSZ	AHB RX Buffer Size in 64 bits. Please refer to AHB RX Buffer Management for more details.

27.7.2.14 Flash Control Register 0 (FLSHA1CR0 - FLSHB2CR0)

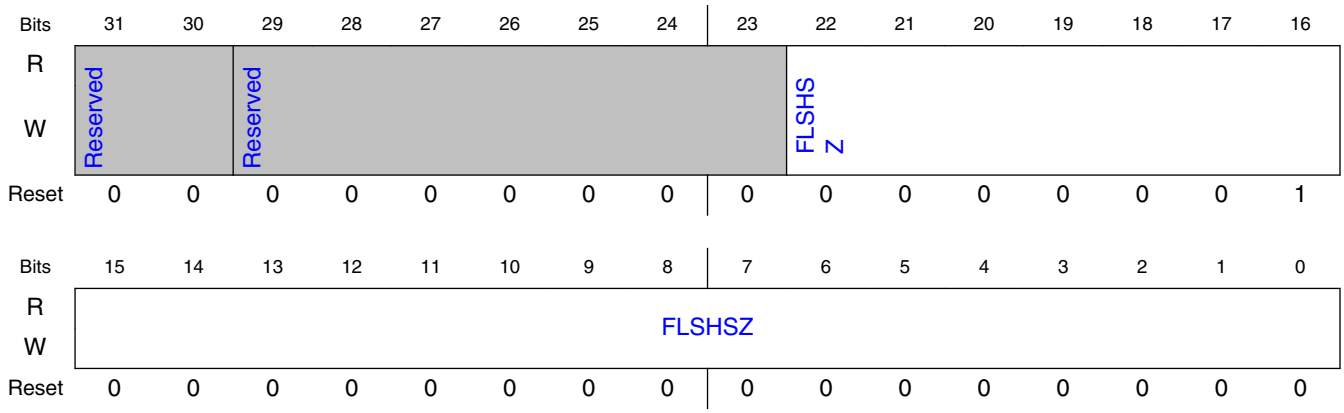
27.7.2.14.1 Offset

Register	Offset
FLSHA1CR0	60h
FLSHA2CR0	64h
FLSHB1CR0	68h
FLSHB2CR0	6Ch

27.7.2.14.2 Function

The Flash control register 0 contains Flash size setting. FlexSPI determines which device is accessed with this register setting (Chip Selection).

27.7.2.14.3 Diagram



27.7.2.14.4 Fields

Field	Function
31-30 —	Reserved
29-23 —	Reserved
22-0 FLSHSZ	Flash Size in KByte. The max flash size supported for each device is 512 MB. When FLSHSZ setting value is greater than 0x400000, the device flash size would be taken as 512 MB. The max total flash size supported (for all 4 devices) is also 512 MB. If the total flash size is larger than 512 MB, only 512 MB address space is accessible.

27.7.2.15 Flash Control Register 1 (FLSHA1CR1 - FLSHB2CR1)

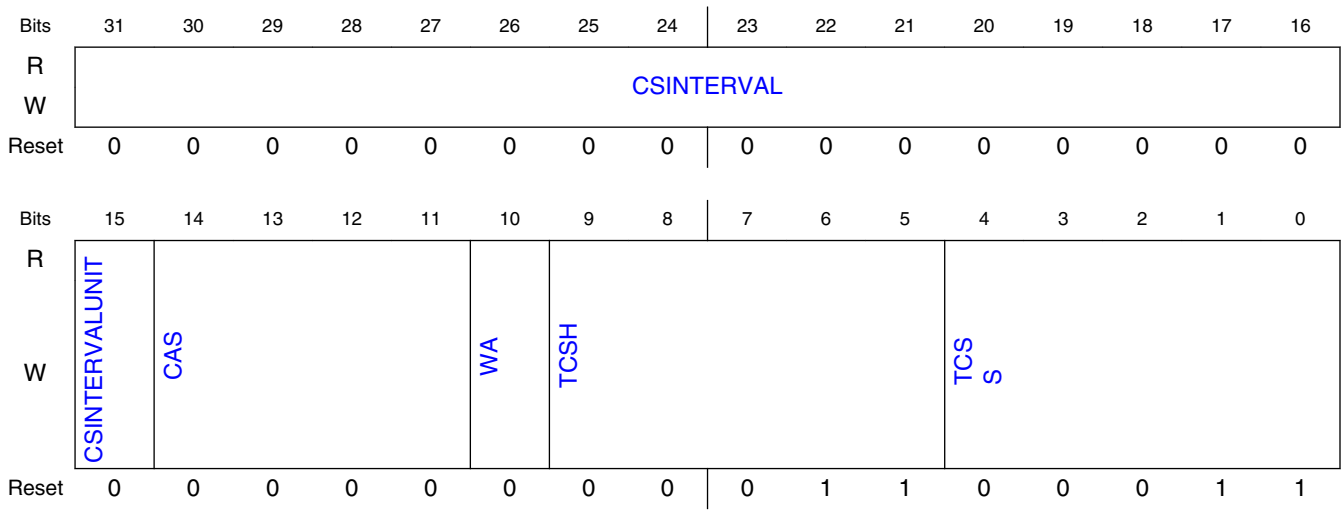
27.7.2.15.1 Offset

Register	Offset
FLSHA1CR1	70h
FLSHA2CR1	74h
FLSHB1CR1	78h
FLSHB2CR1	7Ch

27.7.2.15.2 Function

The Flash control register 1 contains the setting for FlexSPI to meet Flash device specific timings and Flash internal address space.

27.7.2.15.3 Diagram



27.7.2.15.4 Fields

Field	Function
31-16 CSINTERVAL	This field is used to set the minimum interval between flash device Chip selection deassertion and flash device Chip selection assertion. If external flash has a limitation on the interval between command sequences, this field should be set accordingly. If there is no limitation, set this field with value 0x0. When CSINTERVALUNIT is 0x0, the chip selection invalid interval is: CSINTERVAL * 1 serial clock cycle; When CSINTERVALUNIT is 0x1, the chip selection invalid interval is: CSINTERVAL * 256 serial clock cycle. **NOTE:** The chip selection interval is 2 cycle at least even if CSINTERVAL is less than 2.
15 CSINTERVALUNIT	CS interval unit 0b - The CS interval unit is 1 serial clock cycle 1b - The CS interval unit is 256 serial clock cycle
14-11 CAS	Column Address Size. When external flash has separate address field for row address and column address, this field should be set to flash column address bit width. FlexSPI will automatically split flash mapped address to Row address and Column address according to CAS field and WA field setting. This bit should be set to 0x0 when external Flash don't support column address. FlexSPI will transmit all flash address bits as Row address. For flash address mapping, please refer to Flash memory map for more detail.
10 WA	Word Addressable. This bit should be set when external Flash is word addressable. If Flash is word addressable, it should be access in terms of 16 bits. At this time, FlexSPI will not transmit Flash address bit 0 to external Flash. For flash address mapping, please refer to Flash memory map for more detail.
9-5	Serial Flash CS Hold time.

Table continues on the next page...

Field	Function
TCSH	This field is used to meet flash TCSH timing requirement. Serial flash CS Hold time promised by FlexSPI is: TCSH in serial root clock cycles (for both SDR and DDR mode). Please refer to FlexSPI Input Timing for more detail.
4-0 TCSS	Serial Flash CS setup time. This field is used to meet flash TCSS timing requirement. Serial flash CS Setup time promised by FlexSPI is: (TCSS + 1/2) serial root clock cycles (for both SDR and DDR mode). Please refer to FlexSPI Input Timing for more detail.

27.7.2.16 Flash Control Register 2 (FLSHA1CR2 - FLSHB2CR2)

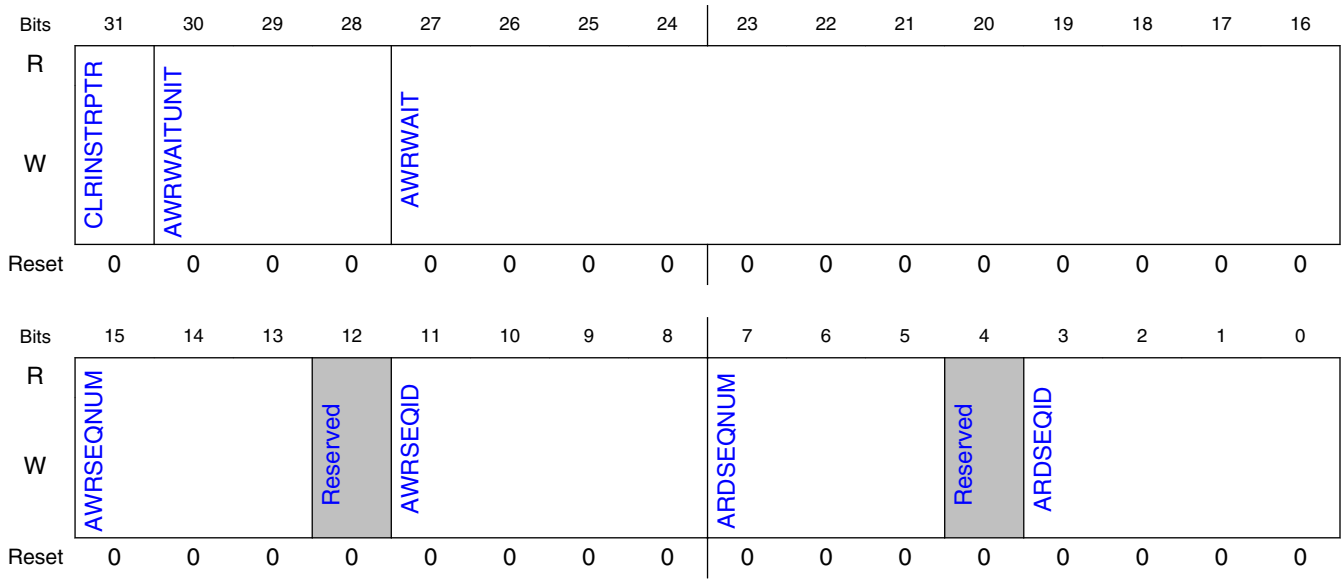
27.7.2.16.1 Offset

Register	Offset
FLSHA1CR2	80h
FLSHA2CR2	84h
FLSHB1CR2	88h
FLSHB2CR2	8Ch

27.7.2.16.2 Function

Flash Control Register 2 contains setting field for AHB Bus access configuration. If the 4 external device are in different types, AHB read/write command may use different command sequences and AHB bus ready wait time may be also different.

27.7.2.16.3 Diagram



27.7.2.16.4 Fields

Field	Function
31 CLRINSTRPTR	Clear the instruction pointer which is internally saved pointer by JMP_ON_CS. Refer Programmable Sequence Engine for details. This field is used for AHB Read access to external Flash supporting XIP (Execute-In-Place) mode.
30-28 AWRWAITUNIT	AWRWAIT unit 000b - The AWRWAIT unit is 2 ahb clock cycle 001b - The AWRWAIT unit is 8 ahb clock cycle 010b - The AWRWAIT unit is 32 ahb clock cycle 011b - The AWRWAIT unit is 128 ahb clock cycle 100b - The AWRWAIT unit is 512 ahb clock cycle 101b - The AWRWAIT unit is 2048 ahb clock cycle 110b - The AWRWAIT unit is 8192 ahb clock cycle 111b - The AWRWAIT unit is 32768 ahb clock cycle
27-16 AWRWAIT	For certain devices (such as FPGA), it need some time to write data into internal memory after the command sequences finished on FlexSPI interface. If another Read command sequence comes before previous programming finished internally, the read data may be wrong. This field is used to hold AHB Bus ready for AHB write access to wait the programming finished in external device. Then there will be no AHB read command triggered before the programming finished in external device. The Wait cycle between AHB triggered command sequences finished on FlexSPI interface and AHB return Bus ready: AWRWAIT * AWRWAITUNIT
15-13 AWRSEQNUM	Sequence Number for AHB Write triggered Command. For certain flash devices , a Flash programming access is done by several command sequences. AHB write Command will trigger (AWRSEQNUM+1) command sequences to external flash every time. FlexSPI executes the sequences in LUT incrementally.

Table continues on the next page...

Field	Function
	NOTE: • Software should make sure the last sequence index never exceed LUT sequence numbers: AWRSEQID+AWRSEQNUM < 16 • Software need to make sure field AWRSEQNUM and LUT is configured correctly according to external device spec. FlexSPI don't check the sequence and just execute them one by one.
12 —	Reserved
11-8 AWRSEQID	Sequence Index for AHB Write triggered Command.
7-5 ARDSEQNUM	Sequence Number for AHB Read triggered Command in LUT. For certain flash devices , a Flash reading access is done by several command sequences. AHB read Command will trigger (ARDSEQNUM+1) command sequences to external flash every time. FlexSPI executes the sequences in LUT incrementally. **NOTE:** • Software should make sure the last sequence index never exceed LUT sequence numbers: ARDSEQID+ARDSEQNUM <= 16 • Software need to make sure field ARDSEQNUM and LUT is configured correctly according to external device spec. FlexSPI don't check the sequence and just execute them one by one.
4 —	Reserved
3-0 ARDSEQID	Sequence Index for AHB Read triggered Command in LUT.

27.7.2.17 Flash Control Register 4 (FLSHCR4)

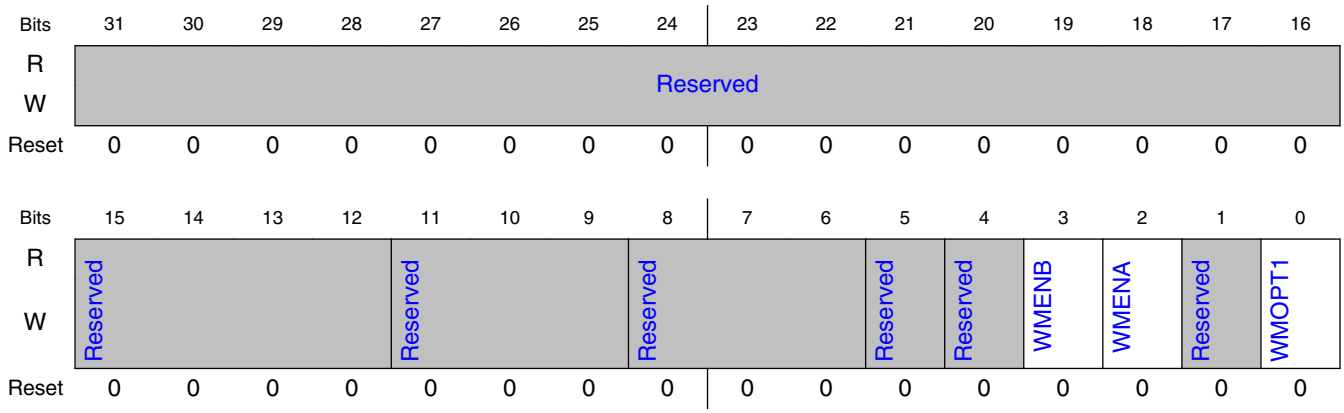
27.7.2.17.1 Offset

Register	Offset
FLSHCR4	94h

27.7.2.17.2 Function

The flash control register 4 provide the configuration for all external devices.

27.7.2.17.3 Diagram



27.7.2.17.4 Fields

Field	Function
31-12 —	Reserved
11-9 —	Reserved
8-6 —	Reserved
5 —	Reserved
4 —	Reserved
3 WMENB	Write mask enable bit for flash device on port B. When write mask function is needed for memory device on port B, this bit must be set. 0b - Write mask is disabled, DQS(RWDS) pin will be un-driven when writing to external device. 1b - Write mask is enabled, DQS(RWDS) pin will be driven by FlexSPI as write mask output when writing to external device.
2 WMENA	Write mask enable bit for flash device on port A. When write mask function is needed for memory device on port A, this bit must be set. 0b - Write mask is disabled, DQS(RWDS) pin will be un-driven when writing to external device. 1b - Write mask is enabled, DQS(RWDS) pin will be driven by FlexSPI as write mask output when writing to external device.
1 —	Reserved
0 WMOPT1	Write mask option bit 1. This option bit could be used to remove AHB write burst start address alignment limitation. 0b - DQS pin will be used as Write Mask when writing to external device. There is no limitation on AHB write burst start address alignment when flash is accessed in individual mode. 1b - DQS pin will not be used as Write Mask when writing to external device. There is limitation on AHB write burst start address alignment when flash is accessed in individual mode.

27.7.2.18 IP Control Register 0 (IPCR0)

27.7.2.18.1 Offset

Register	Offset
IPCR0	A0h

27.7.2.18.2 Function

The IP control registers provide all the configuration required for IP commands. This register provides the flash device's start address, instead of SoC address, to be accessed for IP command. FlexSPI will determine the chip select automatically according to this start address.

NOTE

- It's not allowed to issue IP command crossing Flash device boundaries. Otherwise there will be IPCMDERR interrupt generated.
- This register should be set before IP command triggered.
- This register setting should not be changed while an IP command is in progress.

27.7.2.18.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SFAR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SFAR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

27.7.2.18.4 Fields

Field	Function
31-0	Serial Flash Address for IP command.

Field	Function
SFAR	

27.7.2.19 IP Control Register 1 (IPCR1)

27.7.2.19.1 Offset

Register	Offset
IPCR1	A4h

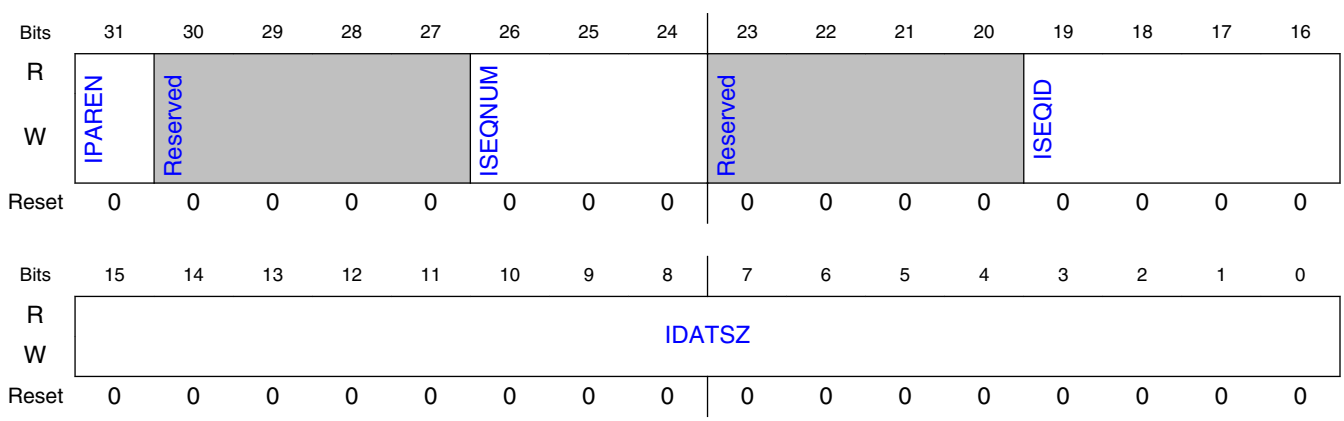
27.7.2.19.2 Function

The IP control registers provide all the configuration required for IP command. This register provides the flash read/program data size, sequence index in LUT, sequence number and individual/parallel mode setting for IP command.

NOTE

- This register should be before IP command triggered.
- This register setting should not be changed before IP command finished.

27.7.2.19.3 Diagram



27.7.2.19.4 Fields

Field	Function
31	Parallel mode Enabled for IP command.

Table continues on the next page...

Field	Function
IPAREN	0b - Flash will be accessed in Individual mode. 1b - Flash will be accessed in Parallel mode.
30-27 —	Reserved
26-24 ISEQNUM	Sequence Number for IP command: ISEQNUM+1.
23-20 —	Reserved
19-16 ISEQID	Sequence Index in LUT for IP command.
15-0 IDATSZ	Flash Read/Program Data Size (in Bytes) for IP command.

27.7.2.20 IP Command Register (IPCMD)

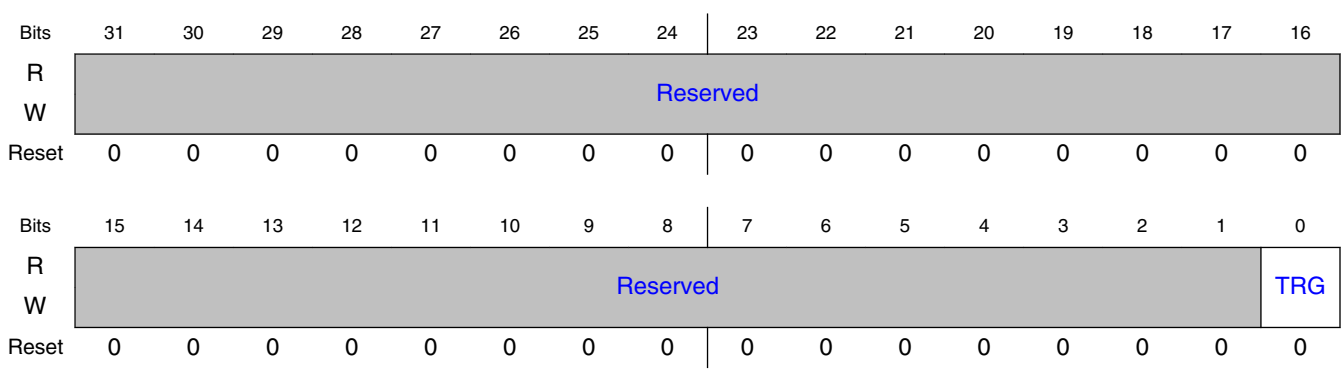
27.7.2.20.1 Offset

Register	Offset
IPCMD	B0h

27.7.2.20.2 Function

This register is used to trigger a IP command to access external flash device. IP command will be executed on FlexSPI interface after granted by arbitrator.

27.7.2.20.3 Diagram



27.7.2.20.4 Fields

Field	Function
31-1 —	Reserved
0 TRG	Setting this bit will trigger an IP Command. **NOTE:** • It's not allowed to trigger another IP command before previous IP command is finished on FlexSPI interface. Software need to poll register bit INTR_IP_CMD_DONE or wait for this interrupt in order to wait for IP command finished.

27.7.2.21 IP RX FIFO Control Register (IPRXFCR)

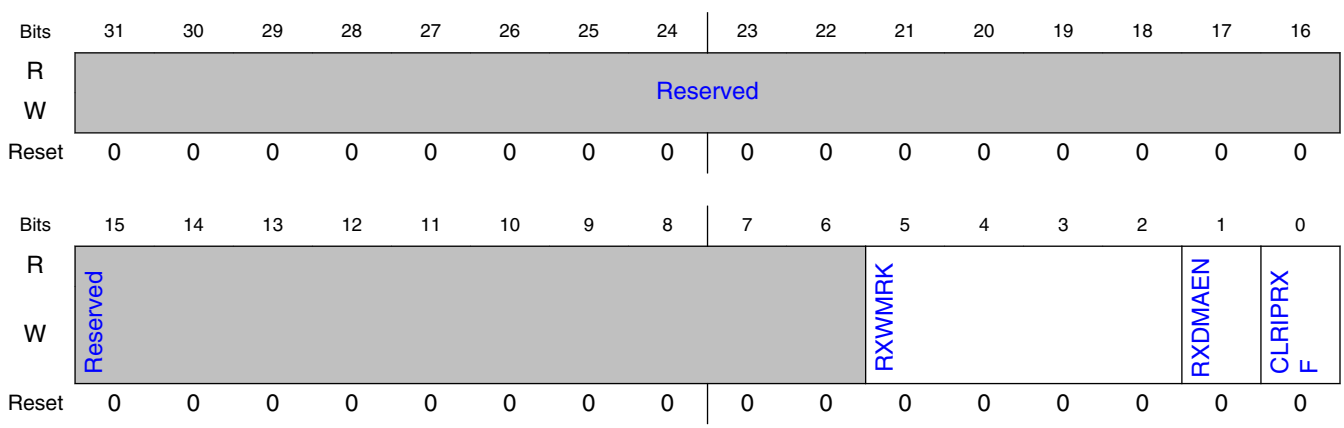
27.7.2.21.1 Offset

Register	Offset
IPRXFCR	B8h

27.7.2.21.2 Function

This register provides the configuration fields for IP RX FIFO management.

27.7.2.21.3 Diagram



27.7.2.21.4 Fields

Field	Function
31-6	Reserved

Table continues on the next page...

Field	Function
—	
5-2 RXWMRK	Watermark level is (RXWMRK+1)*64 Bits. Interrupt register bit IPRXWA is set when filling level in IP RX FIFO is no less than Watermark level by FlexSPI. There will be a DMA request when the filling level is no less than Watermark level and DMA read is enabled (register bit RXDMAEN is set). There will be an IPRXWA (IP RX FIFO watermark available) interrupt generated when the filling level is no less than Watermark level and IPRXWA interrupt is enabled (register bit INTEN_IPRXWA is set). **NOTE:** After write-1-clear to INTR[IPRXWA], read address should be rolled back to start address (memory mapped). If IP RX FIFO is read by IP bus, the read address to IP RX FIFO should roll back to RFDR0; If IP RX FIFO is read by AHB bus, the read address to IP RX FIFO should roll back to ARDF_BASE.
1 RXDMAEN	IP RX FIFO reading by DMA enabled. 0b - IP RX FIFO would be read by processor. 1b - IP RX FIFO would be read by DMA.
0 CLRIPRXF	Clear all valid data entries in IP RX FIFO. The read/write pointers in IP RX FIFO will be reset.

27.7.2.22 IP TX FIFO Control Register (IPTXFCR)

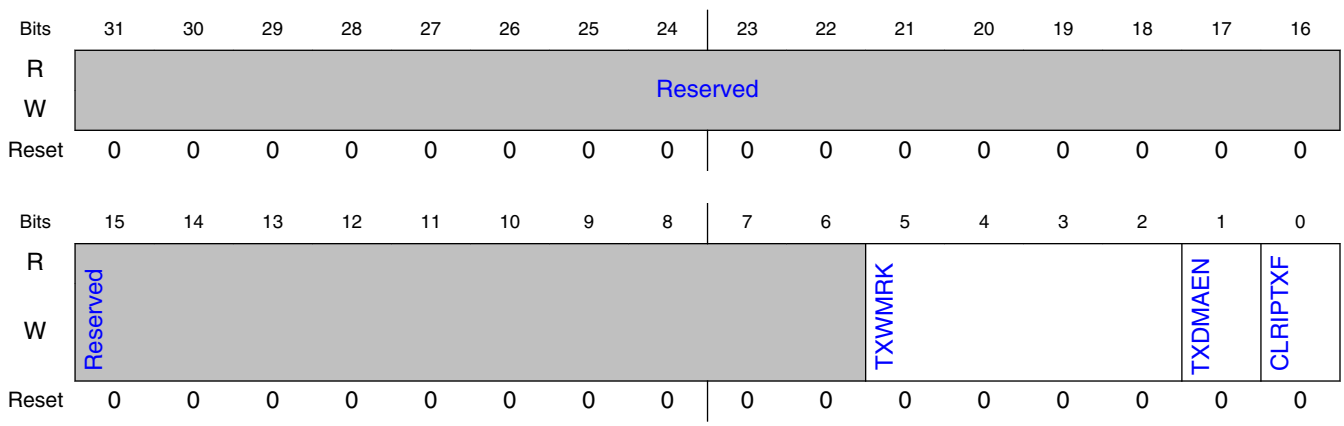
27.7.2.22.1 Offset

Register	Offset
IPTXFCR	BCh

27.7.2.22.2 Function

This register provides the configuration fields for IP TX FIFO management.

27.7.2.22.3 Diagram



27.7.2.22.4 Fields

Field	Function
31-6 —	Reserved
5-2 TXWMRK	Watermark level is (TXWMRK+1)*64 Bits. Interrupt register bit IPTXWE is set when empty level in IP TX FIFO is no less than Watermark level by FlexSPI. There will be a DMA request when empty level is no less than Watermark level and DMA filling is enable (register bit TXDMAEN is set). There will be an IPTXWE (IP TX FIFO Watermark Empty) interrupt generated when empty level is no less than Watermark level and IPTXWE interrupt is enable (register bit INTEN_IPTXWE is set). **NOTE:** • The watermark level should be no more than the write window. • The watermark level should be no more than IP TX FIFO size. • The write address to IP RX FIFO should roll back to the start address of write window after pushing IP TX FIFO by writing-one-clear to INTR[IPTXWE].
1 TXDMAEN	IP TX FIFO filling by DMA enabled. 0b - IP TX FIFO would be filled by processor. 1b - IP TX FIFO would be filled by DMA.
0 CLRIPTXF	Clear all valid data entries in IP TX FIFO. The read/write pointers in IP TX FIFO will be reset.

27.7.2.23 DLL Control Register 0 (DLLACR - DLLBCR)

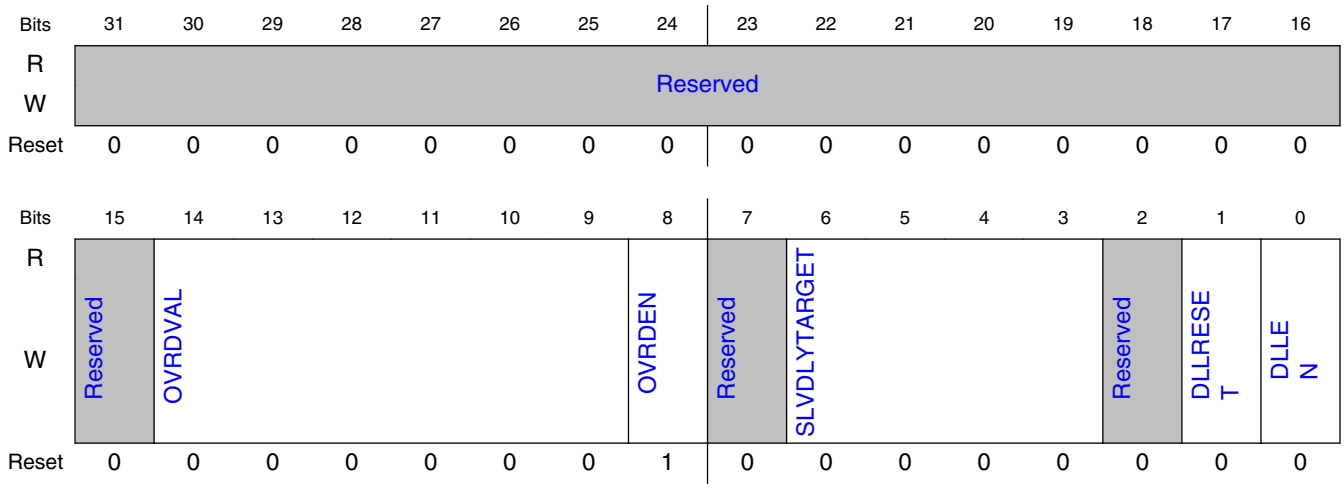
27.7.2.23.1 Offset

Register	Offset
DLLACR	C0h
DLLBCR	C4h

27.7.2.23.2 Function

This register provides the configuration fields for Flash A/B sample clock DLL.

27.7.2.23.3 Diagram



27.7.2.23.4 Fields

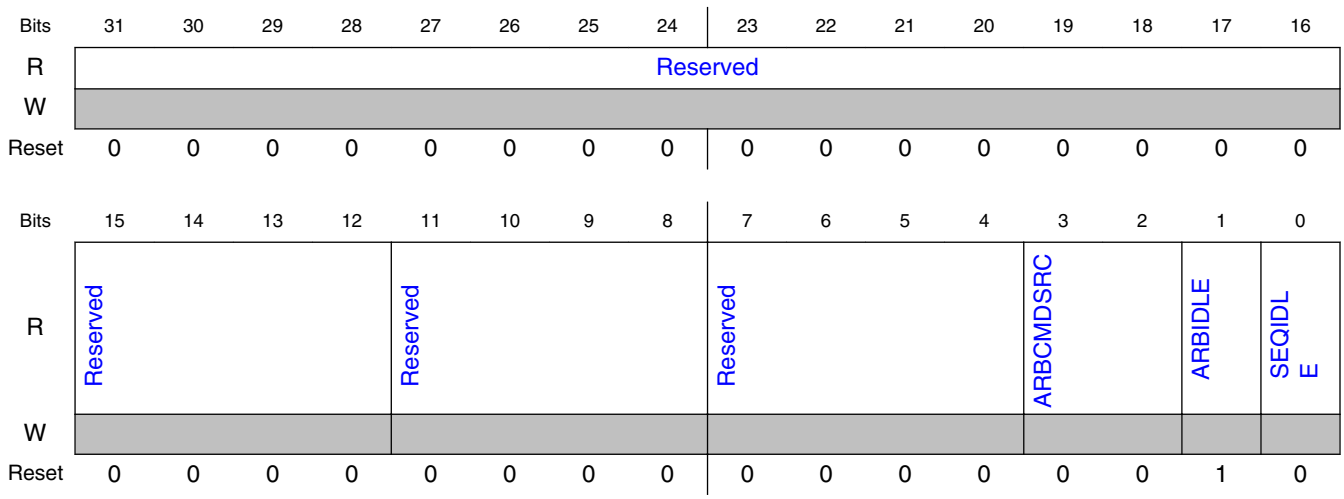
Field	Function
31-15 —	Reserved
14-9 OVRDVAL	Slave clock delay line delay cell number selection override value. When OVRDEN is set 0x1, the delay cell number in DLL is OVRDVAL+1.
8 OVRDEN	Slave clock delay line delay cell number selection override enable.
7 —	Reserved
6-3 SLVDLYTARGET	The delay target for slave delay line is: ((SLVDLYTARGET+1) * 1/32 * clock cycle of reference clock (serial root clock). If serial root clock is >= 100 MHz, DLLEN set to 0x1, OVRDEN set to =0x0, then SLVDLYTARGET setting of 0xF is recommended.
2 —	Reserved
1 DLLRESET	Software could force a reset on DLL by setting this field to 0x1. This will cause the DLL to lose lock and re-calibrate to detect an ref_clock half period phase shift. The reset action is edge triggered, so software need to clear this bit after set this bit (no delay limitation).
0 DLLEN	DLL calibration enable. When this bit is cleared, DLL calibration is disabled and the delay cell number in slave delay line is always 1. Please note that SLV delay line is overridden when OVRDEN bit is set and this bit field setting is ignored.

27.7.2.24 Status Register 0 (STS0)

27.7.2.24.1 Offset

Register	Offset
STS0	E0h

27.7.2.24.2 Diagram



27.7.2.24.3 Fields

Field	Function
31-12 —	Reserved
11-8 —	Reserved
7-4 —	Reserved
3-2 ARBCMDSRC	This status field indicates the trigger source of current command sequence granted by arbitrator. This field value is meaningless when ARB_CTL is not busy (STS0[ARBIDLE]=0x1). 00b - Triggered by AHB read command (triggered by AHB read). 01b - Triggered by AHB write command (triggered by AHB Write). 10b - Triggered by IP command (triggered by setting register bit IPCMD.TRG). 11b - Triggered by suspended command (resumed).
1 ARBIDLE	This status bit indicates the state machine in ARB_CTL is busy and there is command sequence granted by arbitrator and not finished yet on FlexSPI interface. When ARB_CTL state (ARBIDLE=0x1) is idle, there will be no transaction on FlexSPI interface also (SEQIDLE=0x1). So this bit should be polled to wait for FlexSPI controller become idle instead of SEQIDLE.

Table continues on the next page...

Field	Function
0 SEQIDLE	This status bit indicates the state machine in SEQ_CTL is idle and there is command sequence executing on FlexSPI interface.

27.7.2.25 Status Register 1 (STS1)

27.7.2.25.1 Offset

Register	Offset
STS1	E4h

27.7.2.25.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved				IPCMDERRCODE				Reserved				IPCMDERRID			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				AHBCMDERRCODE				Reserved				AHBCMDERRID			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

27.7.2.25.3 Fields

Field	Function
31-28 —	Reserved
27-24 IPCMDERRCODE	Indicates the Error Code when IP command Error detected. This field will be cleared when INTR[IPCMDERR] is write-1-clear(w1c). 0000b - No error. 0010b - IP command with JMP_ON_CS instruction used in the sequence.

Table continues on the next page...

Field	Function
	0011b - There is unknown instruction opcode in the sequence. 0100b - Instruction DUMMY_SDR/DUMMY_RWDS_SDR used in DDR sequence. 0101b - Instruction DUMMY_DDR/DUMMY_RWDS_DDR used in SDR sequence. 0110b - Flash access start address exceed the whole flash address range (A1/A2/B1/B2). 1110b - Sequence execution timeout. 1111b - Flash boundary crossed.
23-20 —	Reserved
19-16 IPCMDERRID	Indicates the sequence Index when IP command error detected. This field will be cleared when INTR[IPCMDERR] is write-1-clear(w1c).
15-12 —	Reserved
11-8 AHBCMDERRCODE	Indicates the Error Code when AHB command Error detected. This field will be cleared when INTR[AHBCMDERR] is write-1-clear(w1c). 0000b - No error. 0010b - AHB Write command with JMP_ON_CS instruction used in the sequence. 0011b - There is unknown instruction opcode in the sequence. 0100b - Instruction DUMMY_SDR/DUMMY_RWDS_SDR used in DDR sequence. 0101b - Instruction DUMMY_DDR/DUMMY_RWDS_DDR used in SDR sequence. 1110b - Sequence execution timeout.
7-4 —	Reserved
3-0 AHBCMDERRID	Indicates the sequence index when an AHB command error is detected. This field will be cleared when INTR[AHBCMDERR] is write-1-clear(w1c).

27.7.2.26 Status Register 2 (STS2)

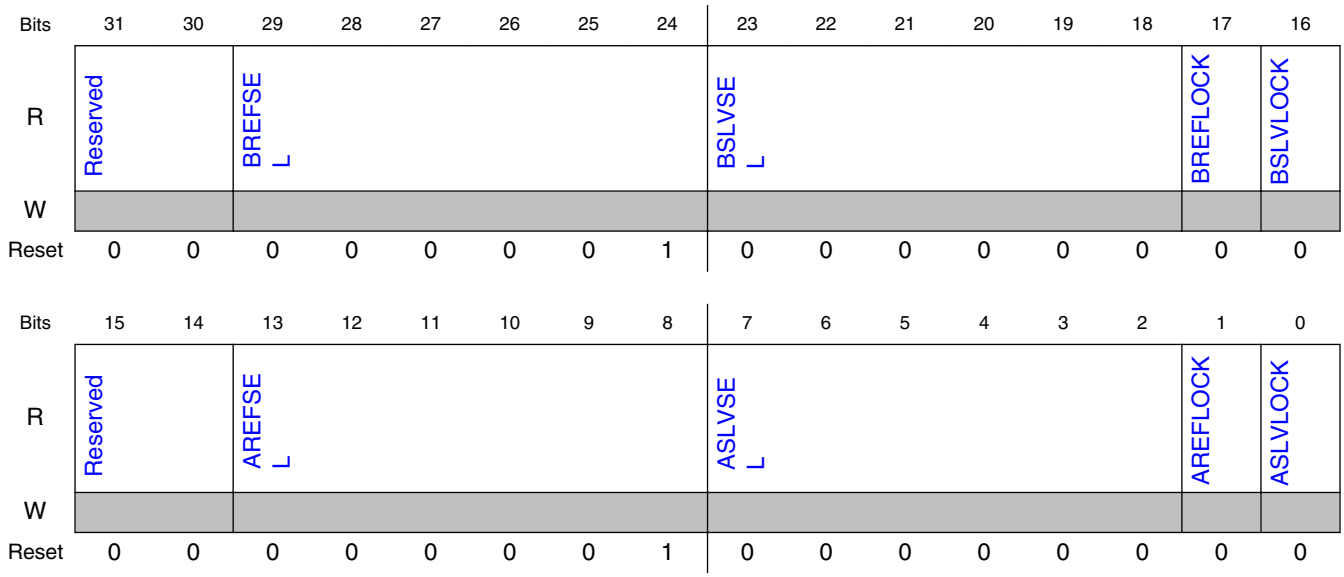
27.7.2.26.1 Offset

Register	Offset
STS2	E8h

27.7.2.26.2 Function

This register indicates the status of Flash A and B sample clock DLLs.

27.7.2.26.3 Diagram



27.7.2.26.4 Fields

Field	Function
31-30 —	Reserved
29-24 BREFSEL	Flash B sample clock reference delay line delay cell number selection.
23-18 BSLVSEL	Flash B sample clock slave delay line delay cell number selection.
17 BREFLOCK	Flash B sample clock reference delay line locked.
16 BSLVLOCK	Flash B sample clock slave delay line locked.
15-14 —	Reserved
13-8 AREFSEL	Flash A sample clock reference delay line delay cell number selection.
7-2 ASLVSEL	Flash A sample clock slave delay line delay cell number selection .
1 AREFLOCK	Flash A sample clock reference delay line locked.
0 ASLVLOCK	Flash A sample clock slave delay line locked.

27.7.2.27 AHB Suspend Status Register (AHBSPNDSTS)

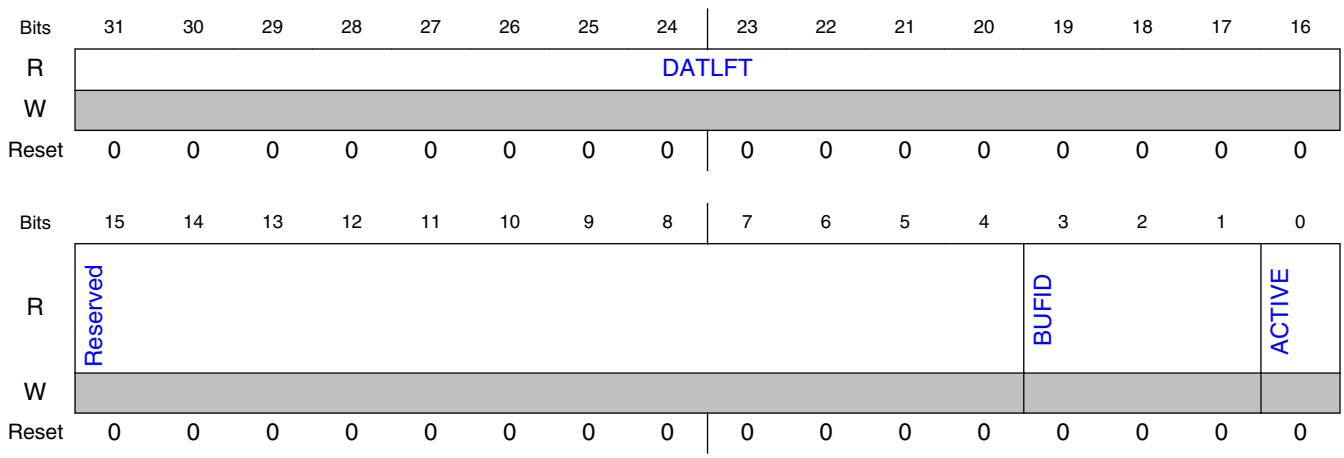
27.7.2.27.1 Offset

Register	Offset
AHBSPNDSTS	ECh

27.7.2.27.2 Function

Indicates the status of Suspended AHB Read Prefetch command sequence. When there is IP/AHB command triggered and arbitrator is processing an AHB Read sequence (prefetching more data not for current AHB burst), the prefetch sequence will be suspended and may be resumed when there is no transaction on FlexSPI any more. FlexSPI saves only one AHB prefetch sequence. When a new prefetch sequence is suspended with an active sequence suspended already, previous suspended sequence will be removed and never resumed. Refer Command Abort and Suspend for more details.

27.7.2.27.3 Diagram



27.7.2.27.4 Fields

Field	Function
31-16 DATLFT	Left Data size for suspended command sequence (in byte).
15-4	Reserved

Table continues on the next page...

Field	Function
—	
3-1 BUFID	AHB RX BUF ID for suspended command sequence.
0 ACTIVE	Indicates if an AHB read prefetch command sequence has been suspended.

27.7.2.28 IP RX FIFO Status Register (IPRXFSTS)

27.7.2.28.1 Offset

Register	Offset
IPRXFSTS	F0h

27.7.2.28.2 Function

This status register indicates the status of IP RX FIFO.

27.7.2.28.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RDCNTR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								FILL							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

27.7.2.28.4 Fields

Field	Function
31-16 RDCNTR	Total Read Data Counter: RDCNTR * 64 Bits.
15-8 —	Reserved

Table continues on the next page...

Field	Function
7-0 FILL	Fill level of IP RX FIFO. Valid Data entries in IP RX FIFO is: FILL * 64 Bits.

27.7.2.29 IP TX FIFO Status Register (IPTXFSTS)

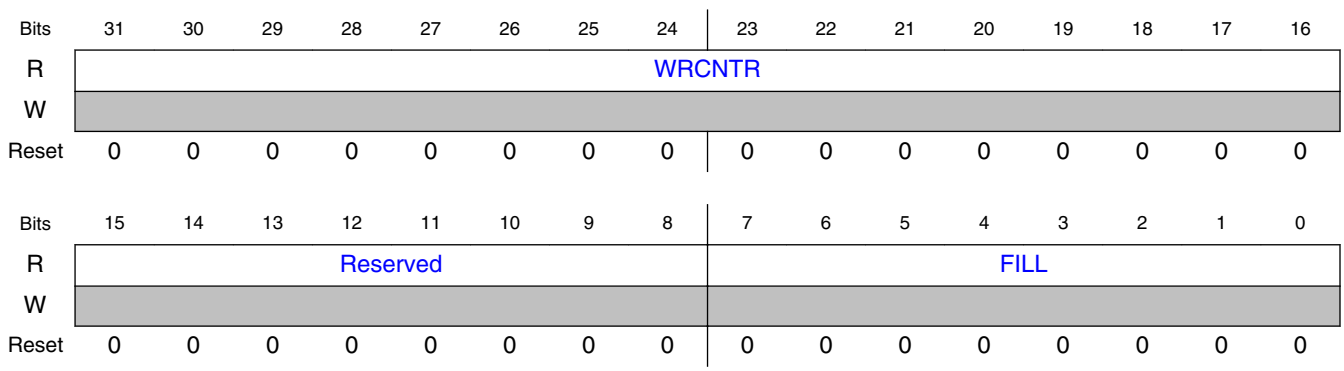
27.7.2.29.1 Offset

Register	Offset
IPTXFSTS	F4h

27.7.2.29.2 Function

This status register indicates the status of IP TX FIFO.

27.7.2.29.3 Diagram



27.7.2.29.4 Fields

Field	Function
31-16 WRCNTR	Total Write Data Counter: WRCNTR * 64 Bits.
15-8 —	Reserved
7-0 FILL	Fill level of IP TX FIFO. Valid Data entries in IP TX FIFO is: FILL * 64 Bits.

27.7.2.30 IP RX FIFO Data Register a (RFDR0 - RFDR31)

27.7.2.30.1 Offset

For a = 0 to 31:

Register	Offset
RFDRa	100h + (a × 4h)

27.7.2.30.2 Function

These registers provide read access to IP RX FIFO by IPS bus. The read value is unknown for read access to invalid entries in IP RX FIFO.

27.7.2.30.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RXDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RXDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

27.7.2.30.4 Fields

Field	Function
31-0 RXDATA	RX Data

27.7.2.31 IP TX FIFO Data Register a (TFDR0 - TFDR31)

27.7.2.31.1 Offset

For a = 0 to 31:

Register	Offset
TFDRa	180h + (a × 4h)

27.7.2.31.2 Function

These registers provide write access to IP TX FIFO by IPS bus.

27.7.2.31.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	TXDATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	TXDATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

27.7.2.31.4 Fields

Field	Function
31-0 TXDATA	TX Data

27.7.2.32 LUT a (LUT0 - LUT63)

27.7.2.32.1 Offset

For a = 0 to 63:

Register	Offset
LUTa	200h + (a × 4h)

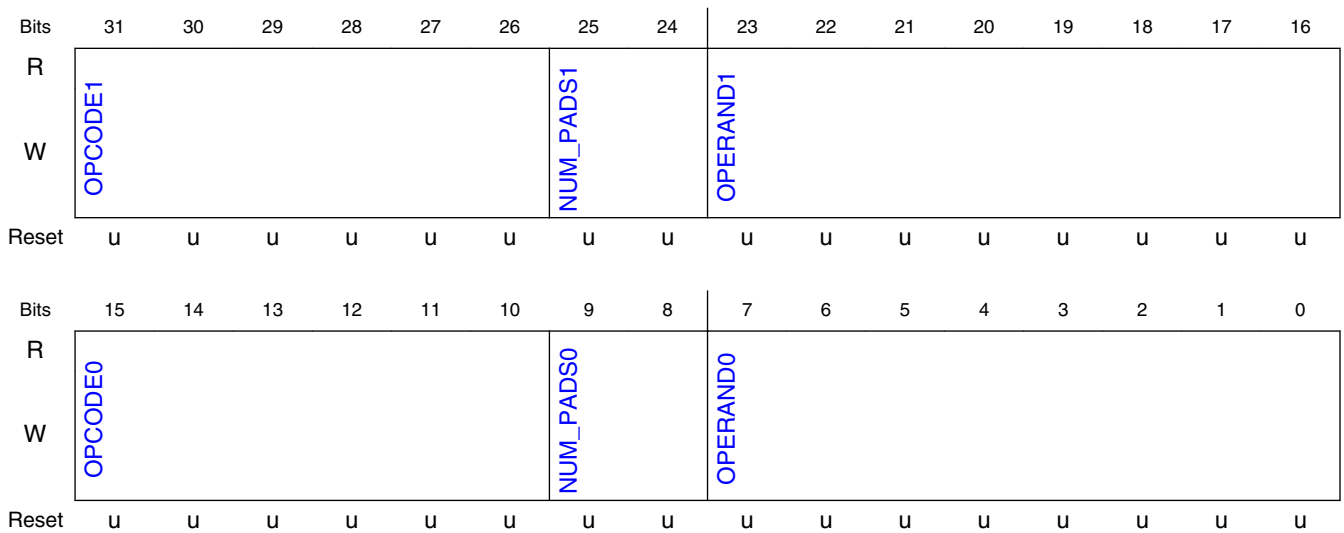
27.7.2.32.2 Function

The LUT is a look-up-table for command sequences. Software should set the sequence index before triggering an IP command or AHB command. FlexSPI will fetch the command sequence from LUT when IP/AHB command triggered. There are 16 command sequences in LUT. Refer Look Up Table for more details.

NOTE

LUT is implemented as memory, so the reset value is unknown.

27.7.2.32.3 Diagram



27.7.2.32.4 Fields

Field	Function
31-26 OPCODE1	OPCODE1
25-24 NUM_PADS1	NUM_PADS1
23-16 OPERAND1	OPERAND1
15-10 OPCODE0	OPCODE
9-8 NUM_PADS0	NUM_PADS0
7-0	OPERAND0

Field	Function
OPERAND0	

27.7.3 FlexSPI register descriptions

27.7.3.1 FlexSPI memory map

FlexSPI2 base address: 402A_4000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Module Control Register 0 (MCR0)	32	RW	FFFF_80C2h
4h	Module Control Register 1 (MCR1)	32	RW	FFFF_FFFFh
8h	Module Control Register 2 (MCR2)	32	RW	2000_81F7h
Ch	AHB Bus Control Register (AHBCR)	32	RW	0000_0018h
10h	Interrupt Enable Register (INTEN)	32	RW	0000_0000h
14h	Interrupt Register (INTR)	32	W1C	0000_0000h
18h	LUT Key Register (LUTKEY)	32	RW	5AF0_5AF0h
1Ch	LUT Control Register (LUTCR)	32	RW	0000_0002h
20h	AHB RX Buffer 0 Control Register 0 (AHBRXBUF0CR0)	32	RW	8000_0020h
24h	AHB RX Buffer 1 Control Register 0 (AHBRXBUF1CR0)	32	RW	8001_0020h
28h	AHB RX Buffer 2 Control Register 0 (AHBRXBUF2CR0)	32	RW	8002_0020h
2Ch	AHB RX Buffer 3 Control Register 0 (AHBRXBUF3CR0)	32	RW	8003_0020h
60h	Flash Control Register 0 (FLSHA1CR0)	32	RW	0001_0000h
64h	Flash Control Register 0 (FLSHA2CR0)	32	RW	0001_0000h
68h	Flash Control Register 0 (FLSHB1CR0)	32	RW	0001_0000h
6Ch	Flash Control Register 0 (FLSHB2CR0)	32	RW	0001_0000h
70h	Flash Control Register 1 (FLSHA1CR1)	32	RW	0000_0063h
74h	Flash Control Register 1 (FLSHA2CR1)	32	RW	0000_0063h
78h	Flash Control Register 1 (FLSHB1CR1)	32	RW	0000_0063h
7Ch	Flash Control Register 1 (FLSHB2CR1)	32	RW	0000_0063h
80h	Flash Control Register 2 (FLSHA1CR2)	32	RW	0000_0000h
84h	Flash Control Register 2 (FLSHA2CR2)	32	RW	0000_0000h
88h	Flash Control Register 2 (FLSHB1CR2)	32	RW	0000_0000h
8Ch	Flash Control Register 2 (FLSHB2CR2)	32	RW	0000_0000h
94h	Flash Control Register 4 (FLSHCR4)	32	RW	0000_0000h
A0h	IP Control Register 0 (IPCR0)	32	RW	0000_0000h

Table continues on the next page...

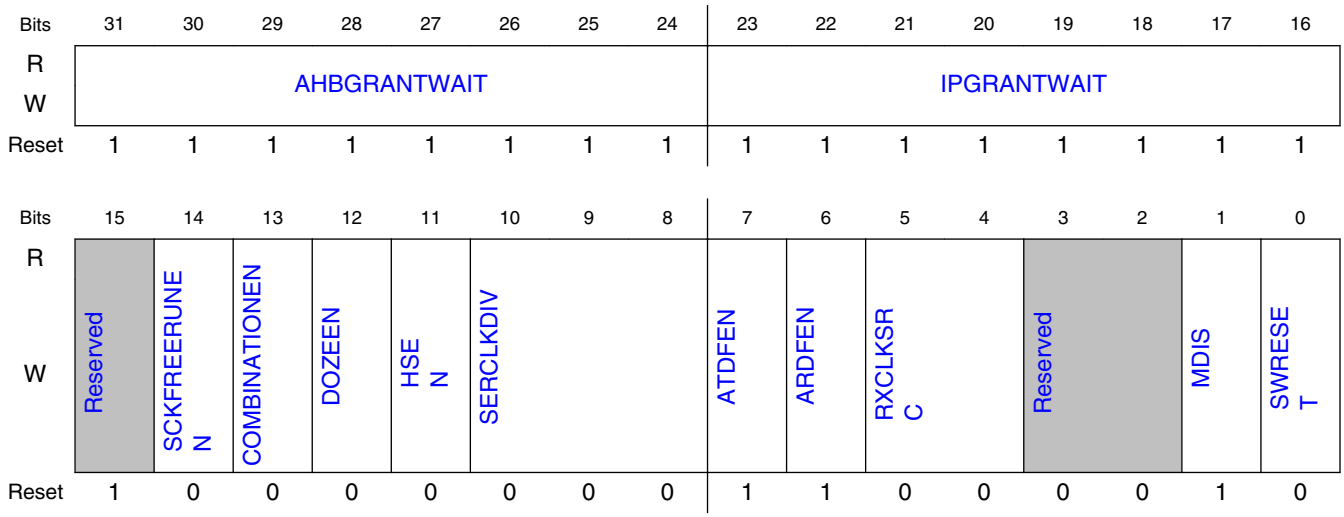
Offset	Register	Width (In bits)	Access	Reset value
A4h	IP Control Register 1 (IPCR1)	32	RW	0000_0000h
B0h	IP Command Register (IPCMD)	32	RW	0000_0000h
B8h	IP RX FIFO Control Register (IPRXFCR)	32	RW	0000_0000h
BCh	IP TX FIFO Control Register (IPTXFCR)	32	RW	0000_0000h
C0h	DLL Control Register 0 (DLLACR)	32	RW	0000_0100h
C4h	DLL Control Register 0 (DLLBCR)	32	RW	0000_0100h
E0h	Status Register 0 (STS0)	32	RO	0000_0002h
E4h	Status Register 1 (STS1)	32	RO	0000_0000h
E8h	Status Register 2 (STS2)	32	RO	0100_0100h
ECh	AHB Suspend Status Register (AHBSPNDSTS)	32	RO	0000_0000h
F0h	IP RX FIFO Status Register (IPRXFSTS)	32	RO	0000_0000h
F4h	IP TX FIFO Status Register (IPTXFSTS)	32	RO	0000_0000h
100h - 17Ch	IP RX FIFO Data Register a (RFDR0 - RFDR31)	32	RO	0000_0000h
180h - 1FCh	IP TX FIFO Data Register a (TFDR0 - TFDR31)	32	WO	0000_0000h
200h - 2FCh	LUT a (LUT0 - LUT63)	32	RW	Table 27-23

27.7.3.2 Module Control Register 0 (MCR0)

27.7.3.2.1 Offset

Register	Offset
MCR0	0h

27.7.3.2.2 Diagram



27.7.3.2.3 Fields

Field	Function
31-24 AHBGRANTWAIT	Timeout wait cycle for AHB command grant. If AHB Triggered Command is not granted by arbitrator, it will timeout after AHBGRANTTIMEOUT * 1024 AHB Clock cycles. This grant timeout may occur when the pending command sequence is IP triggered and the read/write data size is too large. When an AHB command grant time out occurs, there will be an interrupt generated (INTR[AHBCMDGE]) if this interrupt is enabled (INTEN[AHBCMDGEEN] is set) and AHB command is ignored by arbitrator. **NOTE:** This field is for debug only, please keep default value! It is not allowed to set this field to value 0x0.
23-16 IPGRANTWAIT	Time out wait cycle for IP command grant. If IP Triggered Command is not granted by arbitrator, it will timeout after IPGRANTTIMEOUT * 1024 AHB Clock cycles. This grant timeout maybe occur when pending command sequence is AHB triggered and read/write data size is too large. When IP command grant time out occurs, there will be an interrupt generated (INTR[IPCMDGE]) if this interrupt is enabled (INTEN[IPCMDGEEN] is set 0x1) and IP command is ignored by arbitrator. **NOTE:** This field is for debug only, please keep default value! It is not allowed to set this field to value 0x0.
15 —	Reserved
14 SCKFREERUNEN	This bit is used to force SCLK output free-running. For FPGA applications, external device may use SCLK as reference clock to its internal PLL. If SCLK free-running is enabled, data sampling with loopback clock from SCLK pad is not supported (MCR0[RXCLKSRC]=2). 0b - Disable. 1b - Enable.
13 COMBINATIONEN	This bit is to support Flash Octal mode access by combining Port A and B Data pins (A_DATA[3:0] and B_DATA[3:0]), when Port A and Port B are of 4 bit data width.

Table continues on the next page...

Field	Function
	NOTE: Combination mode is not supported if Port A and Port B are 8 bit data width. This bit should be set to zero in this case. 0b - Disable. 1b - Enable.
12 DOZEEN	Doze mode enable bit 0b - Doze mode support disabled. AHB clock and serial clock will not be gated off when there is doze mode request from system. 1b - Doze mode support enabled. AHB clock and serial clock will be gated off when there is doze mode request from system.
11 HSEN	Half Speed Serial Flash access Enable. This bit enables the divide by 2 of the clock to external serial flash devices (A_SCLK/B_SCLK) for all commands (for both SDR and DDR mode). FlexSPI need to be set into MDIS mode before changing value of HSEN. Otherwise, it is possible to cause issue on internal logic/state machine. 0b - Disable divide by 2 of serial flash clock for half speed commands. 1b - Enable divide by 2 of serial flash clock for half speed commands.
10-8 SERCLKDIV	The serial root clock could be divided inside FlexSPI . See Clocks section for more details on clocking. **NOTE:** Don't change this field during IP's normal operation mode. Alter the value after putting IP into stop mode. 000b - Divided by 1 001b - Divided by 2 010b - Divided by 3 011b - Divided by 4 100b - Divided by 5 101b - Divided by 6 110b - Divided by 7 111b - Divided by 8
7 ATDFEN	Enable AHB bus Write Access to IP TX FIFO. 0b - IP TX FIFO should be written by IP Bus. AHB Bus write access to IP TX FIFO memory space will get bus error response. 1b - IP TX FIFO should be written by AHB Bus. IP Bus write access to IP TX FIFO memory space will be ignored but no bus error response.
6 ARDFEN	Enable AHB bus Read Access to IP RX FIFO. 0b - IP RX FIFO should be read by IP Bus. AHB Bus read access to IP RX FIFO memory space will get bus error response. 1b - IP RX FIFO should be read by AHB Bus. IP Bus read access to IP RX FIFO memory space will always return data zero but no bus error response.
5-4 RXCLKSRC	Sample Clock source selection for Flash Reading Refer RX Clock Source Features for more details 00b - Dummy Read strobe generated by FlexSPI Controller and loopback internally. 01b - Dummy Read strobe generated by FlexSPI Controller and loopback from DQS pad. 10b - Reserved 11b - Flash provided Read strobe and input from DQS pad
3-2 —	Reserved
1 MDIS	Module Disable When module disabled, AHB/serial clock will be gated off internally to save power. Only register access (except LUT/IP RX FIFO/IP TX FIFO) is allowed.
0 SWRESET	Software Reset This bit is auto-cleared by hardware after software reset done. Configuration registers will not be reset.

27.7.3.3 Module Control Register 1 (MCR1)

27.7.3.3.1 Offset

Register	Offset
MCR1	4h

27.7.3.3.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SEQWAIT															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	AHBBUSWAIT															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

27.7.3.3.3 Fields

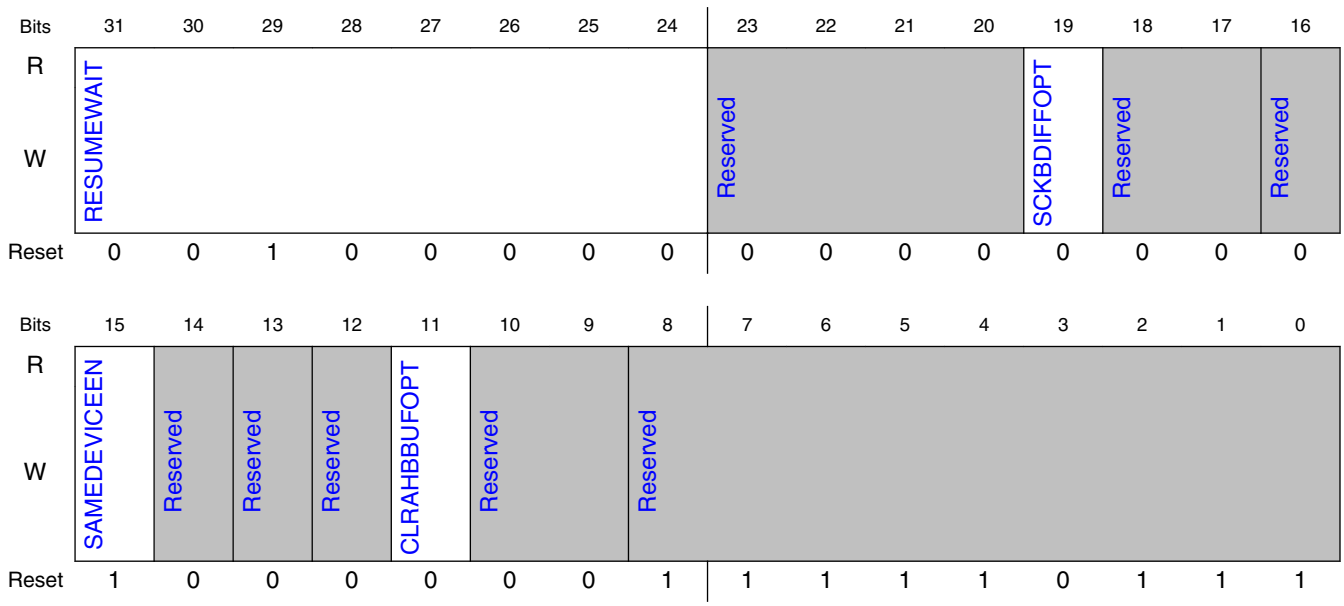
Field	Function
31-16 SEQWAIT	Command Sequence Execution will timeout and abort after SEQWAIT * 1024 Serial Root Clock cycles. When sequence execution time out occurs, there will be an interrupt generated (INTR[SEQTIMEOUT]) if this interrupt is enabled (INTEN[SEQTIMEOUTEN] is set 0x1) and AHB command is ignored by arbitrator. **NOTE:** It is not allowed to set this field to value 0x0.
15-0 AHBBUSWAIT	AHB Read/Write access to Serial Flash Memory space will timeout if not data received from Flash or data not transmitted after AHBBUSWAIT * 1024 ahb clock cycles, AHB Bus will get an error response. When AHB bus time out occurs, there will be an interrupt generated (INTR[AHBBUSTIMEOUT]) if this interrupt is enabled (INTR[AHBBUSTIMEOUT]) is set 0x1) and AHB command is ignored by arbitrator. **NOTE:** It is not allowed to set this field to value 0x0.

27.7.3.4 Module Control Register 2 (MCR2)

27.7.3.4.1 Offset

Register	Offset
MCR2	8h

27.7.3.4.2 Diagram



27.7.3.4.3 Fields

Field	Function
31-24 RESUMEWAIT	Wait cycle (in AHB clock cycle) for idle state before suspended command sequence resumed.
23-20 —	Reserved
19 SCKBDIFFOPT	B_SCLK pad can be used as A_SCLK differential clock output (inverted clock to A_SCLK). In this case, port B flash access is not available. After changing the value of this field, MCR0[SWRESET] should be set. 0b - B_SCLK pad is used as port B SCLK clock output. Port B flash access is available. 1b - B_SCLK pad is used as port A SCLK inverted clock output (Differential clock to A_SCLK). Port B flash access is not available.
18-17 —	Reserved
16 —	Reserved
15	All external devices are same devices (both in types and size) for A1/A2/B1/B2.

Table continues on the next page...

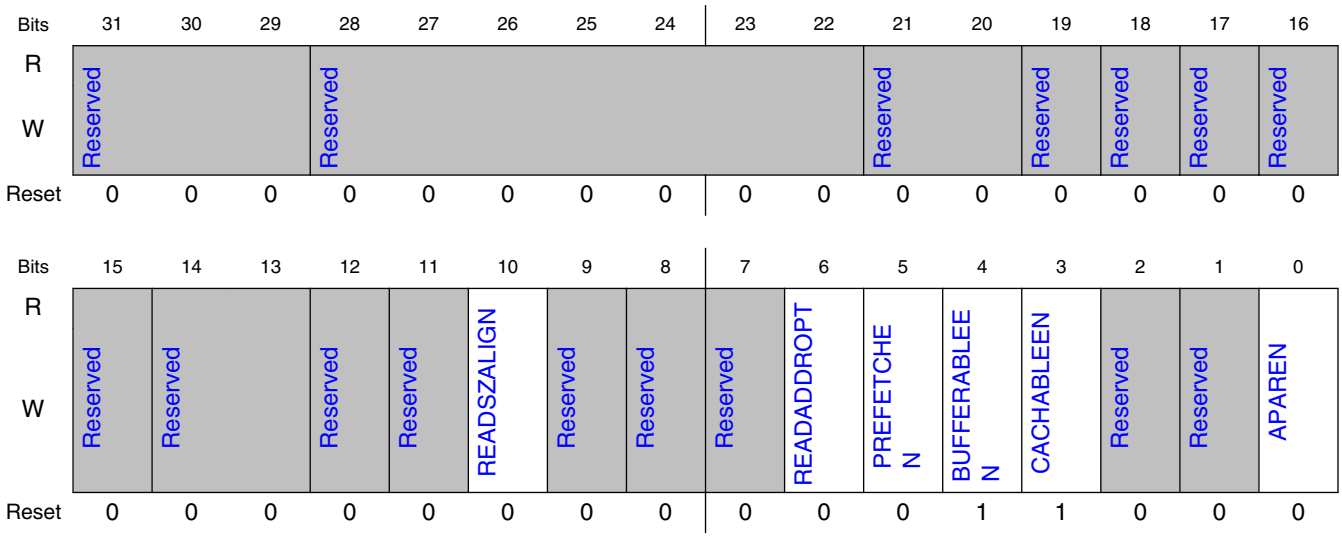
Field	Function
SAMEDEVICEEN	0b - In Individual mode, FLSHA1CRx/FLSHA2CRx/FLSHB1CRx/FLSHB2CRx register setting will be applied to Flash A1/A2/B1/B2 separately. In Parallel mode, FLSHA1CRx register setting will be applied to Flash A1 and B1, FLSHA2CRx register setting will be applied to Flash A2 and B2. FLSHB1CRx/FLSHB2CRx register settings will be ignored. 1b - FLSHA1CR0/FLSHA1CR1/FLSHA1CR2 register settings will be applied to Flash A1/A2/B1/B2. FLSHA2CRx/FLSHB1CRx/FLSHB2CRx will be ignored.
14 —	Reserved
13 —	Reserved
12 —	Reserved
11 CLRAHBBUFOPT	This bit determines whether AHB RX Buffer and AHB TX Buffer will be cleaned automatically when FlexSPI returns STOP mode ACK. Software should set this bit if AHB RX Buffer or AHB TX Buffer will be powered off in STOP mode. Otherwise AHB read access after exiting STOP mode may hit AHB RX Buffer or AHB TX Buffer but their data entries are invalid. 0b - AHB RX/TX Buffer will not be cleaned automatically when FlexSPI return Stop mode ACK. 1b - AHB RX/TX Buffer will be cleaned automatically when FlexSPI return Stop mode ACK.
10-9 —	Reserved
8-0 —	Reserved

27.7.3.5 AHB Bus Control Register (AHBCR)

27.7.3.5.1 Offset

Register	Offset
AHBCR	Ch

27.7.3.5.2 Diagram



27.7.3.5.3 Fields

Field	Function
31-29 —	Reserved
28-22 —	Reserved
21-20 —	Reserved
19 —	Reserved
18 —	Reserved
17 —	Reserved
16 —	Reserved
15 —	Reserved
14-13 —	Reserved
12 —	Reserved
11	Reserved

Table continues on the next page...

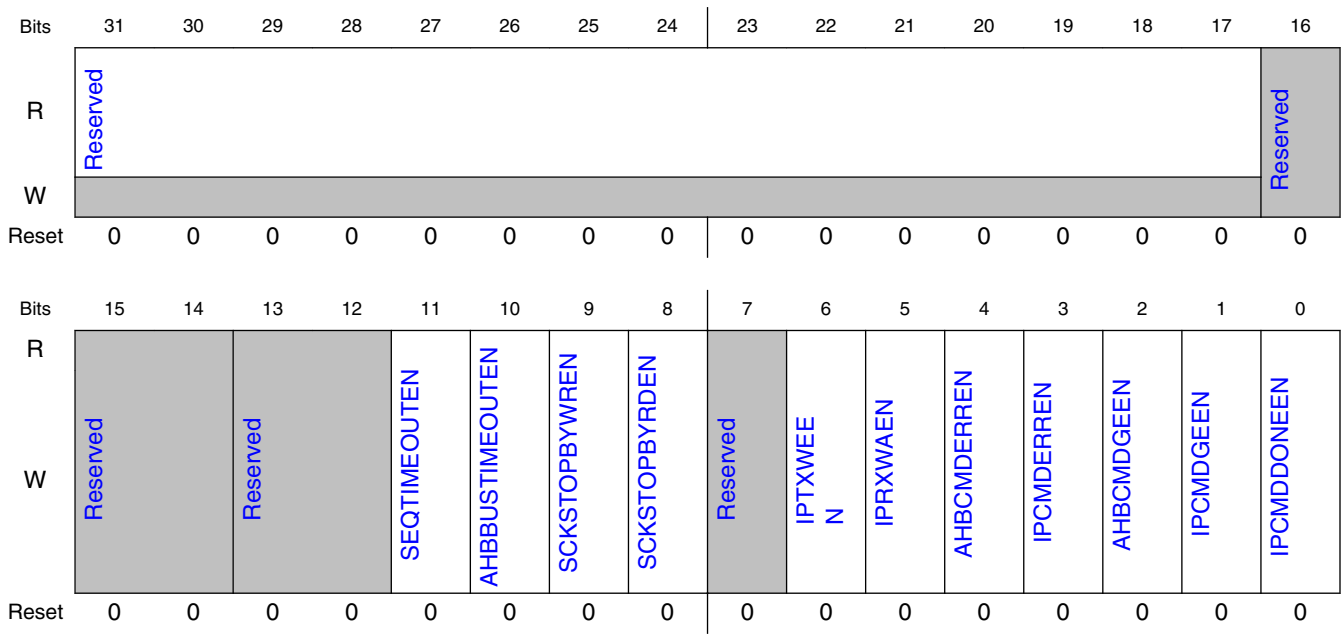
Field	Function
—	
10 READSZALIGN	AHB Read Size Alignment 0b - AHB read size will be decided by other register setting like PREFETCH_EN 1b - AHB read size to up size to 8 bytes aligned, no prefetching
9 —	Reserved
8 —	Reserved
7 —	Reserved
6 READADDROPT	AHB Read Address option bit. This option bit is intend to remove AHB burst start address alignment limitation. When FlexSPI controller is used for FPGA application, there may be requirement that FlexSPI fetch exactly the byte number as AHB burst. In this case, FPGA device should be designed as non-wordaddressable and this option bit should be set 0. 0b - There is AHB read burst start address alignment limitation when flash is accessed in parallel mode or flash is word-addressable. 1b - There is no AHB read burst start address alignment limitation. FlexSPI will fetch more data than AHB burst required to meet the alignment requirement.
5 PREFETCHEN	AHB Read Prefetch Enable. When AHB read prefetch is enabled, FlexSPI will fetch more flash read data than current AHB burst needed so that the read latency for next AHB read access will be reduced.
4 BUFFERABLEEN	Enable AHB bus bufferable write access support. This field affects the last beat of AHB write access, refer for more details about AHB bufferable write. 0b - Disabled. For all AHB write access (no matter bufferable or non-bufferable), FlexSPI will return AHB Bus ready after all data is transmitted to External device and AHB command finished. 1b - Enabled. For AHB bufferable write access, FlexSPI will return AHB Bus ready when the AHB command is granted by arbitrator and will not wait for AHB command finished.
3 CACHABLEEN	Enable AHB bus cachable read access support. 0b - Disabled. When there is AHB bus cachable read access, FlexSPI will not check whether it hit AHB TX Buffer. 1b - Enabled. When there is AHB bus cachable read access, FlexSPI will check whether it hit AHB TX Buffer first.
2 —	Reserved
1 —	Reserved
0 APAREN	Parallel mode enabled for AHB triggered Command (both read and write) . 0b - Flash will be accessed in Individual mode. 1b - Flash will be accessed in Parallel mode.

27.7.3.6 Interrupt Enable Register (INTEN)

27.7.3.6.1 Offset

Register	Offset
INTEN	10h

27.7.3.6.2 Diagram



27.7.3.6.3 Fields

Field	Function
31-17 —	Reserved
16 —	Reserved
15-14 —	Reserved
13-12 —	Reserved
11 SEQTIMEOUTEN	Sequence execution timeout interrupt enable.Refer Interrupts chapter for more details.
10	AHB Bus timeout interrupt.Refer Interrupts chapter for more details.

Table continues on the next page...

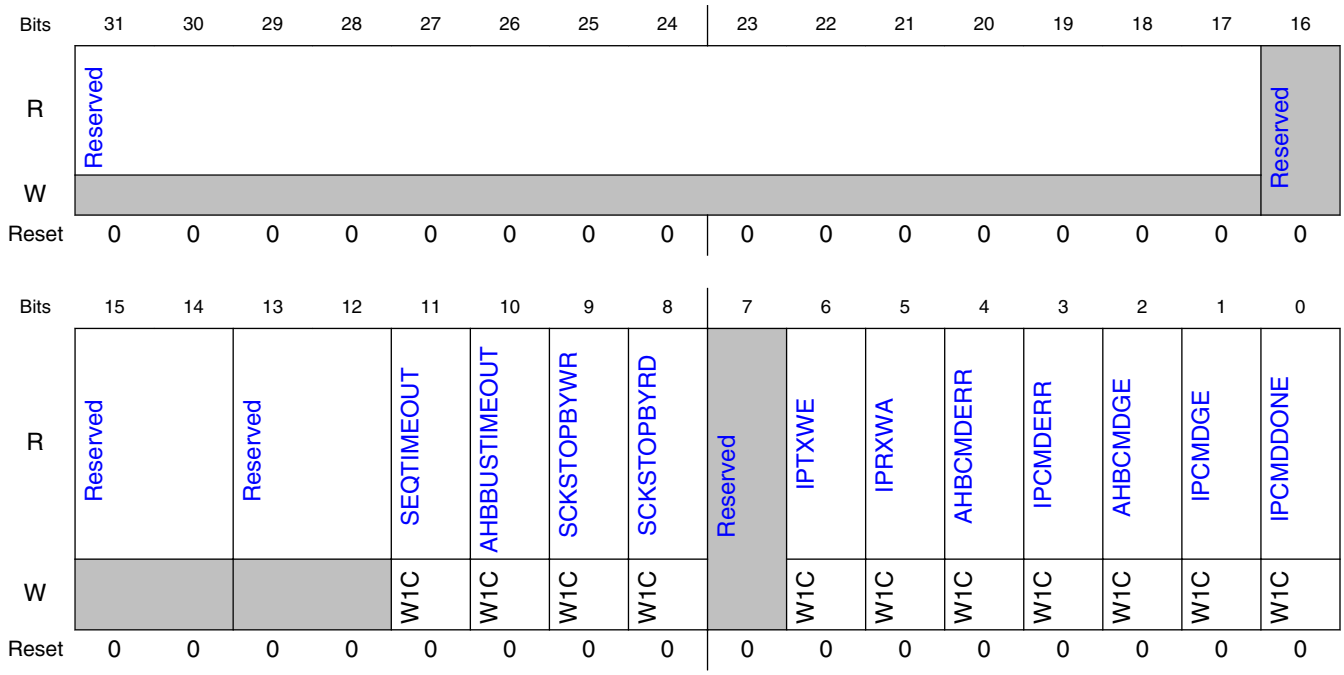
Field	Function
AHBBUSTIMEOUTEN	
9 SCKSTOPBYWREN	SCLK is stopped during command sequence because Async TX FIFO empty interrupt enable.
8 SCKSTOPBYRDEN	SCLK is stopped during command sequence because Async RX FIFO full interrupt enable.
7 —	Reserved
6 IPTXWEEN	IP TX FIFO WaterMark empty interrupt enable. IP TX FIFO has no less empty space than WaterMark level interrupt enable.
5 IPRXWAEN	IP RX FIFO WaterMark available interrupt enable. IP RX FIFO has no less valid data than WaterMark level interrupt enable.
4 AHBCMDERREN	AHB triggered Command Sequences Error Detected interrupt enable.
3 IPCMDERREN	IP triggered Command Sequences Error Detected interrupt enable.
2 AHBCMDGEEN	AHB triggered Command Sequences Grant Timeout interrupt enable.
1 IPCMDGEEN	IP triggered Command Sequences Grant Timeout interrupt enable.
0 IPCMDDONEEN	IP triggered Command Sequences Execution finished interrupt enable.

27.7.3.7 Interrupt Register (INTR)

27.7.3.7.1 Offset

Register	Offset
INTR	14h

27.7.3.7.2 Diagram



27.7.3.7.3 Fields

Field	Function
31-17 —	Reserved
16 —	Reserved
15-14 —	Reserved
13-12 —	Reserved
11 SEQTIMEOUT	Sequence execution timeout interrupt.
10 AHBBUSTIMEOUT	AHB Bus timeout interrupt.Refer Interrupts chapter for more details.
9 SCKSTOPBYWR	SCLK is stopped during command sequence because Async TX FIFO empty interrupt.
8 SCKSTOPBYRD	SCLK is stopped during command sequence because Async RX FIFO full interrupt.

Table continues on the next page...

Field	Function
7 —	Reserved
6 IPTXWE	IP TX FIFO watermark empty interrupt. IP TX FIFO has no less empty space than WaterMark level interrupt.
5 IPRXWA	IP RX FIFO watermark available interrupt. IP RX FIFO has no less valid data than WaterMark level interrupt.
4 AHBCMDERR	AHB triggered Command Sequences Error Detected interrupt. When an error detected for AHB command, this command will be ignored and not executed at all.
3 IPCMDERR	IP triggered Command Sequences Error Detected interrupt. When an error detected for IP command, this command will be ignored and not executed at all.
2 AHBCMDGE	AHB triggered Command Sequences Grant Timeout interrupt.
1 IPCMDGE	IP triggered Command Sequences Grant Timeout interrupt.
0 IPCMDDONE	IP triggered Command Sequences Execution finished interrupt. This interrupt is also generated when there is IPCMDGE or IPCMDERR interrupt generated.

27.7.3.8 LUT Key Register (LUTKEY)

27.7.3.8.1 Offset

Register	Offset
LUTKEY	18h

27.7.3.8.2 Function

The LUT Key Register contains the key to lock and unlock LUT. Refer to Look Up Table for details.

27.7.3.8.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	KEY															
Reset	0	1	0	1	1	0	1	0	1	1	1	1	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	KEY															
Reset	0	1	0	1	1	0	1	0	1	1	1	1	0	0	0	0

27.7.3.8.4 Fields

Field	Function
31-0 KEY	The Key to lock or unlock LUT. The key is 0x5AF05AF0. Read value is always 0x5AF05AF0.

27.7.3.9 LUT Control Register (LUTCR)

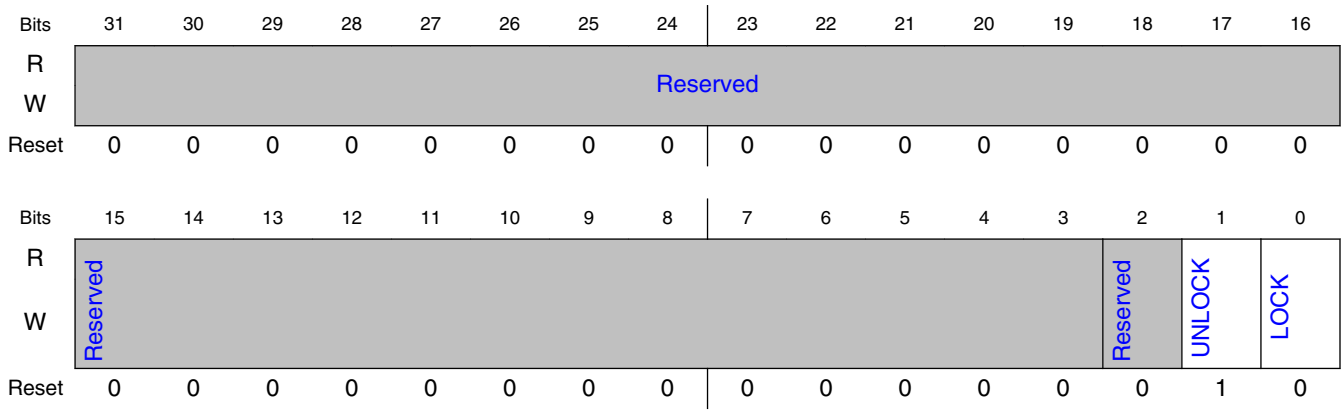
27.7.3.9.1 Offset

Register	Offset
LUTCR	1Ch

27.7.3.9.2 Function

The LUT control register is used along with LUTKEY register to lock or unlock LUT. This register has to be written immediately after writing 0x5AF05AF0 to LUTKEY register for the lock or unlock operation to be successful. Refer Look Up Table for details on locking/unlocking LUT. Setting both the LOCK and UNLOCK bits as "00" or "11" is not allowed.

27.7.3.9.3 Diagram



27.7.3.9.4 Fields

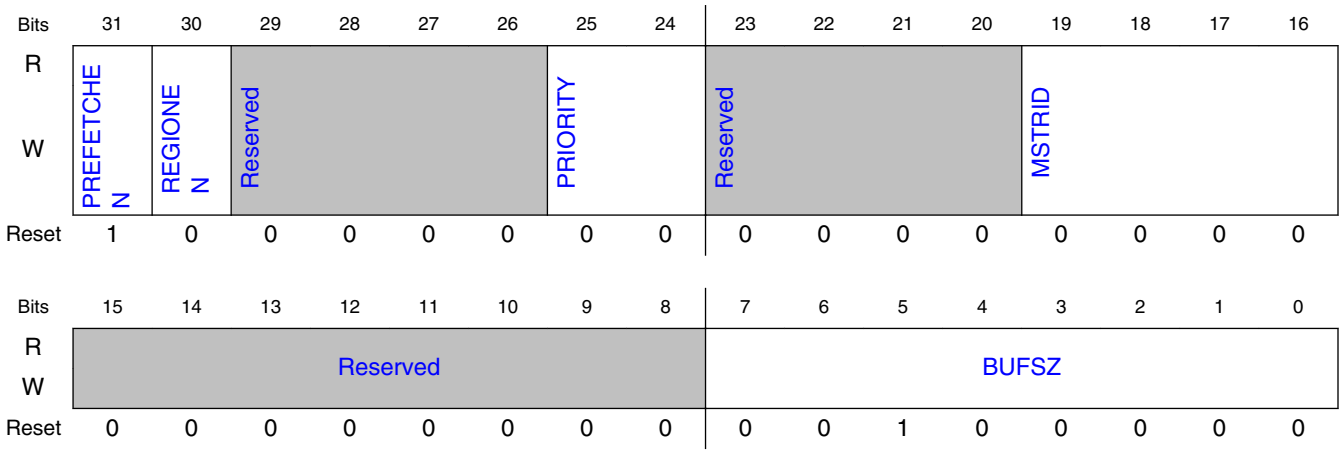
Field	Function
31-3 —	Reserved
2 —	Reserved
1 UNLOCK	Unlock LUT
0 LOCK	Lock LUT

27.7.3.10 AHB RX Buffer 0 Control Register 0 (AHBRXBUF0CR0)

27.7.3.10.1 Offset

Register	Offset
AHBRXBUF0CR0	20h

27.7.3.10.2 Diagram



27.7.3.10.3 Fields

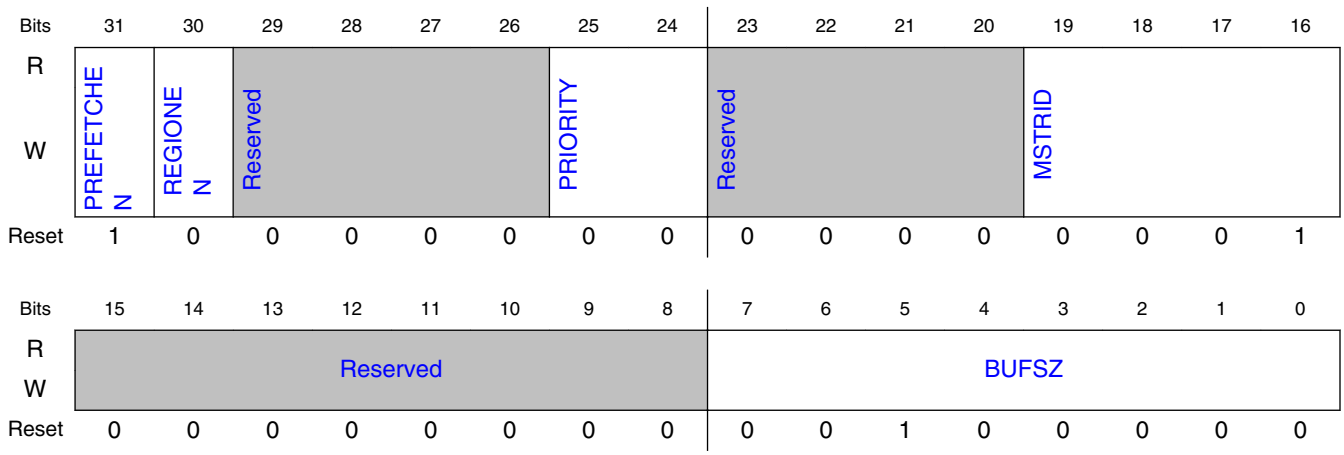
Field	Function
31 PREFETCHEN	AHB Read Prefetch Enable for current AHB RX Buffer corresponding Master. The prefetch feature is disabled when AHBCR[PREFETCHEN] is set 0. This field allows prefetch disable/ enable separately for each master.
30 REGIONEN	AHB RX Buffer address region funciton enable REGIONEN=1, hit current AHB buffer when ahb address is inside of range defined by AHBBUFREGIONiSTART and AHBBUFREGIONiEND. **NOTE:** Only AHBRXBUF0 TO 2 have this function
29-26 —	Reserved
25-24 PRIORITY	This priority for AHB Master Read which this AHB RX Buffer is assigned. 7 is the highest priority, 0 the lowest. Please refer to Command Abort and Suspend for more details.
23-20 —	Reserved
19-16 MSTRID	This AHB RX Buffer is assigned according to AHB Master with ID (MSTR_ID). Please refer to AHB RX Buffer Management for AHB RX Buffer allocation.
15-8 —	Reserved
7-0 BUFSZ	AHB RX Buffer Size in 64 bits. Please refer to AHB RX Buffer Management for more details.

27.7.3.11 AHB RX Buffer 1 Control Register 0 (AHBRXBUF1CR0)

27.7.3.11.1 Offset

Register	Offset
AHBRXBUF1CR0	24h

27.7.3.11.2 Diagram



27.7.3.11.3 Fields

Field	Function
31 PREFETCHEN	AHB Read Prefetch Enable for current AHB RX Buffer corresponding Master. The prefetch feature is disabled when AHBCR[PREFETCHEN] is set 0. This field allows prefetch disable/ enable separately for each master.
30 REGIONEN	AHB RX Buffer address region funciton enable REGIONEN=1, hit current AHB buffer when ahb address is inside of range defined by AHBBUFREGIONiSTART and AHBBUFREGIONiEND. **NOTE:** Only AHBRXBUF0 TO 2 have this function
29-26 —	Reserved
25-24 PRIORITY	This priority for AHB Master Read which this AHB RX Buffer is assigned. 7 is the highest priority, 0 the lowest. Please refer to Command Abort and Suspend for more details.
23-20 —	Reserved
19-16	This AHB RX Buffer is assigned according to AHB Master with ID (MSTR_ID).

Table continues on the next page...

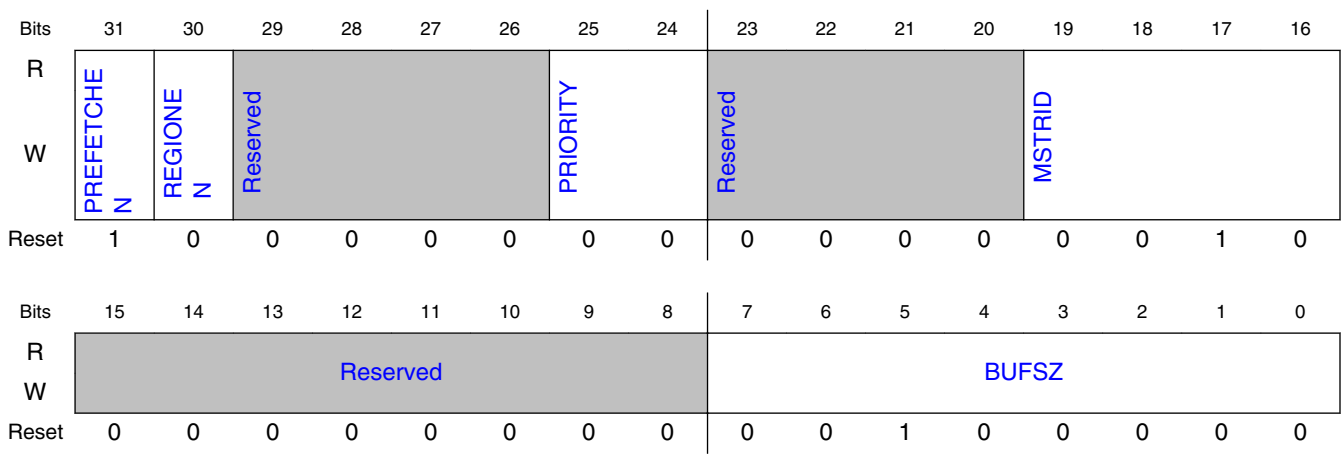
Field	Function
MSTRID	Please refer to AHB RX Buffer Management for AHB RX Buffer allocation.
15-8 —	Reserved
7-0 BUFSZ	AHB RX Buffer Size in 64 bits. Please refer to AHB RX Buffer Management for more details.

27.7.3.12 AHB RX Buffer 2 Control Register 0 (AHBRXBUF2CR0)

27.7.3.12.1 Offset

Register	Offset
AHBRXBUF2CR0	28h

27.7.3.12.2 Diagram



27.7.3.12.3 Fields

Field	Function
31 PREFETCHEN	AHB Read Prefetch Enable for current AHB RX Buffer corresponding Master. The prefetch feature is disabled when AHBCR[PREFETCHEN] is set 0. This field allows prefetch disable/enable separately for each master.
30 REGIONEN	AHB RX Buffer address region funciton enable REGIONEN=1, hit current AHB buffer when ahb address is inside of range defined by AHBBUFREGIONiSTART and AHBBUFREGIONiEND.

Table continues on the next page...

Field	Function
	NOTE: Only AHBRXBUF0 TO 2 have this function
29-26 —	Reserved
25-24 PRIORITY	This priority for AHB Master Read which this AHB RX Buffer is assigned. 7 is the highest priority, 0 the lowest. Please refer to Command Abort and Suspend for more details.
23-20 —	Reserved
19-16 MSTRID	This AHB RX Buffer is assigned according to AHB Master with ID (MSTR_ID). Please refer to AHB RX Buffer Management for AHB RX Buffer allocation.
15-8 —	Reserved
7-0 BUFSZ	AHB RX Buffer Size in 64 bits. Please refer to AHB RX Buffer Management for more details.

27.7.3.13 AHB RX Buffer 3 Control Register 0 (AHBRXBUF3CR0)

27.7.3.13.1 Offset

Register	Offset
AHBRXBUF3CR0	2Ch

27.7.3.13.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PREFETCHEN	REGIONEN	Reserved				PRIORITY		Reserved				MSTRID			
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								BUFSZ							
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

27.7.3.13.3 Fields

Field	Function
31 PREFETCHEN	AHB Read Prefetch Enable for current AHB RX Buffer corresponding Master. The prefetch feature is disabled when AHBCR[PREFETCHEN] is set 0. This field allows prefetch disable/ enable separately for each master.
30 REGIONEN	AHB RX Buffer address region funciton enable REGIONEN=1, hit current AHB buffer when ahb address is inside of range defined by AHBBUFREGIONiSTART and AHBBUFREGIONiEND. **NOTE:** Only AHBRXBUF0 TO 2 have this function
29-26 —	Reserved
25-24 PRIORITY	This priority for AHB Master Read which this AHB RX Buffer is assigned. 7 is the highest priority, 0 the lowest. Please refer to Command Abort and Suspend for more details.
23-20 —	Reserved
19-16 MSTRID	This AHB RX Buffer is assigned according to AHB Master with ID (MSTR_ID). Please refer to AHB RX Buffer Management for AHB RX Buffer allocation.
15-8 —	Reserved
7-0 BUFSZ	AHB RX Buffer Size in 64 bits. Please refer to AHB RX Buffer Management for more details.

27.7.3.14 Flash Control Register 0 (FLSHA1CR0 - FLSHB2CR0)

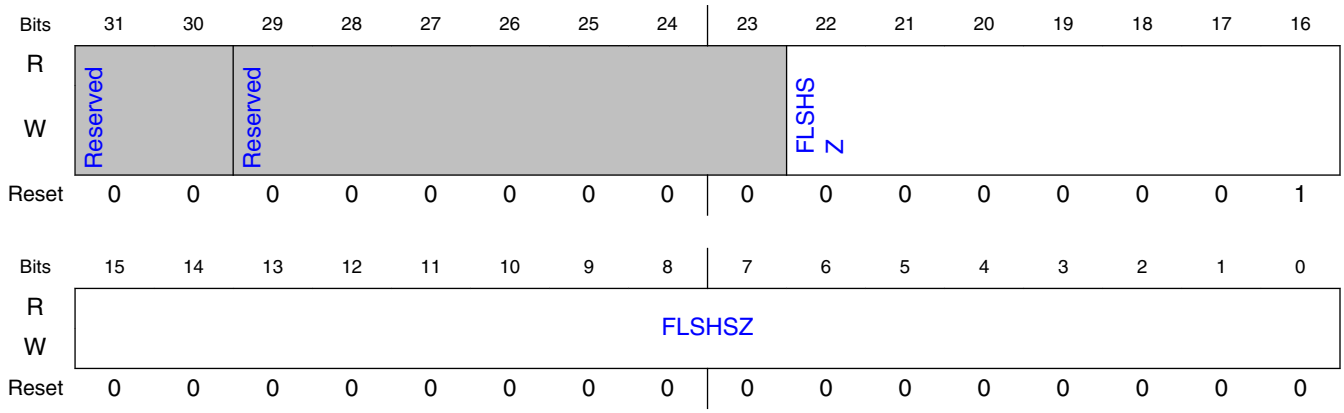
27.7.3.14.1 Offset

Register	Offset
FLSHA1CR0	60h
FLSHA2CR0	64h
FLSHB1CR0	68h
FLSHB2CR0	6Ch

27.7.3.14.2 Function

The Flash control register 0 contains Flash size setting. FlexSPI determines which device is accessed with this register setting (Chip Selection).

27.7.3.14.3 Diagram



27.7.3.14.4 Fields

Field	Function
31-30 —	Reserved
29-23 —	Reserved
22-0 FLSHSZ	Flash Size in KByte. The max flash size supported for each device is 512 MB. When FLSHSZ setting value is greater than 0x400000, the device flash size would be taken as 512 MB. The max total flash size supported (for all 4 devices) is also 512 MB. If the total flash size is larger than 512 MB, only 512 MB address space is accessible.

27.7.3.15 Flash Control Register 1 (FLSHA1CR1 - FLSHB2CR1)

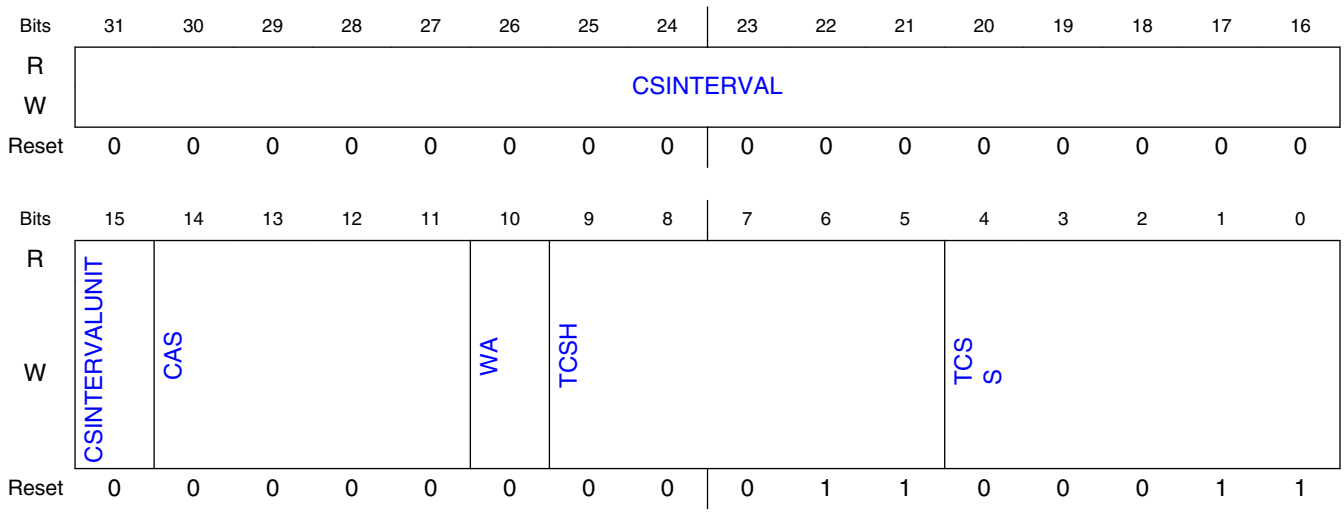
27.7.3.15.1 Offset

Register	Offset
FLSHA1CR1	70h
FLSHA2CR1	74h
FLSHB1CR1	78h
FLSHB2CR1	7Ch

27.7.3.15.2 Function

The Flash control register 1 contains the setting for FlexSPI to meet Flash device specific timings and Flash internal address space.

27.7.3.15.3 Diagram



27.7.3.15.4 Fields

Field	Function
31-16 CSINTERVAL	This field is used to set the minimum interval between flash device Chip selection deassertion and flash device Chip selection assertion. If external flash has a limitation on the interval between command sequences, this field should be set accordingly. If there is no limitation, set this field with value 0x0. When CSINTERVALUNIT is 0x0, the chip selection invalid interval is: CSINTERVAL * 1 serial clock cycle; When CSINTERVALUNIT is 0x1, the chip selection invalid interval is: CSINTERVAL * 256 serial clock cycle. **NOTE:** The chip selection interval is 2 cycle at least even if CSINTERVAL is less than 2.
15 CSINTERVALUNIT	CS interval unit 0b - The CS interval unit is 1 serial clock cycle 1b - The CS interval unit is 256 serial clock cycle
14-11 CAS	Column Address Size. When external flash has separate address field for row address and column address, this field should be set to flash column address bit width. FlexSPI will automatically split flash mapped address to Row address and Column address according to CAS field and WA field setting. This bit should be set to 0x0 when external Flash don't support column address. FlexSPI will transmit all flash address bits as Row address. For flash address mapping, please refer to Flash memory map for more detail.
10 WA	Word Addressable. This bit should be set when external Flash is word addressable. If Flash is word addressable, it should be access in terms of 16 bits. At this time, FlexSPI will not transmit Flash address bit 0 to external Flash. For flash address mapping, please refer to Flash memory map for more detail.
9-5	Serial Flash CS Hold time.

Table continues on the next page...

Field	Function
TCSH	This field is used to meet flash TCSH timing requirement. Serial flash CS Hold time promised by FlexSPI is: TCSH in serial root clock cycles (for both SDR and DDR mode). Please refer to FlexSPI Input Timing for more detail.
4-0 TCSS	Serial Flash CS setup time. This field is used to meet flash TCSS timing requirement. Serial flash CS Setup time promised by FlexSPI is: (TCSS + 1/2) serial root clock cycles (for both SDR and DDR mode). Please refer to FlexSPI Input Timing for more detail.

27.7.3.16 Flash Control Register 2 (FLSHA1CR2 - FLSHB2CR2)

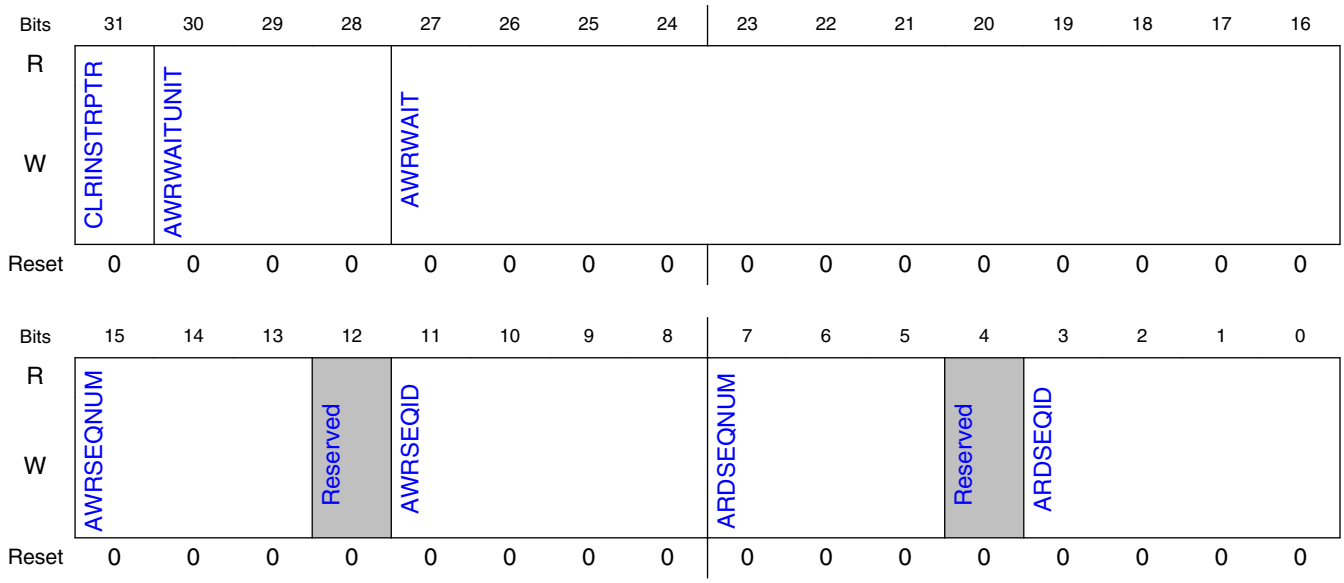
27.7.3.16.1 Offset

Register	Offset
FLSHA1CR2	80h
FLSHA2CR2	84h
FLSHB1CR2	88h
FLSHB2CR2	8Ch

27.7.3.16.2 Function

Flash Control Register 2 contains setting field for AHB Bus access configuration. If the 4 external device are in different types, AHB read/write command may use different command sequences and AHB bus ready wait time may be also different.

27.7.3.16.3 Diagram



27.7.3.16.4 Fields

Field	Function
31 CLRINSTRPTR	Clear the instruction pointer which is internally saved pointer by JMP_ON_CS. Refer Programmable Sequence Engine for details. This field is used for AHB Read access to external Flash supporting XIP (Execute-In-Place) mode.
30-28 AWRWAITUNIT	AWRWAIT unit 000b - The AWRWAIT unit is 2 ahb clock cycle 001b - The AWRWAIT unit is 8 ahb clock cycle 010b - The AWRWAIT unit is 32 ahb clock cycle 011b - The AWRWAIT unit is 128 ahb clock cycle 100b - The AWRWAIT unit is 512 ahb clock cycle 101b - The AWRWAIT unit is 2048 ahb clock cycle 110b - The AWRWAIT unit is 8192 ahb clock cycle 111b - The AWRWAIT unit is 32768 ahb clock cycle
27-16 AWRWAIT	For certain devices (such as FPGA), it need some time to write data into internal memory after the command sequences finished on FlexSPI interface. If another Read command sequence comes before previous programming finished internally, the read data may be wrong. This field is used to hold AHB Bus ready for AHB write access to wait the programming finished in external device. Then there will be no AHB read command triggered before the programming finished in external device. The Wait cycle between AHB triggered command sequences finished on FlexSPI interface and AHB return Bus ready: AWRWAIT * AWRWAITUNIT
15-13 AWRSEQNUM	Sequence Number for AHB Write triggered Command. For certain flash devices , a Flash programming access is done by several command sequences. AHB write Command will trigger (AWRSEQNUM+1) command sequences to external flash every time. FlexSPI executes the sequences in LUT incrementally.

Table continues on the next page...

Field	Function
	NOTE: • Software should make sure the last sequence index never exceed LUT sequence numbers: AWRSEQID+AWRSEQNUM < 16 • Software need to make sure field AWRSEQNUM and LUT is configured correctly according to external device spec. FlexSPI don't check the sequence and just execute them one by one.
12 —	Reserved
11-8 AWRSEQID	Sequence Index for AHB Write triggered Command.
7-5 ARDSEQNUM	Sequence Number for AHB Read triggered Command in LUT. For certain flash devices , a Flash reading access is done by several command sequences. AHB read Command will trigger (ARDSEQNUM+1) command sequences to external flash every time. FlexSPI executes the sequences in LUT incrementally. **NOTE:** • Software should make sure the last sequence index never exceed LUT sequence numbers: ARDSEQID+ARDSEQNUM <= 16 • Software need to make sure field ARDSEQNUM and LUT is configured correctly according to external device spec. FlexSPI don't check the sequence and just execute them one by one.
4 —	Reserved
3-0 ARDSEQID	Sequence Index for AHB Read triggered Command in LUT.

27.7.3.17 Flash Control Register 4 (FLSHCR4)

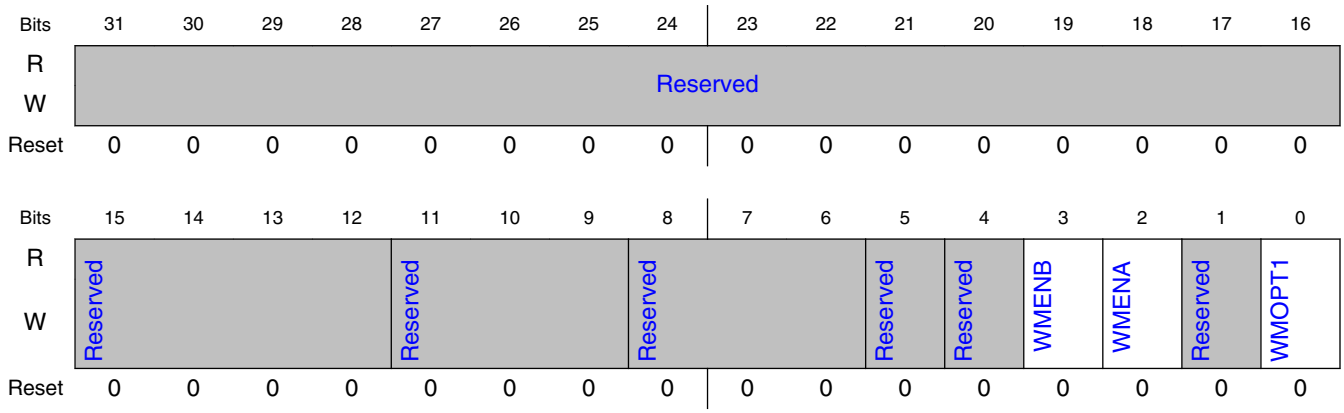
27.7.3.17.1 Offset

Register	Offset
FLSHCR4	94h

27.7.3.17.2 Function

The flash control register 4 provide the configuration for all external devices.

27.7.3.17.3 Diagram



27.7.3.17.4 Fields

Field	Function
31-12 —	Reserved
11-9 —	Reserved
8-6 —	Reserved
5 —	Reserved
4 —	Reserved
3 WMENB	Write mask enable bit for flash device on port B. When write mask function is needed for memory device on port B, this bit must be set. 0b - Write mask is disabled, DQS(RWDS) pin will be un-driven when writing to external device. 1b - Write mask is enabled, DQS(RWDS) pin will be driven by FlexSPI as write mask output when writing to external device.
2 WMENA	Write mask enable bit for flash device on port A. When write mask function is needed for memory device on port A, this bit must be set. 0b - Write mask is disabled, DQS(RWDS) pin will be un-driven when writing to external device. 1b - Write mask is enabled, DQS(RWDS) pin will be driven by FlexSPI as write mask output when writing to external device.
1 —	Reserved
0 WMOPT1	Write mask option bit 1. This option bit could be used to remove AHB write burst start address alignment limitation. 0b - DQS pin will be used as Write Mask when writing to external device. There is no limitation on AHB write burst start address alignment when flash is accessed in individual mode. 1b - DQS pin will not be used as Write Mask when writing to external device. There is limitation on AHB write burst start address alignment when flash is accessed in individual mode.

27.7.3.18 IP Control Register 0 (IPCR0)

27.7.3.18.1 Offset

Register	Offset
IPCR0	A0h

27.7.3.18.2 Function

The IP control registers provide all the configuration required for IP commands. This register provides the flash device's start address, instead of SoC address, to be accessed for IP command. FlexSPI will determine the chip select automatically according to this start address.

NOTE

- It's not allowed to issue IP command crossing Flash device boundaries. Otherwise there will be IPCMDERR interrupt generated.
- This register should be set before IP command triggered.
- This register setting should not be changed while an IP command is in progress.

27.7.3.18.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SFAR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SFAR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

27.7.3.18.4 Fields

Field	Function
31-0	Serial Flash Address for IP command.

Field	Function
SFAR	

27.7.3.19 IP Control Register 1 (IPCR1)

27.7.3.19.1 Offset

Register	Offset
IPCR1	A4h

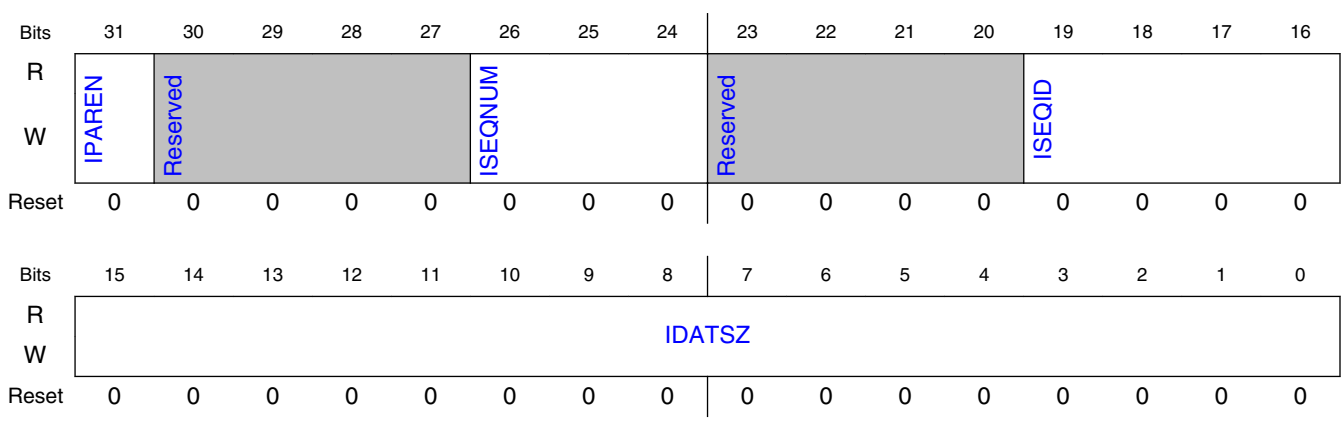
27.7.3.19.2 Function

The IP control registers provide all the configuration required for IP command. This register provides the flash read/program data size, sequence index in LUT, sequence number and individual/parallel mode setting for IP command.

NOTE

- This register should be before IP command triggered.
- This register setting should not be changed before IP command finished.

27.7.3.19.3 Diagram



27.7.3.19.4 Fields

Field	Function
31	Parallel mode Enabled for IP command.

Table continues on the next page...

Field	Function
IPAREN	0b - Flash will be accessed in Individual mode. 1b - Flash will be accessed in Parallel mode.
30-27 —	Reserved
26-24 ISEQNUM	Sequence Number for IP command: ISEQNUM+1.
23-20 —	Reserved
19-16 ISEQID	Sequence Index in LUT for IP command.
15-0 IDATSZ	Flash Read/Program Data Size (in Bytes) for IP command.

27.7.3.20 IP Command Register (IPCMD)

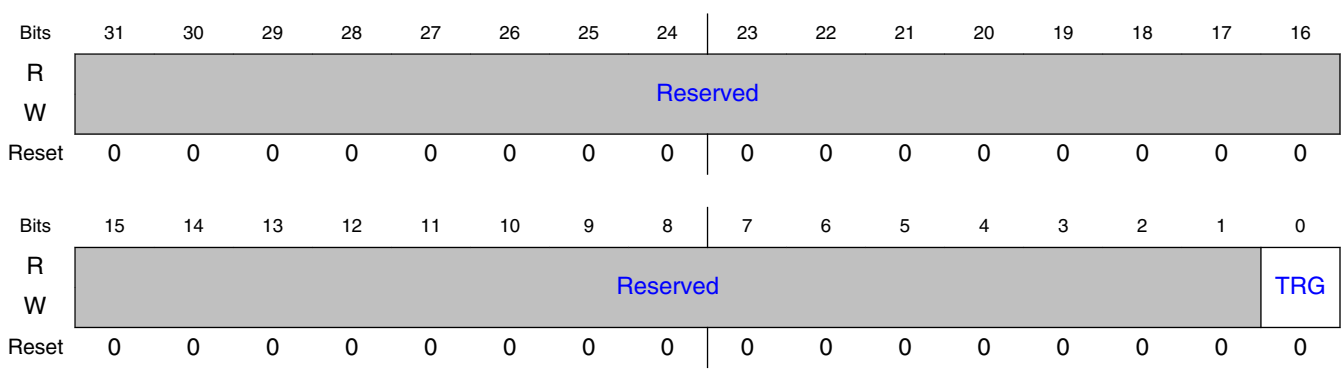
27.7.3.20.1 Offset

Register	Offset
IPCMD	B0h

27.7.3.20.2 Function

This register is used to trigger a IP command to access external flash device. IP command will be executed on FlexSPI interface after granted by arbitrator.

27.7.3.20.3 Diagram



27.7.3.20.4 Fields

Field	Function
31-1 —	Reserved
0 TRG	Setting this bit will trigger an IP Command. **NOTE:** • It's not allowed to trigger another IP command before previous IP command is finished on FlexSPI interface. Software need to poll register bit INTR_IP_CMD_DONE or wait for this interrupt in order to wait for IP command finished.

27.7.3.21 IP RX FIFO Control Register (IPRXFCR)

27.7.3.21.1 Offset

Register	Offset
IPRXFCR	B8h

27.7.3.21.2 Function

This register provides the configuration fields for IP RX FIFO management.

27.7.3.21.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved										RXWMRK				RXDMAEN	CLRIPRXF
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

27.7.3.21.4 Fields

Field	Function
31-6	Reserved

Table continues on the next page...

Field	Function
—	
5-2 RXWMRK	Watermark level is (RXWMRK+1)*64 Bits. Interrupt register bit IPRXWA is set when filling level in IP RX FIFO is no less than Watermark level by FlexSPI. There will be a DMA request when the filling level is no less than Watermark level and DMA read is enabled (register bit RXDMAEN is set). There will be an IPRXWA (IP RX FIFO watermark available) interrupt generated when the filling level is no less than Watermark level and IPRXWA interrupt is enabled (register bit INTEN_IPRXWA is set). **NOTE:** After write-1-clear to INTR[IPRXWA], read address should be rolled back to start address (memory mapped). If IP RX FIFO is read by IP bus, the read address to IP RX FIFO should roll back to RFDR0; If IP RX FIFO is read by AHB bus, the read address to IP RX FIFO should roll back to ARDF_BASE.
1 RXDMAEN	IP RX FIFO reading by DMA enabled. 0b - IP RX FIFO would be read by processor. 1b - IP RX FIFO would be read by DMA.
0 CLRIPRXF	Clear all valid data entries in IP RX FIFO. The read/write pointers in IP RX FIFO will be reset.

27.7.3.22 IP TX FIFO Control Register (IPTXFCR)

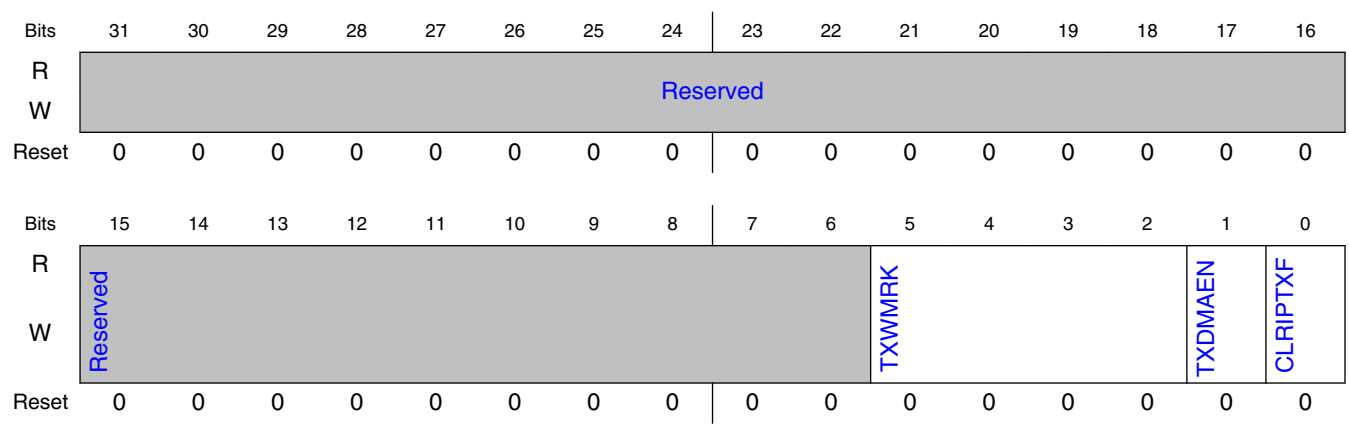
27.7.3.22.1 Offset

Register	Offset
IPTXFCR	BCh

27.7.3.22.2 Function

This register provides the configuration fields for IP TX FIFO management.

27.7.3.22.3 Diagram



27.7.3.22.4 Fields

Field	Function
31-6 —	Reserved
5-2 TXWMRK	Watermark level is (TXWMRK+1)*64 Bits. Interrupt register bit IPTXWE is set when empty level in IP TX FIFO is no less than Watermark level by FlexSPI. There will be a DMA request when empty level is no less than Watermark level and DMA filling is enable (register bit TXDMAEN is set). There will be an IPTXWE (IP TX FIFO Watermark Empty) interrupt generated when empty level is no less than Watermark level and IPTXWE interrupt is enable (register bit INTEN_IPTXWE is set). **NOTE:** • The watermark level should be no more than the write window. • The watermark level should be no more than IP TX FIFO size. • The write address to IP RX FIFO should roll back to the start address of write window after pushing IP TX FIFO by writing-one-clear to INTR[IPTXWE].
1 TXDMAEN	IP TX FIFO filling by DMA enabled. 0b - IP TX FIFO would be filled by processor. 1b - IP TX FIFO would be filled by DMA.
0 CLRIPTXF	Clear all valid data entries in IP TX FIFO. The read/write pointers in IP TX FIFO will be reset.

27.7.3.23 DLL Control Register 0 (DLLACR - DLLBCR)

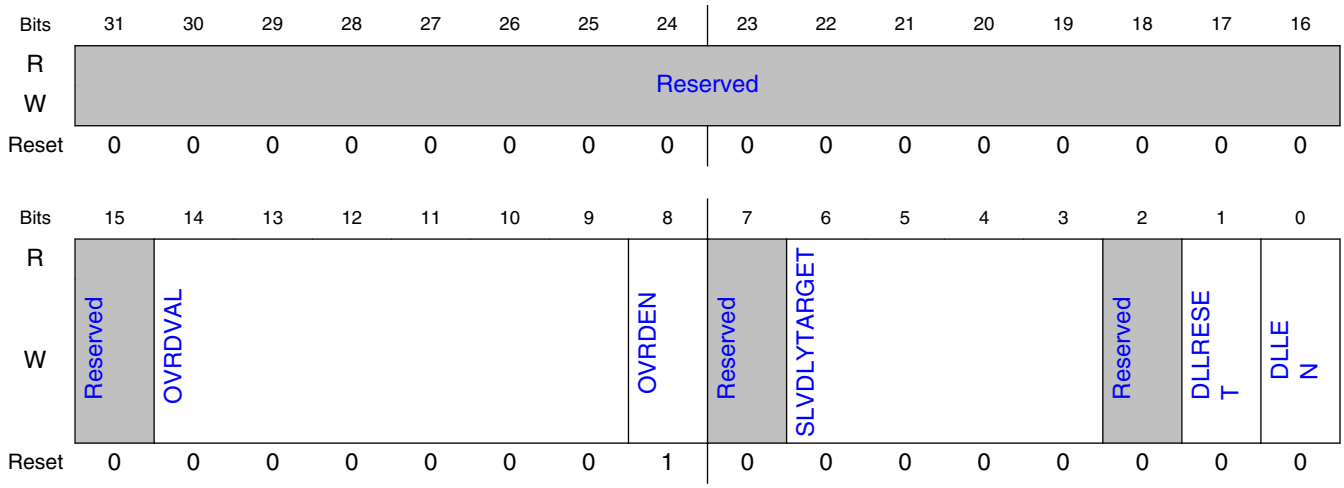
27.7.3.23.1 Offset

Register	Offset
DLLACR	C0h
DLLBCR	C4h

27.7.3.23.2 Function

This register provides the configuration fields for Flash A/B sample clock DLL.

27.7.3.23.3 Diagram



27.7.3.23.4 Fields

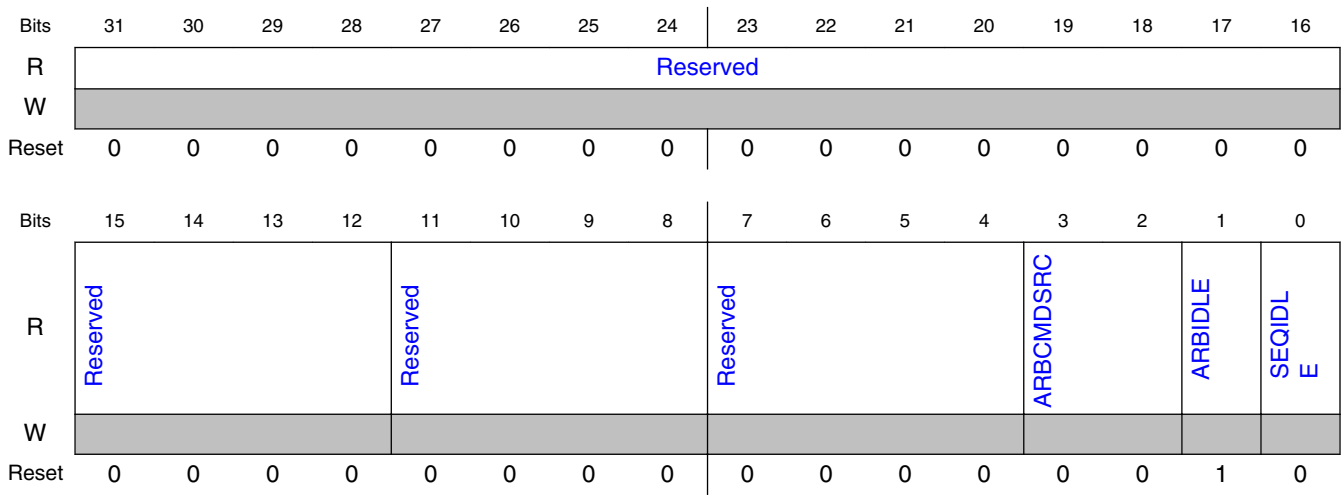
Field	Function
31-15 —	Reserved
14-9 OVRDVAL	Slave clock delay line delay cell number selection override value. When OVRDEN is set 0x1, the delay cell number in DLL is OVRDVAL+1.
8 OVRDEN	Slave clock delay line delay cell number selection override enable.
7 —	Reserved
6-3 SLVDLYTARGET	The delay target for slave delay line is: ((SLVDLYTARGET+1) * 1/32 * clock cycle of reference clock (serial root clock). If serial root clock is >= 100 MHz, DLLEN set to 0x1, OVRDEN set to =0x0, then SLVDLYTARGET setting of 0xF is recommended.
2 —	Reserved
1 DLLRESET	Software could force a reset on DLL by setting this field to 0x1. This will cause the DLL to lose lock and re-calibrate to detect an ref_clock half period phase shift. The reset action is edge triggered, so software need to clear this bit after set this bit (no delay limitation).
0 DLLEN	DLL calibration enable. When this bit is cleared, DLL calibration is disabled and the delay cell number in slave delay line is always 1. Please note that SLV delay line is overridden when OVRDEN bit is set and this bit field setting is ignored.

27.7.3.24 Status Register 0 (STS0)

27.7.3.24.1 Offset

Register	Offset
STS0	E0h

27.7.3.24.2 Diagram



27.7.3.24.3 Fields

Field	Function
31-12 —	Reserved
11-8 —	Reserved
7-4 —	Reserved
3-2 ARBCMDSRC	This status field indicates the trigger source of current command sequence granted by arbitrator. This field value is meaningless when ARB_CTL is not busy (STS0[ARBIDLE]=0x1). 00b - Triggered by AHB read command (triggered by AHB read). 01b - Triggered by AHB write command (triggered by AHB Write). 10b - Triggered by IP command (triggered by setting register bit IPCMD.TRG). 11b - Triggered by suspended command (resumed).
1 ARBIDLE	This status bit indicates the state machine in ARB_CTL is busy and there is command sequence granted by arbitrator and not finished yet on FlexSPI interface. When ARB_CTL state (ARBIDLE=0x1) is idle, there will be no transaction on FlexSPI interface also (SEQIDLE=0x1). So this bit should be polled to wait for FlexSPI controller become idle instead of SEQIDLE.

Table continues on the next page...

Field	Function
0 SEQIDLE	This status bit indicates the state machine in SEQ_CTL is idle and there is command sequence executing on FlexSPI interface.

27.7.3.25 Status Register 1 (STS1)

27.7.3.25.1 Offset

Register	Offset
STS1	E4h

27.7.3.25.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved				IPCMDERRCODE				Reserved				IPCMDERRID			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				AHBCMDERRCODE				Reserved				AHBCMDERRID			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

27.7.3.25.3 Fields

Field	Function
31-28 —	Reserved
27-24 IPCMDERRCODE	Indicates the Error Code when IP command Error detected. This field will be cleared when INTR[IPCMDERR] is write-1-clear(w1c). 0000b - No error. 0010b - IP command with JMP_ON_CS instruction used in the sequence.

Table continues on the next page...

Field	Function
	0011b - There is unknown instruction opcode in the sequence. 0100b - Instruction DUMMY_SDR/DUMMY_RWDS_SDR used in DDR sequence. 0101b - Instruction DUMMY_DDR/DUMMY_RWDS_DDR used in SDR sequence. 0110b - Flash access start address exceed the whole flash address range (A1/A2/B1/B2). 1110b - Sequence execution timeout. 1111b - Flash boundary crossed.
23-20 —	Reserved
19-16 IPCMDERRID	Indicates the sequence Index when IP command error detected. This field will be cleared when INTR[IPCMDERR] is write-1-clear(w1c).
15-12 —	Reserved
11-8 AHBCMDERRCODE	Indicates the Error Code when AHB command Error detected. This field will be cleared when INTR[AHBCMDERR] is write-1-clear(w1c). 0000b - No error. 0010b - AHB Write command with JMP_ON_CS instruction used in the sequence. 0011b - There is unknown instruction opcode in the sequence. 0100b - Instruction DUMMY_SDR/DUMMY_RWDS_SDR used in DDR sequence. 0101b - Instruction DUMMY_DDR/DUMMY_RWDS_DDR used in SDR sequence. 1110b - Sequence execution timeout.
7-4 —	Reserved
3-0 AHBCMDERRID	Indicates the sequence index when an AHB command error is detected. This field will be cleared when INTR[AHBCMDERR] is write-1-clear(w1c).

27.7.3.26 Status Register 2 (STS2)

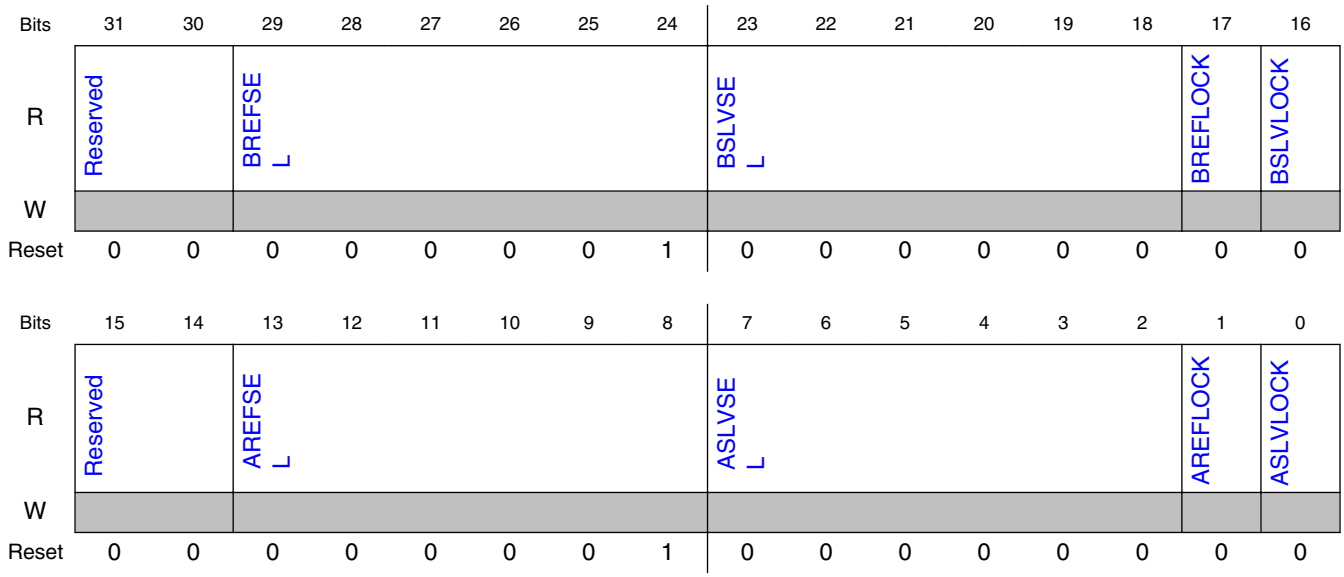
27.7.3.26.1 Offset

Register	Offset
STS2	E8h

27.7.3.26.2 Function

This register indicates the status of Flash A and B sample clock DLLs.

27.7.3.26.3 Diagram



27.7.3.26.4 Fields

Field	Function
31-30 —	Reserved
29-24 BREFSEL	Flash B sample clock reference delay line delay cell number selection.
23-18 BSLVSEL	Flash B sample clock slave delay line delay cell number selection.
17 BREFLOCK	Flash B sample clock reference delay line locked.
16 BSLVLOCK	Flash B sample clock slave delay line locked.
15-14 —	Reserved
13-8 AREFSEL	Flash A sample clock reference delay line delay cell number selection.
7-2 ASLVSEL	Flash A sample clock slave delay line delay cell number selection .
1 AREFLOCK	Flash A sample clock reference delay line locked.
0 ASLVLOCK	Flash A sample clock slave delay line locked.

27.7.3.27 AHB Suspend Status Register (AHBSPNDSTS)

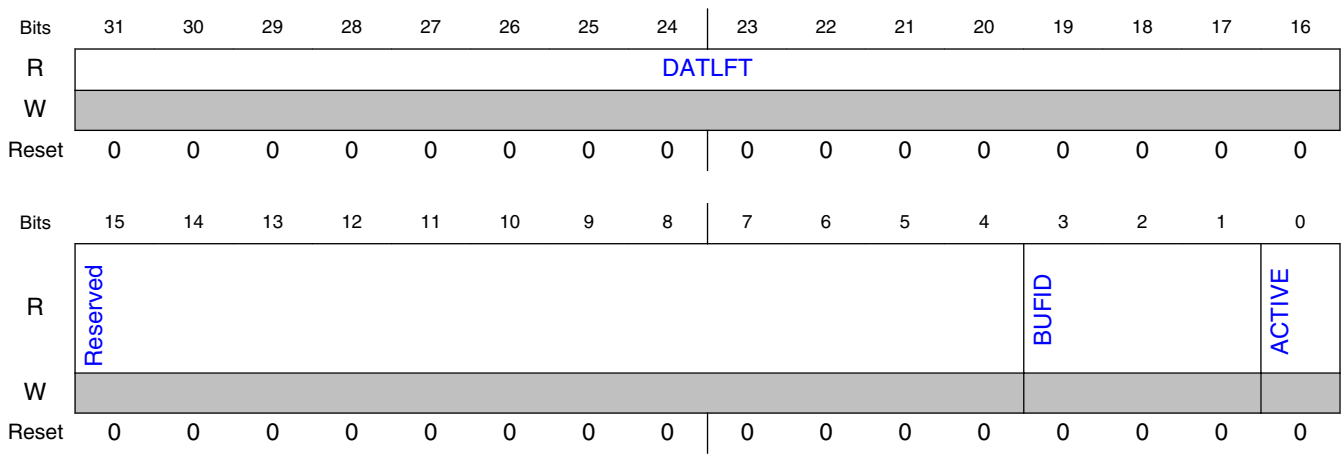
27.7.3.27.1 Offset

Register	Offset
AHBSPNDSTS	ECh

27.7.3.27.2 Function

Indicates the status of Suspended AHB Read Prefetch command sequence. When there is IP/AHB command triggered and arbitrator is processing an AHB Read sequence (prefetching more data not for current AHB burst), the prefetch sequence will be suspended and may be resumed when there is no transaction on FlexSPI any more. FlexSPI saves only one AHB prefetch sequence. When a new prefetch sequence is suspended with an active sequence suspended already, previous suspended sequence will be removed and never resumed. Refer Command Abort and Suspend for more details.

27.7.3.27.3 Diagram



27.7.3.27.4 Fields

Field	Function
31-16 DATLFT	Left Data size for suspended command sequence (in byte).
15-4	Reserved

Table continues on the next page...

Field	Function
—	
3-1 BUFID	AHB RX BUF ID for suspended command sequence.
0 ACTIVE	Indicates if an AHB read prefetch command sequence has been suspended.

27.7.3.28 IP RX FIFO Status Register (IPRXFSTS)

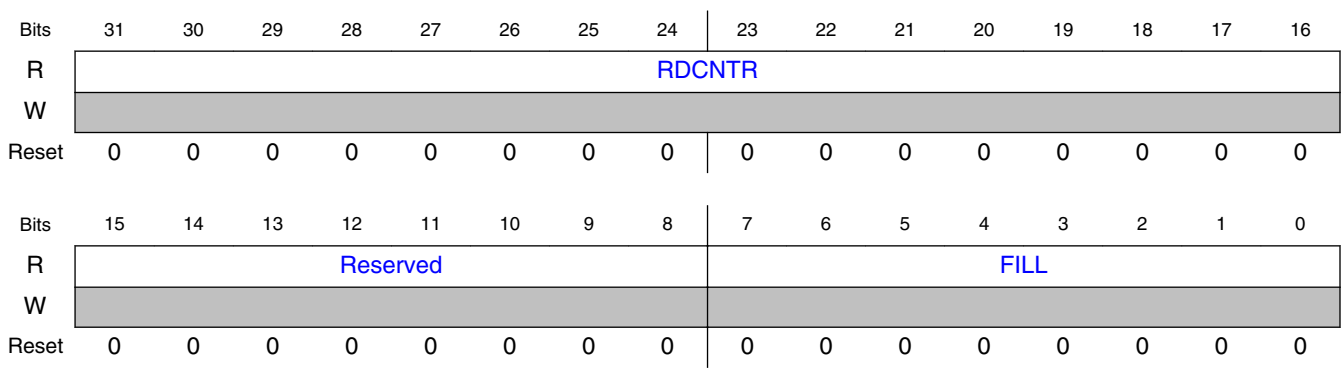
27.7.3.28.1 Offset

Register	Offset
IPRXFSTS	F0h

27.7.3.28.2 Function

This status register indicates the status of IP RX FIFO.

27.7.3.28.3 Diagram



27.7.3.28.4 Fields

Field	Function
31-16 RDCNTR	Total Read Data Counter: RDCNTR * 64 Bits.
15-8 —	Reserved

Table continues on the next page...

Field	Function
7-0 FILL	Fill level of IP RX FIFO. Valid Data entries in IP RX FIFO is: FILL * 64 Bits.

27.7.3.29 IP TX FIFO Status Register (IPTXFSTS)

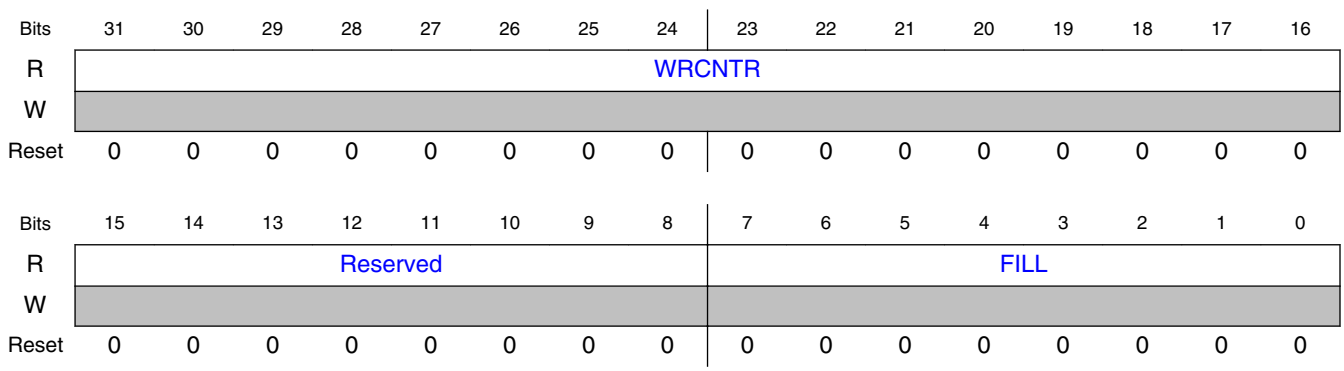
27.7.3.29.1 Offset

Register	Offset
IPTXFSTS	F4h

27.7.3.29.2 Function

This status register indicates the status of IP TX FIFO.

27.7.3.29.3 Diagram



27.7.3.29.4 Fields

Field	Function
31-16 WRCNTR	Total Write Data Counter: WRCNTR * 64 Bits.
15-8 —	Reserved
7-0 FILL	Fill level of IP TX FIFO. Valid Data entries in IP TX FIFO is: FILL * 64 Bits.

27.7.3.30 IP RX FIFO Data Register a (RFDR0 - RFDR31)

27.7.3.30.1 Offset

For a = 0 to 31:

Register	Offset
RFDRa	100h + (a × 4h)

27.7.3.30.2 Function

These registers provide read access to IP RX FIFO by IPS bus. The read value is unknown for read access to invalid entries in IP RX FIFO.

27.7.3.30.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RXDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RXDATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

27.7.3.30.4 Fields

Field	Function
31-0 RXDATA	RX Data

27.7.3.31 IP TX FIFO Data Register a (TFDR0 - TFDR31)

27.7.3.31.1 Offset

For a = 0 to 31:

Register	Offset
TFDRa	180h + (a × 4h)

27.7.3.31.2 Function

These registers provide write access to IP TX FIFO by IPS bus.

27.7.3.31.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	TXDATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	TXDATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

27.7.3.31.4 Fields

Field	Function
31-0 TXDATA	TX Data

27.7.3.32 LUT a (LUT0 - LUT63)

27.7.3.32.1 Offset

For a = 0 to 63:

Register	Offset
LUTa	200h + (a × 4h)

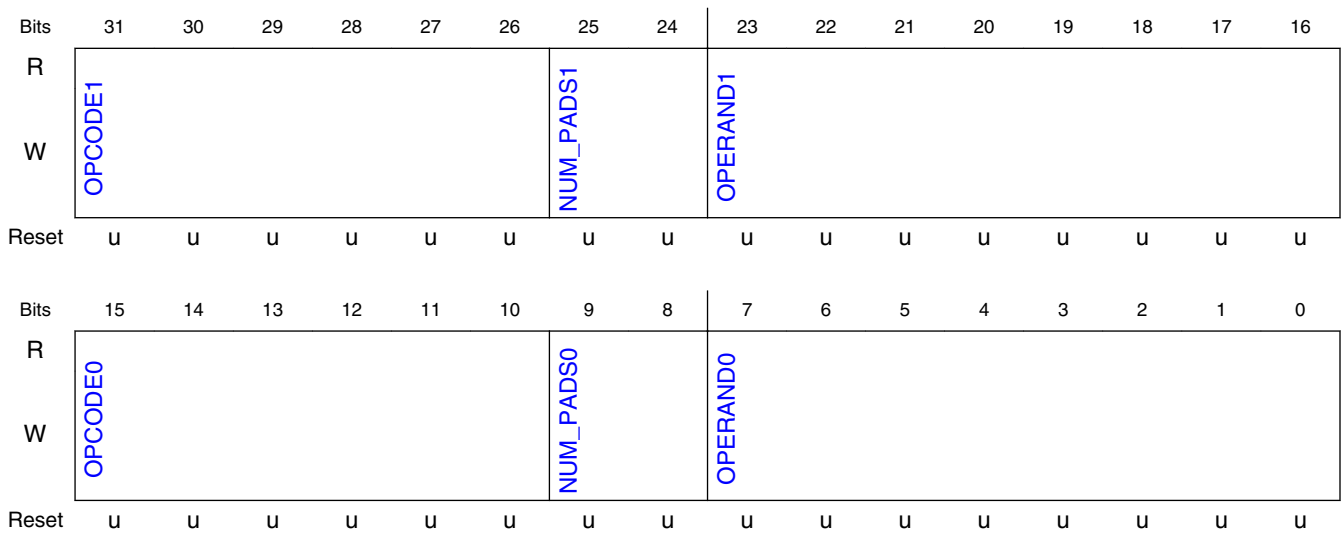
27.7.3.32.2 Function

The LUT is a look-up-table for command sequences. Software should set the sequence index before triggering an IP command or AHB command. FlexSPI will fetch the command sequence from LUT when IP/AHB command triggered. There are 16 command sequences in LUT. Refer Look Up Table for more details.

NOTE

LUT is implemented as memory, so the reset value is unknown.

27.7.3.32.3 Diagram



27.7.3.32.4 Fields

Field	Function
31-26 OPCODE1	OPCODE1
25-24 NUM_PADS1	NUM_PADS1
23-16 OPERAND1	OPERAND1
15-10 OPCODE0	OPCODE
9-8 NUM_PADS0	NUM_PADS0
7-0	OPERAND0

Field	Function
OPERAND0	

27.8 AHB Memory Map definition

This section describes FlexSPI module AHB memory map in detail.

27.8.1 AHB Memory Map for Serial Flash memory access

AHB read/write access for serial flash memory is mapped to a specific address range. See the Memory Map and register definition for specific address ranges supported.

AHB Bus feature supported for Serial Flash memory reading:
- Cachable and Non-Cachable access
- Prefetch Enable/Disable
- Burst size: 8/16/32/64 bits
- All burst type: SINGLE/INCR/WRAP4/INCR4/WRAP8/INCR8/WRAP16/INCR16

AHB Bus feature for Serial Flash memory writing:
- Bufferable and Non-Bufferable access
- Burst size: 8/16/32/64 bits
- All burst type: SINGLE/INCR/WRAP4/INCR4/WRAP8/INCR8/WRAP16/INCR16

Refer Flash access by AHB Command for more details about AHB access to Serial Flash memory.

27.8.2 AHB Memory Map for IP RX FIFO read access

See chip-specific chapter for the address range mapped for AHB read access.

AHB Bus feature supported for IP RX FIFO reading:
- Burst size: 8/16/32/64 bits
- All burst type: SINGLE/INCR/WRAP4/INCR4/WRAP8/INCR8/WRAP16/INCR16

Refer Reading Data from IP RX FIFO for more details about IP RX FIFO reading.

27.8.3 AHB Memory Map for IP TX FIFO write access

See chip-specific chapter for the address range mapped for AHB write access.

AHB Bus feature supported for IP TX FIFO writing:

- Burst size: 8/16/32/64 bits
- All burst type: SINGLE/INCR/WRAP4/INCR4/WRAP8/INCR8/WRAP16/INCR16

Refer Filling Data to IP TX FIFO for more details about IP TX FIFO filling.

Chapter 28
ARM Cortex M7 Platform

28.1 Chip-specific Arm Cortex M7 information

Table 28-1. Reference links to related information

Topic	Related module(s)	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
Interrupts, DMA Events and XBAR Assignments	-	Interrupts, DMA Events and XBAR Assignments
System Debug	-	System Debug

There are 16 regions implemented in the MPU.

Below is a summary of the CoreSight components used in RT10xx and their corresponding Arm documents:

Table 28-2. CoreSight Components and Corresponding Arm Documents

CoreSight Component	Arm Document	Comment
TSGEN (Timestamp generator)	Arm CoreSight™ SoC-400 Revision: r3p2 Technical Reference Manual	Read-only APB interface of the TSGEN is not applicable.
ETM	Arm CoreSight™ ETM-M7 Revision: r0p1 Technical Reference Manual	
DAP	Arm Cortex-M7 Processor Revision: r1p2	

Table continues on the next page...

Table 28-2. CoreSight Components and Corresponding Arm Documents (continued)

CoreSight Component	Arm Document	Comment
	Integration and Implementation Manual	
ITM	Arm Cortex-M7 Processor Revision: r1p2 Technical Reference Manual	
TPIU		
DWT		
CTI		

28.2 Overview

The Cortex-M7 platform features a single Arm®Cortex®-M7 processor in this chip. The Arm®Cortex®-M7 processor is a highly efficient, high-performance, embedded processor that features low interrupt latency, low-cost debug, and has backwards compatibility with existing Cortex-M profile processors. The processor has an in-order super-scalar pipeline by which many instructions can be dual-issued, including load/load and load/store instruction pairs because of multiple memory interfaces.

Memory interfaces that the processor supports include:

- Tightly-Coupled Memory (TCM)
- Harvard instruction and data caches and AXI master (AXIM) interface
- Dedicated low-latency AHB-Lite peripheral (AHBP) interface

The Arm Cortex-M7 Platform supports the following:

- 32 KB L1 Instruction Cache
- 32 KB L1 Data Cache
- Floating Point Unit (FPU) with support for the FPv5 architecture
- Internal Trace (TRC)

The number of IRQs supported for this chip is 160. In addition, it supports various components composing the Arm CoreSight debug/Trace system, such as ETM and CTI.

NOTE

This chip supports up to 16 interrupt priority levels, i.e. it implements bits [7:4] of each NVIC Interrupt Priority Register.

28.2.1 Block Diagram

A block diagram for the Cortex-M7 is given below:

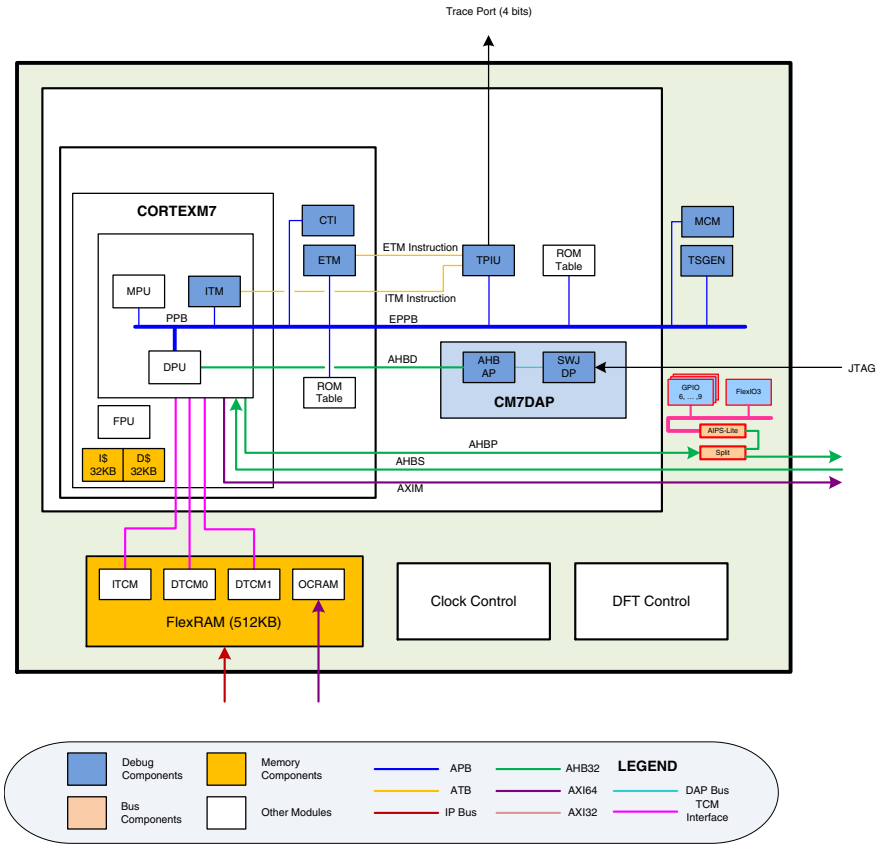


Figure 28-1. Cortex-M7 Platform Block Diagram

28.3 Functional Description

28.3.1 CM7_MCM sub-module

CM7_MCM is a miscellaneous control module for ARM Cortex-M7. It is connected onto the Cortex-M7 PPB bus. See the register descriptions for all its features.

28.3.2 Clocks

Table 28-3. Arm Cortex-M7 Clocks

Clock Name	Description
ipg_clk	Peripheral clock
axi_clk	Bus clock
main_clk	Arm core clock
trace_clk_in	Clock signal

28.3.3 Reset

This CM7_MCM module uses Cortex-M7's reset. Cold or soft resetting the CM7 core would also reset this module.

28.3.4 Interrupts

The CM7_MCM module has one interrupt request: mcm_irq.

The mcm_irq will be asserted when any of the interrupts in the register ISCR occurs.

28.4 External Signals

Table 28-4. External Signals

Signal	Description	Pad	Mode	Direction
ARM_TRACE0	Trace signal	GPIO_B0_04 pin 40 (MAU)	ALT3	O
ARM_TRACE1	Trace signal	GPIO_B0_05 pin 41 (MAU)	ALT3	O
ARM_TRACE2	Trace signal	GPIO_B0_06 pin 42 (MAU)	ALT3	O
ARM_TRACE3	Trace signal	GPIO_B0_07 pin 43 (MAU)	ALT3	O
ARM_TRACE_CLK	Clock Signal	GPIO_B0_12 pin 32	ALT2	O
ARM_EVENTO	Output Event signal	GPIO_B0_14	ALT2	O
ARM_EVENTI	Input Event Signal	GPIO_B0_15	ALT2	I

28.5 Memory Map and register definition

This section includes the CM7_MCM module memory map and detailed descriptions of all registers.

28.5.1 CM7_MCM register descriptions

28.5.1.1 CM7_MCM memory map

CM7_MCM base address: E008_0000h

Offset	Register	Width (In bits)	Access	Reset value
10h	Interrupt Status and Control Register (ISCR)	32	RW	0000_0000h

28.5.1.2 Interrupt Status and Control Register (ISCR)

28.5.1.2.1 Offset

Register	Offset
ISCR	10h

28.5.1.2.2 Function

The ISCR register defines the configuration and reports status for a number of core-related interrupt exception conditions. It includes the enable and status bits associated with the core's floating-point exceptions, and bus errors on TCM. The individual event indicators are first qualified with their exception enables and then logically summed to form an interrupt request sent to the core's NVIC. Bits 15-8 are read-only indicator flags based on the processor's FPSCR register. Attempted writes to these bits are ignored. Once set, the flags remain asserted until software clears the corresponding FPSCR bit.

28.5.1.2.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	FIDCE	0		FIXCE	FUFCE	FOFCE	FDZCE	FIOCE	0		WABE	0				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FIDC	0		FIXC	FUFC	FOFC	FDZC	FIOC	0	WABSO	WABS	0				
W											W1C					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

28.5.1.2.4 Fields

Field	Function
31 FIDCE	FPU Input Denormal Interrupt Enable 0b - Disable interrupt 1b - Enable interrupt
30-29 —	Reserved
28 FIXCE	FPU Inexact Interrupt Enable 0b - Disable interrupt 1b - Enable interrupt
27 FUFCE	FPU Underflow Interrupt Enable 0b - Disable interrupt 1b - Enable interrupt
26 FOFCE	FPU Overflow Interrupt Enable 0b - Disable interrupt 1b - Enable interrupt
25 FDZCE	FPU Divide-by-Zero Interrupt Enable 0b - Disable interrupt 1b - Enable interrupt
24 FIOCE	FPU Invalid Operation Interrupt Enable 0b - Disable interrupt 1b - Enable interrupt
23-22 —	Reserved
21 WABE	TCM Write Abort Interrupt enable 0b - Disable interrupt 1b - Enable interrupt
20-16 —	Reserved
15 FIDC	FPU Input Denormal Interrupt Status This read-only bit is a copy of the core's FPSCR[IDC] bit and signals input denormalized number has been detected in the processor's FPU. Once set, this bit remains set until software clears the FPSCR[IDC] bit. 0b - No interrupt 1b - Interrupt occured
14-13 —	Reserved
12 FIXC	FPU Inexact Interrupt Status This read-only bit is a copy of the core's FPSCR[IXC] bit and signals an inexact number has been detected in the processor's FPU. Once set, this bit remains set until software clears the FPSCR[IXC] bit. 0b - No interrupt 1b - Interrupt occured
11 FUFC	FPU Underflow Interrupt Status This read-only bit is a copy of the core's FPSCR[UFC] bit and signals an underflow has been detected in the processor's FPU. Once set, this bit remains set until software clears the FPSCR[UFC] bit. 0b - No interrupt

Table continues on the next page...

Field	Function
	1b - Interrupt occured
10 FOFC	FPU Overflow interrupt status This read-only bit is a copy of the core's FPSCR[OFC] bit and signals an overflow has been detected in the processor's FPU. Once set, this bit remains set until software clears the FPSCR[OFC] bit. 0b - No interrupt 1b - Interrupt occured
9 FDZC	FPU Divide-by-Zero Interrupt Status This read-only bit is a copy of the core's FPSCR[DZC] bit and signals an divide-by-zero operation has been detected in the processor's FPU. Once set, this bit remains set until software clears FPSCR[DZC] bit. 0b - No interrupt 1b - Interrupt occured
8 FIOC	FPU Invalid Operation interrupt Status This read-only bit is a copy of the core's FPSCR[IOC] bit and signals an illegal operation has been detected in the processor's FPU. Once set, this bit remains set until software clears the FPSCR[IOC] bit. 0b - No interrupt 1b - Interrupt occured
7 —	Reserved
6 WABSO	Write Abort on Slave Overrun This read-only bit indicates if another write abort is detected before system software has retrieved all the information from the original event. When the WABS is cleared, this bit is also cleared. 0b - No write abort overrun 1b - Write abort overrun occurred
5 WABS	Write Abort on Slave This bit indicates when a write abort has occurred on AHBS interface. 0b - No abort 1b - Abort
4-0 —	Reserved

Chapter 29
Network Interconnect Bus System (NIC-301)

29.1 Chip-specific NIC-301 information

Table 29-1. Reference links to related information

Topic	Related module(s)	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
Interrupts, DMA Events and XBAR Assignments	-	Interrupts, DMA Events and XBAR Assignments

29.2 Overview

This section provides an overview of the

- NIC-301 (Network Inter-Connect) AXI arbiter IP

  The NIC-301 (by Arm Ltd.) is a configurable AXI arbiter between several masters and slaves. The NIC-301 IP is designed so that many configuration options are selected at the hardware design stage, determined by SoC characteristics and needs, while several other configuration options are software-controlled.

This chapter covers in brief the NIC-301 while providing configuration details on the NIC-301 instances used in the chip. For complete details on the NIC-301 design, see the Arm specification, *AMBA® Network Interconnect (NIC-301) Technical Reference Manual, version r2p3*.

NOTE

The NIC-301 default settings, in most cases, should not be modified by the customer. The default settings have gone through exhaustive testing during the validation of the part, and have proven to work well for the part's intended target applications. Changes to the default settings may result in a degradation in system performance.

29.2.1 NIC-301 Main Features

Key features of the NIC-301 module include the following:

- Address space memory mapping.
- Programmer's view, for software-configured parameters, via internal "GPV" ports.
- Support for cross-clock domain synchronization.

29.2.2 Modes and Operations

The NIC-301 supports a normal functional mode only.

29.3 External Signals

There are no external I/O interfaces for NIC-301.

29.4 Memory Map and Register Definition

The bus system is composed of five instances: SIM_M7, SIM_PER, SIM_M, SIM_MAIN and SIM_EMS. Three of them have GPV registers which are helpful for bus arbitration and performance.For detailed descriptions of these registers, see the Arm document: *DDI0397I_corelink_network_interconnect_nic301_r2p3_trm.pdf*.

1. **SIM_MAIN registers**

   The SIM_MAIN GPV base address is GPV0_BASE = 0x41000000. The following registers are implemented in this NIC.

Register Name	Port Name	Module Name	Absolute Address	Reset Value	Descriptions
read_qos	m_a_2	LCD	GPV0_BASE + 0x44000 + 0x100	1	Set the priority of master's read. The priority level would be used when the master's read transaction is being arbitrated by the NIC. Legal programmable values are from 0 to 15. Higher number sets higher priority (0: the lowest arbitration priority; ...; 15: the highest priority).
read_qos	m_a_3	CSI	GPV0_BASE + 0x45000 + 0x100	4	Same as above
read_qos	m_a_4	PXP	GPV0_BASE + 0x46000 + 0x100	2	Same as above
write_qos	m_a_2	LCD	GPV0_BASE + 0x44000 + 0x104	1	Set the priority of master's write. The priority level would be used when the master's write transaction is being arbitrated by the NIC. Legal programmable values are from 0 to 15. Higher number sets higher priority (0: the lowest arbitration priority; ...; 15: the highest priority).
write_qos	m_a_3	CSI	GPV0_BASE + 0x45000 + 0x104	4	Same as above
write_qos	m_a_4	PXP	GPV0_BASE + 0x46000 + 0x104	2	Same as above
fn_mod	m_a_2	LCD	GPV0_BASE + 0x44000 + 0x108	0	Issuing functionality modification register. This register sets the block issuing capability to one outstanding transaction. Legal programmable values are 0, 1, 2 and 3.

Table continues on the next page...

Register Name	Port Name	Module Name	Absolute Address	Reset Value	Descriptions
					NOTE: It is recommended to keep its reset value to obtain the best performance. • 0: Default setting • 1: Set read issuing capability to one outstanding transaction • 2: Set write issuing capability to one outstanding transaction • 3: Set both read and write issuing capability to one outstanding transaction
fn_mod	m_a_3	CSI	GPV0_BASE + 0x45000 + 0x108	0	Same as above
fn_mod	m_a_4	PXP	GPV0_BASE + 0x46000 + 0x108	0	Same as above

2. SIM_M registers

The SIM_M GPV base address is GPV1_BASE = 0x41100000. The following registers are implemented in this NIC.

Register Name	Port Name	Module Name	Absolute Address	Reset Value	Descriptions
fn_mod2	m_c_0	DCP	GPV1_BASE + 0x42000 + 0x024	0	Legal programmable values are 0 and 1. • 0: Enable the size merge function • 1: Bypass the size merge function

Table continues on the next page...

Register Name	Port Name	Module Name	Absolute Address	Reset Value	Descriptions
					NOTE: It is recommended to keep its reset value to obtain the best performance.
fn_mod_ahb	m_c_1	ENET	GPV1_BASE + 0x43000 + 0x028	0	This register has 3 control bits. • Bit 0: rd_incr_override. Writting 1 to this bit forces NIC to convert all AHB read transactions to a series of AXI singles. • Bit 1: wr_incr_override. Writting 1 to this bit forces NIC to convert all AHB write transactions to a series of AXI singles. • Bit 2: lock_override. Writting 1 to this bit forces NIC not to create any AXI lock transactions. **NOTE:** It is recommended to keep its reset value to obtain the best performance.
fn_mod_ahb	m_c_5	TestPort	GPV1_BASE + 0x47000 + 0x028	0	Same as above
fn_mod_ahb	m_c_6	ENET2	GPV1_BASE + 0x48000 + 0x028	0	Same as above
read_qos	m_c_0	DCP	GPV1_BASE + 0x42000 + 0x100	0	Set the priority of master's read. The priority level would be used when the master's read transaction is being arbitrated by the NIC.

Table continues on the next page...

Register Name	Port Name	Module Name	Absolute Address	Reset Value	Descriptions
					Legal programmable values are from 0 to 15. Higher number sets higher priority (0: the lowest arbitration priority; ...; 15: the highest priority).
read_qos	m_c_1	ENET	GPV1_BASE + 0x43000 + 0x100	3	Same as above
read_qos	m_c_2	USBO2	GPV1_BASE + 0x44000 + 0x100	1	Same as above
read_qos	m_c_3	USDHC1	GPV1_BASE + 0x45000 + 0x100	2	Same as above
read_qos	m_c_4	USDHC2	GPV1_BASE + 0x46000 + 0x100	2	Same as above
read_qos	m_c_5	TestPort	GPV1_BASE + 0x47000 + 0x100	0	Read-only register.
read_qos	m_c_6	ENET2	GPV1_BASE + 0x48000 + 0x100	3	Set the priority of master's read. The priority level would be used when the master's read transaction is being arbitrated by the NIC. Legal programmable values are from 0 to 15. Higher number sets higher priority (0: the lowest arbitration priority; ...; 15: the highest priority).
write_qos	m_c_0	DCP	GPV1_BASE + 0x42000 + 0x104	0	Set the priority of master's write. The priority level would be used when the master's write transaction is being arbitrated by the NIC. Legal programmable values are from 0 to 15. Higher number sets higher priority (0: the lowest arbitration; ...; 15: the highest priority).
write_qos	m_c_1	ENET	GPV1_BASE + 0x43000 + 0x104	3	Same as above

Table continues on the next page...

Register Name	Port Name	Module Name	Absolute Address	Reset Value	Descriptions
write_qos	m_c_2	USBO2	GPV1_BASE + 0x44000 + 0x104	1	Same as above
write_qos	m_c_3	USDHC1	GPV1_BASE + 0x45000 + 0x104	2	Same as above
write_qos	m_c_4	USDHC2	GPV1_BASE + 0x46000 + 0x104	2	Same as above
write_qos	m_c_5	TestPort	GPV1_BASE + 0x47000 + 0x104	0	Read-only register.
write_qos	m_c_6	ENET2	GPV1_BASE + 0x48000 + 0x104	3	Set the priority of master's write. The priority level would be used when the master's write transaction is being arbitrated by the NIC. Legal programmable values are from 0 to 15. Higher number sets higher priority (0: the lowest arbitration; ...; 15: the highest priority).
fn_mod	m_c_0	DCP	GPV1_BASE + 0x42000 + 0x108	0	Issuing functionality modification register. This register sets the block issuing capability to one outstanding transaction. Legal programmable values are 0, 1, 2 and 3. **NOTE:** It is recommended to keep its reset value to obtain the best performance. • 0: Default setting • 1: Set read issuing capability to one outstanding transaction • 2: Set write issuing capability to

Table continues on the next page...

Register Name	Port Name	Module Name	Absolute Address	Reset Value	Descriptions
					one outstanding transaction • 3: Set both read and write issuing capability to one outstanding transaction
fn_mod	m_c_1	ENET	GPV1_BASE + 0x43000 + 0x108	0	Same as above
fn_mod	m_c_2	USBO2	GPV1_BASE + 0x44000 + 0x108	0	Same as above
fn_mod	m_c_3	USDHC1	GPV1_BASE + 0x45000 + 0x108	0	Same as above
fn_mod	m_c_4	USDHC2	GPV1_BASE + 0x46000 + 0x108	0	Same as above
fn_mod	m_c_5	TestPort	GPV1_BASE + 0x47000 + 0x108	0	Same as above
fn_mod	m_c_6	ENET2	GPV1_BASE + 0x48000 + 0x108	0	Same as above

3. SIM_M7 registers

The SIM_M7 GPV base address is GPV4_BASE = 0x41400000. The following registers are implemented in this NIC.

Register Name	Port Name	Module Name	Absolute Address	Reset Value	Descriptions
fn_mod_ahb	m_b_1	DMA	GPV4_BASE + 0x43000 + 0x028	0	This register has 3 control bits. • Bit 0: rd_incr_override. Writting 1 to this bit forces NIC to convert all AHB read transactions to a series of AXI singles. • Bit 1: wr_incr_override. Writting 1 to this bit forces NIC to convert all AHB write

Table continues on the next page...

Register Name	Port Name	Module Name	Absolute Address	Reset Value	Descriptions
					transactions to a series of AXI singles. • Bit 2: lock_override. Writing 1 to this bit forces NIC not to create any AXI lock transactions. **NOTE:** It is recommended to keep its reset value to obtain the best performance.
wr_tidemark	m_b_0	Cortex-M7	GPV4_BASE + 0x42000 + 0x040	4	The write data FIFO depth is 4, so the valid programmable values to this register are 0, 1, 2, 3 and 4. This is a tidemark level that stalls the release of the transaction until: • The NIC receives the WLAST beat. • The write FIFO becomes full. • The number of occupied slots in the write data FIFO exceeds the write tidemark.
read_qos	m_b_0	Cortex-M7	GPV4_BASE + 0x42000 + 0x100	4	Set the priority of master's read. The priority level would be used when the master's read transaction is being arbitrated by the NIC. Legal programmable values are from 0 to 15. Higher number sets higher priority (0: the lowest

Table continues on the next page...

Register Name	Port Name	Module Name	Absolute Address	Reset Value	Descriptions
					arbitration priority; ...; 15: the highest priority).
read_qos	m_b_1	DMA	GPV4_BASE + 0x43000 + 0x100	3	Same as above
write_qos	m_b_0	Cortex-M7	GPV4_BASE + 0x42000 + 0x104	4	Set the priority of master's write. The priority level would be used when the master's write transaction is being arbitrated by the NIC. Legal programmable values are from 0 to 15. Higher number sets higher priority (0: the lowest arbitration priority; ...; 15: the highest priority).
write_qos	m_b_1	DMA	GPV4_BASE + 0x43000 + 0x104	3	Same as above
fn_mod	m_b_0	Cortex-M7	GPV4_BASE + 0x42000 + 0x108	0	Issuing functionality modification register. This register sets the block issuing capability to one outstanding transaction. Legal programmable values are 0, 1, 2 and 3. **NOTE:** It is recommended to keep its reset value to obtain the best performance. • 0: Default setting • 1: Set read issuing capability to one outstanding transaction • 2: Set write issuing capability to

Table continues on the next page...

Register Name	Port Name	Module Name	Absolute Address	Reset Value	Descriptions
					one outstanding transaction • 3: Set both read and write issuing capability to one outstanding transaction
fn_mod	m_b_1	DMA	GPV4_BASE + 0x43000 + 0x108	0	Same as above

4. **IB registers**

   There are no IB registers implemented in this chip.

5. **Address region control registers**

   There are not such type of registers implemented in this chip.

6. **Peripheral ID registers**

   The peripheral ID registers are implemented in SIM_MAIN, SIM_M, and SIM_M7. For more details, please see the Arm document: *DDI0397I_corelink_network_interconnect_nic301_r2p3_trm.pdf*.

Chapter 30
On-Chip RAM Memory Controller (OCRAM)

30.1 Chip-specific OCRAM information

Table 30-1. Reference links to related information

Topic	Related module(s)	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
Interrupts, DMA Events and XBAR Assignments	-	Interrupts, DMA Events and XBAR Assignments

30.2 Overview

NOTE

For the M7 OCRAM (within FlexRAM module), see the FlexRAM chapter for introduction and details.

There is 1 OCRAM controller implemented in the chip. One controller is for up to 512 KB on-chip RAM.

The OCRAM block is implemented as an slave module on the 64-bit system AXI bus. Designed as a simple on-chip memory controller, it supports only one AXI port with memory banks. For the AXI port, the read and write transactions are handled by two independent modules. As it is possible to have simultaneous read and write request from

the AXI bus, each memory bank has an arbiter with round-robin scheme. After arbitration, the granted read or write access command can then be issued to the memory cell through a read/write MUX.

The memory banks are organized with the lower 2 bits of the address which is the AXI bus address and is 64 bits aligned interleaved. This allows a read access and a write access to be processed at the same time if they are targeted to different memory banks.

Various options are provided for adding a pipeline or wait-states in a read/write access, in order to ensure flexible timing control at both high and low frequencies.

30.2.1 Block diagram

The OCRAM block diagram is shown in the figure below.

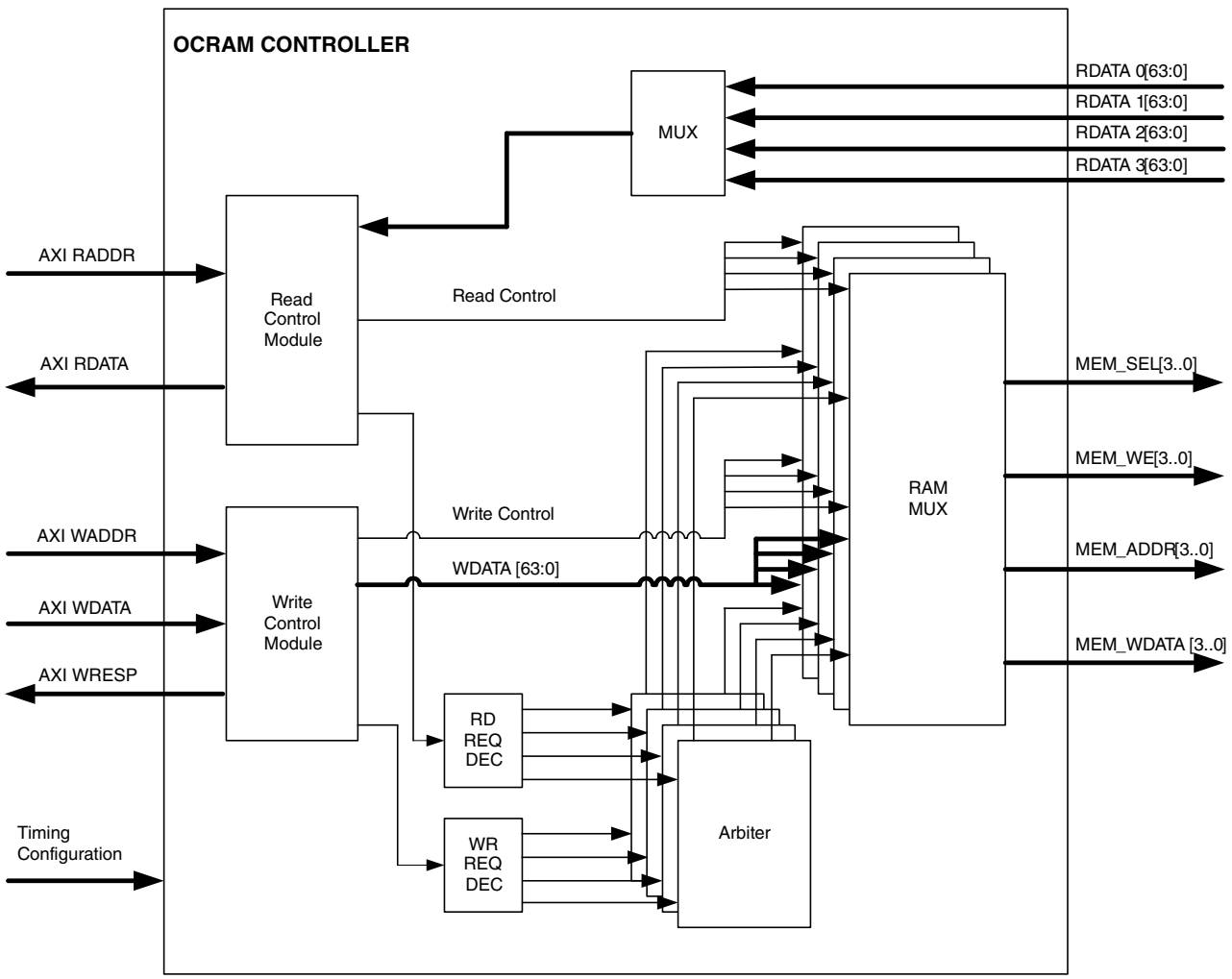


Figure 30-1. OCRAM Block Diagram

30.3 Functional Description

30.3.1 Read/Write Arbitration

The detailed rules used in arbitration are as follows:

- If there is no granted read or write in the last cycle, and there is only a read request or a write request, the request will be granted.
- If there is no granted read or write in the last cycle, and there are both read or write requests coming in at the same time, the read request will be granted first.
- If a granted read/write transaction has just finished, the write/read request will have the higher priority in the next cycle.
- If the first read/write access request in a transaction is granted, all the data transfer in this burst will be finished before the next arbitration begins, that is, the round-robin arbitration mechanism is based on AXI transaction, not data access.

30.3.2 Advanced Features

This section describes some advanced features designed to avoid timing issues when the OCRAM is working at high frequency.

All of the features can be disabled/enabled by programming the corresponding fields of the General Purpose Register (IOMUXC.GPR3) bits [3:0] in the IOMUX chapter.

30.3.2.1 Read Data Wait State

When the wait state is enabled, it will take 2 cycles for each read access (each beat of a read burst).

This can avoid the potential timing problem caused by the longer memory access time at higher frequency.

When this feature is disabled, it only takes 1 clock cycle to finish a read transaction. That is, read data is available in the next cycle of read request becomes valid on the bus.

For the OCRAM, the read data wait state is configurable via IOMUXC.GPR3[0].

30.3.2.2 Read Address Pipeline

When this feature is enabled, the read address from the AXI master is delayed 1 cycle before it can be accepted by the OCRAM.

This can avoid setup time issues for the read access on the memory cell at high frequency. Enabling this feature can cost, at most, 1 more clock cycle for each AXI read transaction, that is, at most 1 more clock cycle for each read burst with multiple beats of data.

When this feature is disabled, the read address from the AXI master can be accepted by the OCRAM without delay, and data can become ready for master at next clock cycle (if no other access and no read data wait).

For the OCRAM, the read address pipeline is configurable via IOMUXC.GPR3[1].

30.3.2.3 Write Data Pipeline

When this feature is enabled, the write data from the AXI master would be delayed 1 cycle before it can be accepted by the OCRAM.

This can avoid setup time issue for the write access on the memory cell at high frequency. Enabling this feature would cost at most 1 more clock cycle for each AXI write transaction, that is, at most 1 more clock cycle for each write burst with multiple beats of data.

When this feature is disabled, the write data from the AXI master can be accepted by the OCRAM without delay, and data can be written to memory at this cycle (if no other access and write address is also ready at this cycle).

For the OCRAM, the write data pipeline is configurable via IOMUXC.GPR3[2].

30.3.2.4 Write Address Pipeline

When this feature is enabled, the write address from the AXI master would be delayed 1 cycle before it can be accepted by the OCRAM.

This can avoid setup time issue for the write access on the memory cell at high frequency. Enabling this feature would take at most 1 more clock cycle for each AXI write transaction, that is, at most 1 more clock cycle for each write burst with multiple beats of data.

When this feature is disabled, the write address from the AXI master can be accepted by the OCRAM without delay, and data can be written to memory at this cycle (if no other access and write data is also ready at this cycle).

For the OCRAM, the write address pipeline is configurable via IOMUXC.GPR3[3].

30.4 Programmable Registers

There are no programmable registers in this block. However, OCRAM configurable bits can be found in the IOMUX Controller (IOMUXC) general purpose registers as below.

- TrustZone bits: IOMUXC_GPR_GPR10

Chapter 31
FlexRAM

31.1 Chip-specific FlexRAM information

Table 31-1. Reference links to related information

Topic	Related module(s)	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
Interrupts, DMA Events and XBAR Assignments	-	Interrupts, DMA Events and XBAR Assignments

For additional information see application note AN12077: Using the i.MX RT FlexRAM. This document describes the flexible memory array available on the i.MX RT MCUs. It includes the following information:

- Configuration of the bank array
- Memory type size definition
- Available memory controllers
- Power domains and clocks
- Interrupt request generation
- Example of FlexRAM configuration usage on a specific application use case

31.2 Overview

31.2.1 Introduction

This SoC has 512 KB of on-chip RAM which is shared by I-TCM, D-TCM and general purpose On Chip RAM (OCRAM). The FlexRAM is the manager of the big on-chip RAM array.

31.2.2 Features

The FLEXRAM includes the following features:

- Integrated I-TCM and D-TCM RAM controller
  - 64-bit I-TCM interface and 2x 32-bit D-TCM interface.
  - The controller supports up to 512 KB TCM (for both ITCM and DTCM) space.
  - The controller supports two access modes:
    - Fast mode: RAM accesses is expected to be finished in 1-cycle for both read and write.
    - Wait mode: RAM accesses is expected to be finished in 2-cycle. Wait mode for read and write path can be enabled separately.
  - Synchronous interface to the M7 Core,run at the same frequency as the core
  - Automatically clock gating control to reduce power consumption
- Integrated OCRAM controller
  - single SRAM bank controller.
  - support up to 512 KB SRAM size
  - Synchronous to the system bus, runs at the same frequency as bus fabric
- Parameterized RAM Array
  - Support up to 512 KB total memory space
  - Support up to 32 block of 32-bit RAM
- RAM Array portioning
  - RAM size of ITCM, DTCM and OCRAM can be configured from 0 to full RAM Array size separately
  - Step of the RAM size partitioning for ITCM, DTCM and OCRAM is (Total RAM SIZE)/16
  - Flexible RAM bank organization.
- Flexible power mode:
  - Run mode: All RAM banks are powered on

- Retention mode: Only 1 (Bank0) out of the 16 RAM banks is on while the rest banks are powered off
- Partial mode: 8 out of the 16 RAM banks (bank8 to bank15) can be powered off dynamically

- Bank8~15 can be disabled with hardware fuse setting. When the fuse is set, total RAM size will be limit to 256 KB
- Generates interrupt upon TCM, OCRAM access out of configured RAM range

31.2.3 Block diagram

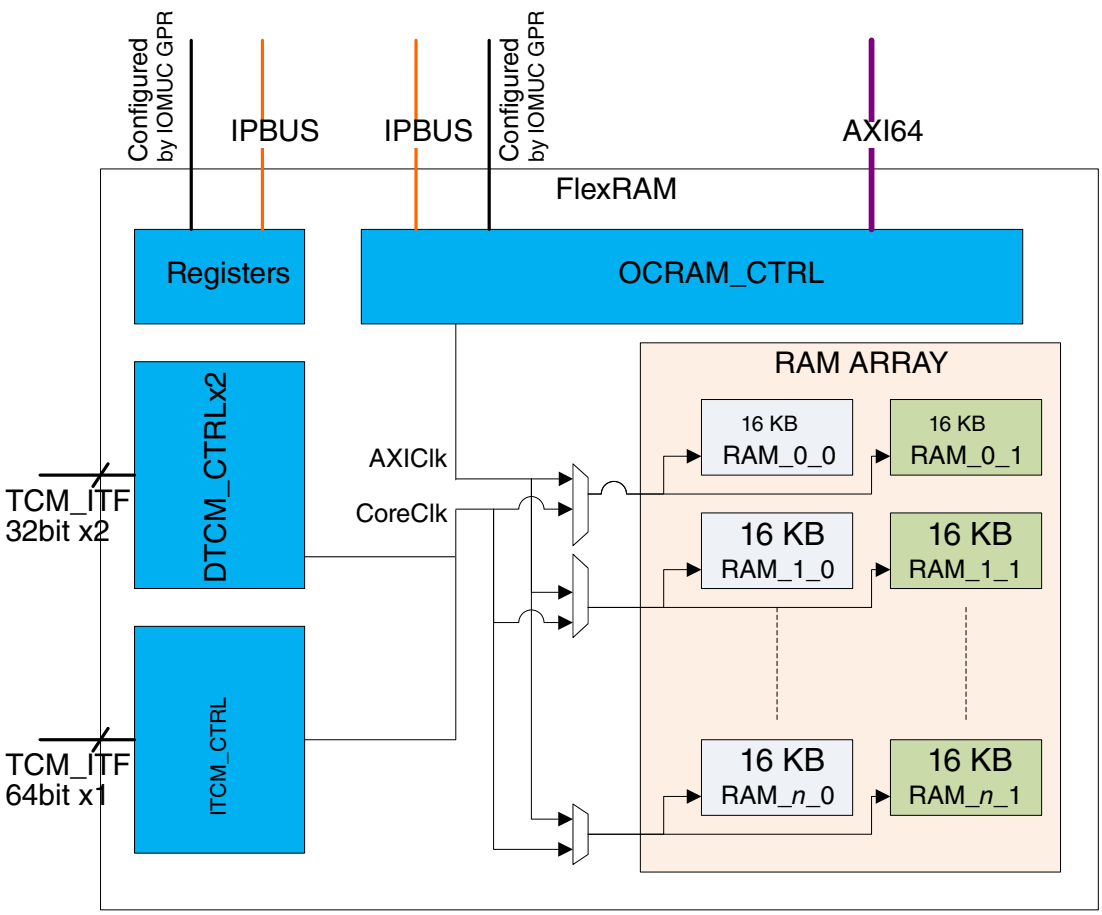


Figure 31-1. FLEXRAM block diagram

31.3 Functional description

FLEXRAM converts AXI and TCM interface signals to RAM interface signals and implements mux control for OCRAM , DTCM and ITCM access to on-chip RAM.

31.3.1 Interface Conversion

FLEXRAM integrate OCRAM controller which converts AXI master interface signals to RAM interface signals(Please see OCRAM section). OCRAM controller can support to add pipeline or wait-state in read/write access by IOMUX GPR.

FLEXRAM also implement TCM controller module to convert DTCM/ITCM interface to RAM interface. TCM read/write access can be extended to 2 cycles by setting TCM_CTRL register. When DTCM and ITCM access unallocated RAM address, TCM controller can send tcm_err to DTCM/ITCM interface.

31.3.2 RAM Bank Allocation

OCRAM, DTCM and ITCM share 512 KB of on-chip RAM. SW can configure from 0 to 512 KB full RAM array size for OCRAM, DTCM and ITCM by IOMUX GPR16 and GPR17. The RAM size step is 32KB. The allocated RAM bank can be not overlapped for OCRAM, DTCM and ITCM. Table below show an example of a 128KB ITCM, 128KB DTCM and 256KB OCRAM configuration.

Table 31-2. FLEXRAM Partitioning Example

	ITCM	DTCM	OCRAM
Bank0	0	0	1
Bank1	0	0	1
Bank2	0	0	1
Bank3	0	0	1
Bank4	1	0	0
Bank5	1	0	0
Bank6	0	1	0
Bank7	0	1	0
Bank8	0	0	1
Bank9	0	0	1
Bank10	0	0	1
Bank11	0	0	1
Bank12	1	0	0
Bank13	1	0	0
Bank14	0	1	0
Bank15	0	1	0

Each RAM bank has 2 bits bank configuration from IOMUX GPR17.

Table 31-3. RAM Bank configuration

	RAM Bank Configuration
00	Not used
01	OCRAM
10	DTCM
11	ITCM

31.3.3 Low power modes

FLEXRAM has input fuse control signal to control ocram bank8~bank15 interface access. When the fuse bit is programmed, bank8~bank15 can't be accessed again and can be power down.

31.3.4 Clocks

The following table describes the clock sources of the FlexRAM module.

Table 31-4. FlexRAM Clocks

Clock name	Clock Root	Description
core_clk	arm_clk_root	ARM core clock
axi_clk	axi_clk_root	AXI bus clock
ipg_clk	ipg_clk_root	Peripheral clock
ipg_clk_s	ipg_clk_root	Peripheral access clock for register accesses

31.3.5 Reset

FLAMRAM has 2 reset siganls. One is used for OCRAM interface and in always on power domain. The other is used for TCM interface and in core power domain.

31.3.6 Interrupts

FLEXRAM can generate interrput in the following cases:

- OCRAM, DTCM or ITCM access the RAM address which is not allocated.

31.4 Memory Map and register definition

This section includes the FLEXRAM module memory map and detailed descriptions of all registers.

31.4.1 FLEXRAM register descriptions

31.4.1.1 FLEXRAM memory map

FlexRAM base address: 400B_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	TCM CRTL Register (TCM_CTRL)	32	RW	0000_0000h
10h	Interrupt Status Register (INT_STATUS)	32	W1C	0000_0000h
14h	Interrupt Status Enable Register (INT_STAT_EN)	32	RW	0000_0000h
18h	Interrupt Enable Register (INT_SIG_EN)	32	RW	0000_0000h

31.4.1.2 TCM CRTL Register (TCM_CTRL)

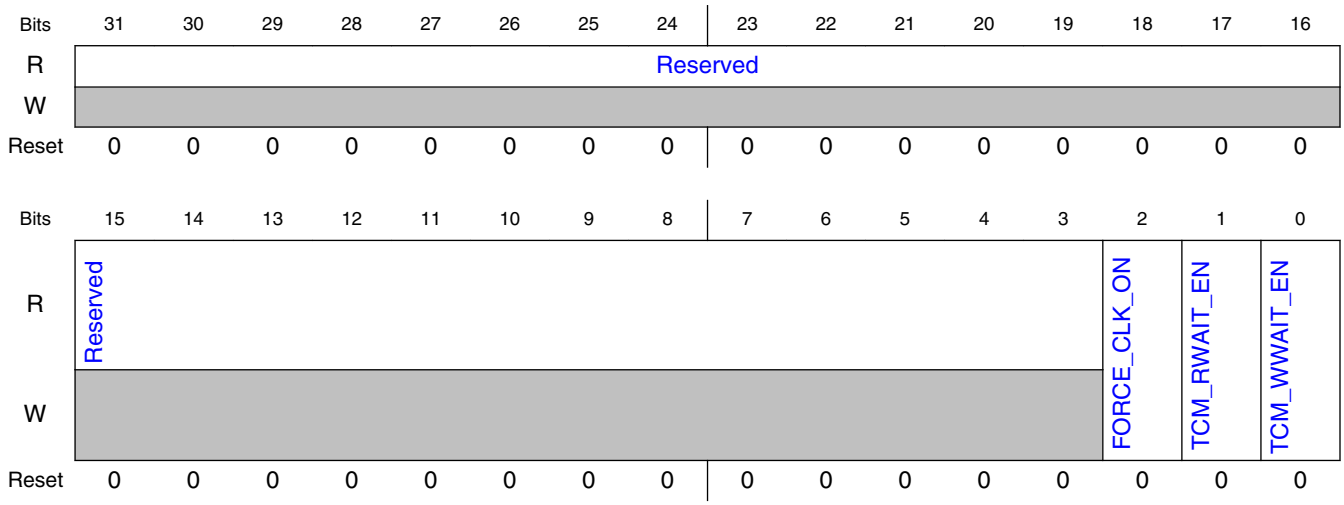
31.4.1.2.1 Offset

Register	Offset
TCM_CTRL	0h

31.4.1.2.2 Function

.

31.4.1.2.3 Diagram



31.4.1.2.4 Fields

Field	Function
31-3 Reserved	Reserved
2 FORCE_CLK_ON	Force RAM Clock Always On
1 TCM_RWAIT_EN	TCM Read Wait Mode Enable 0b - TCM read fast mode: Read RAM accesses are expected to be finished in 1-cycle. 1b - TCM read wait mode: Read RAM accesses are expected to be finished in 2-cycles.
0 TCM_WWAIT_EN	TCM Write Wait Mode Enable 0b - TCM write fast mode: Write RAM accesses are expected to be finished in 1-cycle. 1b - TCM write wait mode: Write RAM accesses are expected to be finished in 2-cycles.

31.4.1.3 Interrupt Status Register (INT_STATUS)

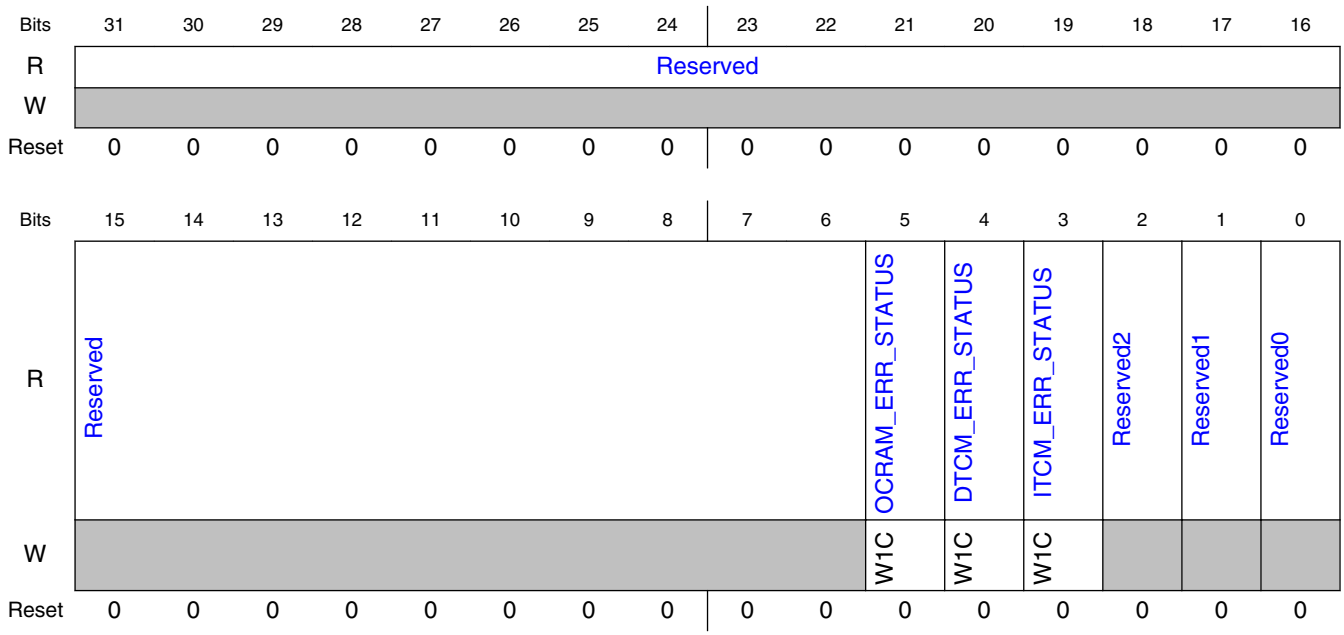
31.4.1.3.1 Offset

Register	Offset
INT_STATUS	10h

31.4.1.3.2 Function

.

31.4.1.3.3 Diagram



31.4.1.3.4 Fields

Field	Function
31-6 Reserved	Reserved
5 OCRAM_ERR_STATUS	OCRAM Access Error Status When OCRAM access unallocated address, this bit will be asserted if corresponding status enable bit in INT_STAT_EN register is set. 0b - OCRAM access error does not happen 1b - OCRAM access error happens.
4 DTCM_ERR_STATUS	DTCM Access Error Status When DTCM access unallocated address, this bit will be asserted if corresponding status enable bit in INT_STAT_EN register is set. 0b - DTCM access error does not happen 1b - DTCM access error happens.
3 ITCM_ERR_STATUS	ITCM Access Error Status When ITCM access unallocated address, this bit will be asserted if corresponding status enable bit in INT_STAT_EN register is set. 0b - ITCM access error does not happen 1b - ITCM access error happens.
2	Reserved

Table continues on the next page...

Field	Function
Reserved2	
1 Reserved1	Reserved
0 Reserved0	Reserved

31.4.1.4 Interrupt Status Enable Register (INT_STAT_EN)

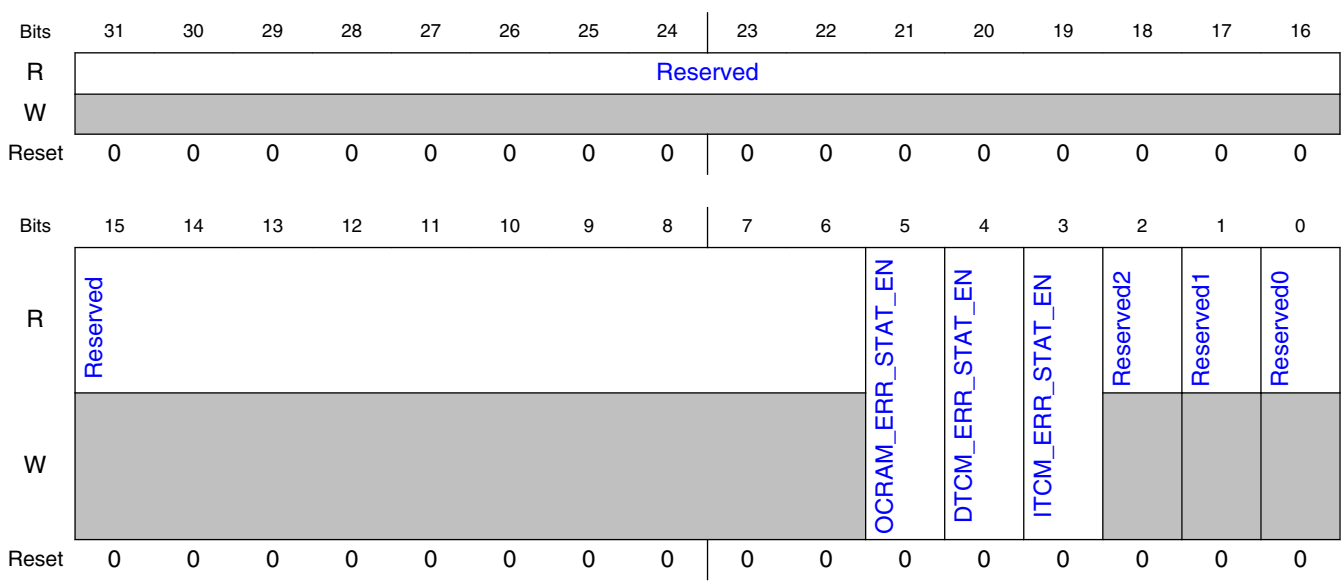
31.4.1.4.1 Offset

Register	Offset
INT_STAT_EN	14h

31.4.1.4.2 Function

Setting the bits in this register to 1 enables the corresponding Interrupt Status to be set by the specified event.

31.4.1.4.3 Diagram



31.4.1.4.4 Fields

Field	Function
31-6 Reserved	Reserved
5 OCRAM_ERR_STAT_EN	OCRAM Access Error Status Enable 0b - Masked 1b - Enabled
4 DTCM_ERR_STAT_EN	DTCM Access Error Status Enable 0b - Masked 1b - Enabled
3 ITCM_ERR_STAT_EN	ITCM Access Error Status Enable 0b - Masked 1b - Enabled
2 Reserved2	Reserved
1 Reserved1	Reserved
0 Reserved0	Reserved

31.4.1.5 Interrupt Enable Register (INT_SIG_EN)

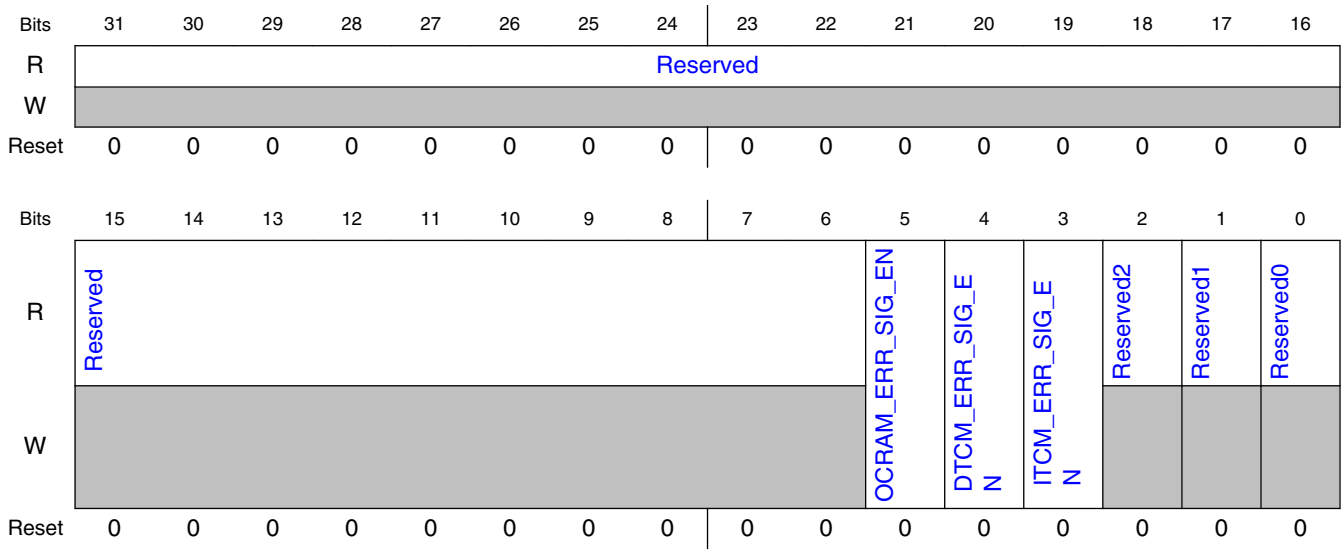
31.4.1.5.1 Offset

Register	Offset
INT_SIG_EN	18h

31.4.1.5.2 Function

This register is used to select which interrupt status is indicated to the Host System as the interrupt. These status bits all share the same interrupt line. Setting any of these bits to 1 enables interrupt generation. The corresponding Status register bit will generate an interrupt when the corresponding interrupt signal enable bit is set.

31.4.1.5.3 Diagram



31.4.1.5.4 Fields

Field	Function
31-6 Reserved	Reserved
5 OCRAM_ERR_SIG_EN	OCRAM Access Error Interrupt Enable 0b - Masked 1b - Enabled
4 DTCM_ERR_SIG_EN	DTCM Access Error Interrupt Enable 0b - Masked 1b - Enabled
3 ITCM_ERR_SIG_EN	ITCM Access Error Interrupt Enable 0b - Masked 1b - Enabled
2 Reserved2	Reserved
1 Reserved1	Reserved
0 Reserved0	Reserved

Chapter 32
AHB to IP Bridge (AIPSTZ)

32.1 Chip-specific AIPSTZ information

Table 32-1. Reference links to related information

Topic	Related module(s)	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Power management	PMU	Power Management Power Management Unit

32.2 Overview

This section provides an overview of the AHB to IP Bridge (AIPSTZ). This particular peripheral is designed as the bridge between AHB bus and peripherals with the lower bandwidth IP Slave (IPS) buses.

32.2.1 Features

The following list summarizes the key features of the bridge:

- The bridge supports the IPS slave bus signals. This interface is only meant for slave peripherals.
- The bridge supports 8-, 16-, and 32-bit IPS peripherals. (Accesses larger than the size of a peripheral are not supported, except to 32-bit memory.)
- The bridge supports a pair of IPS accesses for 64-bit and certain misaligned AHB transfers to 32-bit memory in 64-bit platforms.
- The bridge directly supports up to 32 16-Kbyte external IPS peripherals, and 2 global external IPS peripheral spaces. The bridge occupies 1 MBytes of total address space.

- The bridge provides configurable per-block and per-master access protections. More details on the protection features and configuration can be found in the Security Reference Manual
- Peripheral read transactions require a minimum of 2 hclk clocks, and unbuffered write transactions require a minimum of 3 hclk clocks.
- The bridge uses one single asynchronous reset and one global clock.

32.3 Clocks

The following table describes the clock sources for AIPSTZ. Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 32-2. AIPSTZ Clocks

Clock name	Clock Root	Description
hclk	ahb_clk_root	Module clock

32.4 Functional Description

The AIPS bridge serves as a protocol translator between the AHB system bus and the IP bus.

Support is provided for generating a pair of 32-bit IP bus accesses when targeted by a 64-bit system bus access, or a misaligned access which crosses a 32-bit boundary. No other bus-sizing access support is provided.

The AHB to IP bridge is the interface between the AHB and on-chip IPS peripherals, which are sub-blocks containing readable/writable control and status registers.

The AHB master reads and writes these registers through the AIPSTZ. The bridge generates block enables, the block address, transfer attributes, byte enables and write data as inputs to the IPS peripherals. The bridge captures read data from the IPS interface and drives it on the AHB.

Each bridge that connects to the IPS (or peripherals) are referred as AIPS. The chip has several separate AIPS modules, and peripherals are grouped and assigned under each AIPS block. The list of peripherals are indicated as n-1, ... and n-*x* for AIPS-1, ... and AIPS-*x* respectively.

AIPS occupies a 1-Mbyte portion of the address space. The register maps of the IPS peripherals are located on 16-Kbyte boundaries. Each IPS peripheral is allocated one 16-Kbyte block of the memory map, and is activated by one of the block enables from the bridge. Up to thirty-two 16-Kbyte external IPS peripherals may be implemented, occupying contiguous blocks of 16-Kbytes. Two global external IPS block enables are available for the remaining address space to allow for customization and expansion of addressed peripheral devices. In addition, a single "non-global" block enable is also asserted whenever any of the thirty-two non-global block enables is asserted.

The bridge is responsible for indicating to IPS peripherals if an access is in supervisor or user mode. It may block user mode accesses to certain IPS peripherals or it may allow the individual IPS peripherals to determine if user mode accesses are allowed. In addition, peripherals may be designated as write-protected.

The bridge supports the notion of "trusted" masters for security purposes. Masters may be individually designated as trusted for reads, trusted for writes, or trusted for both reads and writes, as well as being forced to look as though all accesses from a master are in user-mode privilege level. Refer to AIPSTZ Memory Map/Register Definition for more information.

All peripheral devices are expected to only require aligned accesses equal to or smaller in size than the peripheral size. An exception to this rule is supported for 32-bit peripherals to allow memory to be placed on the IPS.

32.5 Access Protections

The AIPSTZ bridge provides programmable access protections for both masters and peripherals. It allows the privilege level of a master to be overridden, forcing it to user-mode privilege, and allows masters to be designated as trusted or untrusted.

Peripherals may require supervisor privilege level for access, may restrict access to a trusted master only, and may be write-protected. IP bus peripherals are subject to access control policies set in both CSU registers and AIPSTZ registers. An access is blocked if it is denied by either policy.

32.6 Access Support

Aligned 64-bit accesses, aligned and misaligned word and half word accesses, as well as byte accesses are supported for 32-bit peripherals. Misaligned accesses are supported to allow memory to be placed on the IPS.

Peripheral registers must not be misaligned, although no explicit checking is performed by the AIPS bridge. The bridge will perform two IPS transfers for 64-bit accesses, word accesses with byte offsets of 1, 2, or 3, and for half word accesses with a byte offset of 3. All other accesses will be performed with a single IPS transfer.

Only aligned half word and byte accesses are supported for 16-bit peripherals. All other accesses types are unsupported, and results of such accesses are undefined. They are not terminated with an error response.

Only byte accesses are supported for 8-bit peripherals. All other accesses types are unsupported, and results of such accesses are undefined. They are not terminated with an error response.

32.7 Initialization Information

The AIPS bridge should be programmed before use.

The following registers should be initialized: The Master Privilege Registers (AIPSTZ_MPRs), the Peripheral Access Control registers (AIPSTZ_PACRs), and the Off-platform Peripheral Access Control registers (AIPSTZ_OPACRs) described in AIPSTZ Memory Map/Register Definition.

32.7.1 Security Block

The AIPSTZ contains a security block that is connected to each off-platform peripheral. This block filters accesses based on write/read, non-secure, and supervisor signals.

Each peripheral can be individually configured to allow or deny each of the following transactions as described in the table below:

Table 32-3. Peripheral Access Configuration options

Config Bit	Write	Non-Secure	Supervisor	Meaning
0	0	0	0	Secure User Read
1	0	0	1	Secure Supervisor Read
2	0	1	0	Non-Secure User Read
3	0	1	1	Non-Secure Supervisor Read
4	1	0	0	Secure User Write
5	1	0	1	Secure Supervisor Write
6	1	1	0	Non-Secure User Write
7	1	1	1	Non-Secure Supervisor Write

Each peripheral has a security configuration (sec_config_X) input for determining whether to allow or deny a given access type. These are 8-bit vectors, with each bit corresponding to one of the transactions above as listed in the Config Bit column of Table 32-3. If the bit is asserted (1'b1), the transaction is allowed. If the bit is negated (1'b0), the transaction is not allowed.

For example, if peripheral 0 is configured as follows:

sec_config_0 [7:0] = 8'b0011_0011

This peripheral can only be accessed by secure transactions. Bits 0, 1, 4, and 5 are asserted and these bits refer to the four types of secure transactions. If an insecure transaction is attempted to this peripheral, it will result in an error.

Eight bits per peripheral across an entire system can result in a large number of configuration bits that must be assigned and controlled, most likely in a series of registers in another block. To reduce the number of register bits required predefined sets of security profiles can be defined and encapsulated in an external security translation block. The table below describes one set of security profiles that has been proposed for use with the AIPSTZ.

Table 32-4. Security Levels

CSU_SEC_LEVEL	Non-Secure User	Non-Secure Supervisor	Secure User	Secure Supervisor
0	RD+WR	RD+WR	RD+WR	RD+WR
1	NOT ALLOWED	RD+WR	RD+WR	RD+WR
2	Read Only	Read Only	RD+WR	RD+WR
3	NOT ALLOWED	Read Only	RD+WR	RD+WR
4	NOT ALLOWED	NOT ALLOWED	RD+WR	RD+WR
5	NOT ALLOWED	NOT ALLOWED	NOT ALLOWED	RD+WR
6	NOT ALLOWED	NOT ALLOWED	Read Only	Read Only
7	NOT ALLOWED	NOT ALLOWED	NOT ALLOWED	NOT ALLOWED

Information regarding CSU is provided in the Security Reference Manual. Contact your NXP representative for information about obtaining this document.

A 3-bit input, 8-bit output translation block can be used such that only three register bits are required to set the security profile and the translation block will drive the correct 8-bit configuration vector. Each peripheral connected to the AIPSTZ would require this translation block. The top level AIPSTZ has this three bit input line `csu_sec_level[2:0]' corresponding to each peripheral X.

32.8 AIPSTZ Memory Map/Register Definition

The memory map for the AIPS SW-visible registers is shown in the table below.

The MPROT and OPACR fields are 4 bits in width. Some bits may be reserved depending on device.

AIPSTZ memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4007_C000	Master Priviledge Registers (AIPSTZ1_MPR)	32	R/W	7700_0000h	32.8.1/1799
4007_C040	Off-Platform Peripheral Access Control Registers (AIPSTZ1_OPACR)	32	R/W	4444_4444h	32.8.2/1801
4007_C044	Off-Platform Peripheral Access Control Registers (AIPSTZ1_OPACR1)	32	R/W	4444_4444h	32.8.3/1804
4007_C048	Off-Platform Peripheral Access Control Registers (AIPSTZ1_OPACR2)	32	R/W	4444_4444h	32.8.4/1807
4007_C04C	Off-Platform Peripheral Access Control Registers (AIPSTZ1_OPACR3)	32	R/W	4444_4444h	32.8.5/1810
4007_C050	Off-Platform Peripheral Access Control Registers (AIPSTZ1_OPACR4)	32	R/W	4444_4444h	32.8.6/1813
4017_C000	Master Priviledge Registers (AIPSTZ2_MPR)	32	R/W	7700_0000h	32.8.1/1799
4017_C040	Off-Platform Peripheral Access Control Registers (AIPSTZ2_OPACR)	32	R/W	4444_4444h	32.8.2/1801
4017_C044	Off-Platform Peripheral Access Control Registers (AIPSTZ2_OPACR1)	32	R/W	4444_4444h	32.8.3/1804
4017_C048	Off-Platform Peripheral Access Control Registers (AIPSTZ2_OPACR2)	32	R/W	4444_4444h	32.8.4/1807
4017_C04C	Off-Platform Peripheral Access Control Registers (AIPSTZ2_OPACR3)	32	R/W	4444_4444h	32.8.5/1810
4017_C050	Off-Platform Peripheral Access Control Registers (AIPSTZ2_OPACR4)	32	R/W	4444_4444h	32.8.6/1813
4027_C000	Master Priviledge Registers (AIPSTZ3_MPR)	32	R/W	7700_0000h	32.8.1/1799
4027_C040	Off-Platform Peripheral Access Control Registers (AIPSTZ3_OPACR)	32	R/W	4444_4444h	32.8.2/1801
4027_C044	Off-Platform Peripheral Access Control Registers (AIPSTZ3_OPACR1)	32	R/W	4444_4444h	32.8.3/1804
4027_C048	Off-Platform Peripheral Access Control Registers (AIPSTZ3_OPACR2)	32	R/W	4444_4444h	32.8.4/1807
4027_C04C	Off-Platform Peripheral Access Control Registers (AIPSTZ3_OPACR3)	32	R/W	4444_4444h	32.8.5/1810
4027_C050	Off-Platform Peripheral Access Control Registers (AIPSTZ3_OPACR4)	32	R/W	4444_4444h	32.8.6/1813

AIPSTZ memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4037_C000	Master Priviledge Registers (AIPSTZ4_MPR)	32	R/W	7700_0000h	32.8.1/1799
4037_C040	Off-Platform Peripheral Access Control Registers (AIPSTZ4_OPACR)	32	R/W	4444_4444h	32.8.2/1801
4037_C044	Off-Platform Peripheral Access Control Registers (AIPSTZ4_OPACR1)	32	R/W	4444_4444h	32.8.3/1804
4037_C048	Off-Platform Peripheral Access Control Registers (AIPSTZ4_OPACR2)	32	R/W	4444_4444h	32.8.4/1807
4037_C04C	Off-Platform Peripheral Access Control Registers (AIPSTZ4_OPACR3)	32	R/W	4444_4444h	32.8.5/1810
4037_C050	Off-Platform Peripheral Access Control Registers (AIPSTZ4_OPACR4)	32	R/W	4444_4444h	32.8.6/1813

32.8.1 Master Priviledge Registers (AIPSTZx_MPR)

Each AIPSTZ_MPR specifies 16 4-bit fields defining the access privilege level associated with a bus master in the platform, as well as specifying whether write accesses from this master are bufferable shown in Table 32-5

The registers provide one field per bus master, where field 15 corresponds to master 15, field 14 to master 14,... field 0 to master 0 (typically the processor core). The master index allocation is shown in the table below.

Table 32-5. MPROT Field

Bit	Field	Description
3	MBW	**Master Buffer Writes** - This bit determines whether the AIPSTZ is enabled to buffer writes from this master.
2	MTR	**Master Trusted for Reads** - This bit determines whether the master is trusted for read accesses.
1	MTW	**Master Trusted for Writes** - This bit determines whether the master is trusted for write accesses.
0	MPL	**Master Privilege Level** - This bit determines how the privilege level of the master is determined.

NOTE

The reset value is set to 0000_0000_7700_0000, which makes master 0 and master 1 (Arm CORE) the trusted masters. Trusted software can change the settings after reset.

Table 32-6. Master Index Allocation

Master Index	Master Name	Comments
Master 0	Arm core	
Master 1	eDMA	
Master 2	DCP	
Master 3	Others	Share the same number allocation.
Master 4-15	Reserved	

Address: Base address + 0h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MPROT0				MPROT1				MPROT2				MPROT3				Reserved				Reserved				Reserved				Reserved			
Reset	0	1	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

AIPSTZx_MPR field descriptions

Field	Description
31–28 MPROT0	Master 0 Priviledge, Buffer, Read, Write Control xxx0 **MPL0** — Accesses from this master are forced to user-mode (ips_supervisor_access is forced to zero) regardless of the hprot[1] access attribute. xxx1 **MPL1** — Accesses from this master are not forced to user-mode. The hprot[1] access attribute is used directly to determine ips_supervisor_access. xx0x **MTW0** — This master is not trusted for write accesses. xx1x **MTW1** — This master is trusted for write accesses. x0xx **MTR0** — This master is not trusted for read accesses. x1xx **MTR1** — This master is trusted for read accesses. 0xxx **MBW0** — Write accesses from this master are not bufferable 1xxx **MBW1** — Write accesses from this master are allowed to be buffered
27–24 MPROT1	Master 1 Priviledge, Buffer, Read, Write Control xxx0 **MPL0** — Accesses from this master are forced to user-mode (ips_supervisor_access is forced to zero) regardless of the hprot[1] access attribute. xxx1 **MPL1** — Accesses from this master are not forced to user-mode. The hprot[1] access attribute is used directly to determine ips_supervisor_access. xx0x **MTW0** — This master is not trusted for write accesses. xx1x **MTW1** — This master is trusted for write accesses. x0xx **MTR0** — This master is not trusted for read accesses. x1xx **MTR1** — This master is trusted for read accesses. 0xxx **MBW0** — Write accesses from this master are not bufferable 1xxx **MBW1** — Write accesses from this master are allowed to be buffered
23–20 MPROT2	Master 2 Priviledge, Buffer, Read, Write Control xxx0 **MPL0** — Accesses from this master are forced to user-mode (ips_supervisor_access is forced to zero) regardless of the hprot[1] access attribute. xxx1 **MPL1** — Accesses from this master are not forced to user-mode. The hprot[1] access attribute is used directly to determine ips_supervisor_access. xx0x **MTW0** — This master is not trusted for write accesses. xx1x **MTW1** — This master is trusted for write accesses.

Table continues on the next page...

AIPSTZx_MPR field descriptions (continued)

Field	Description
	x0xx **MTR0** — This master is not trusted for read accesses. x1xx **MTR1** — This master is trusted for read accesses. 0xxx **MBW0** — Write accesses from this master are not bufferable 1xxx **MBW1** — Write accesses from this master are allowed to be buffered
19–16 MPROT3	Master 3 Priviledge, Buffer, Read, Write Control. xxx0 **MPL0** — Accesses from this master are forced to user-mode (ips_supervisor_access is forced to zero) regardless of the hprot[1] access attribute. xxx1 **MPL1** — Accesses from this master are not forced to user-mode. The hprot[1] access attribute is used directly to determine ips_supervisor_access. xx0x **MTW0** — This master is not trusted for write accesses. xx1x **MTW1** — This master is trusted for write accesses. x0xx **MTR0** — This master is not trusted for read accesses. x1xx **MTR1** — This master is trusted for read accesses. 0xxx **MBW0** — Write accesses from this master are not bufferable 1xxx **MBW1** — Write accesses from this master are allowed to be buffered
15–12 -	This field is reserved. Reserved
11–8 -	This field is reserved. Reserved
7–4 -	This field is reserved. Reserved
-	This field is reserved. Reserved

32.8.2 Off-Platform Peripheral Access Control Registers (AIPSTZx_OPACR)

Each of the off-platform peripherals have an Off-platform Peripheral Access Control Register (AIPSTZ_OPACR) which defines the access levels supported by the given block.

Each AIPSTZ_OPACR has the following format shown in Table 32-7

Table 32-7. OPAC Field

Bit	Field	Description
3	BW	**Buffer Writes** - This bit determines whether write accesses to this peripheral are allowed to be buffered.[1]
2	SP	**Supervisor Protect** - This bit determines whether the peripheral requires supervisor privilege level for access.
1	WP	**Write Protect** - This bit determines whether the peripheral allows write accesses.
0	TP	**Trusted Protect** - This bit determines whether the peripheral allows accesses from an untrusted master.

1. Buffered writes are not available for AIPSTZ. This bit should be set to '0'.

Address: Base address + 40h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	OPAC0				OPAC1				OPAC2				OPAC3				OPAC4				OPAC5				OPAC6				OPAC7			
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

AIPSTZx_OPACR field descriptions

Field	Description
31–28 OPAC0	Off-platform Peripheral Access Control 0 xxx0 **TP0** — Accesses from an untrusted master are allowed. xxx1 **TP1** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP0** — This peripheral allows write accesses. xx1x **WP1** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP0** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP1** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW0** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW1** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
27–24 OPAC1	Off-platform Peripheral Access Control 1 xxx0 **TP0** — Accesses from an untrusted master are allowed. xxx1 **TP1** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP0** — This peripheral allows write accesses. xx1x **WP1** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP0** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP1** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW0** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW1** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
23–20 OPAC2	Off-platform Peripheral Access Control 2 xxx0 **TP0** — Accesses from an untrusted master are allowed. xxx1 **TP1** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP0** — This peripheral allows write accesses. xx1x **WP1** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP0** — This peripheral does not require supervisor privilege level for accesses.

Table continues on the next page...

AIPSTZx_OPACR field descriptions (continued)

Field	Description
	x1xx **SP1** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW0** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW1** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
19–16 OPAC3	Off-platform Peripheral Access Control 3 xxx0 **TP0** — Accesses from an untrusted master are allowed. xxx1 **TP1** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP0** — This peripheral allows write accesses. xx1x **WP1** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP0** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP1** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW0** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW1** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
15–12 OPAC4	Off-platform Peripheral Access Control 4 xxx0 **TP0** — Accesses from an untrusted master are allowed. xxx1 **TP1** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP0** — This peripheral allows write accesses. xx1x **WP1** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP0** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP1** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW0** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW1** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
11–8 OPAC5	Off-platform Peripheral Access Control 5 xxx0 **TP0** — Accesses from an untrusted master are allowed. xxx1 **TP1** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP0** — This peripheral allows write accesses. xx1x **WP1** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP0** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP1** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the

Table continues on the next page...

AIPSTZx_OPACR field descriptions (continued)

Field	Description
	master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW0** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW1** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
7–4 OPAC6	Off-platform Peripheral Access Control 6 xxx0 **TP0** — Accesses from an untrusted master are allowed. xxx1 **TP1** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP0** — This peripheral allows write accesses. xx1x **WP1** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP0** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP1** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW0** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW1** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
OPAC7	Off-platform Peripheral Access Control 7 xxx0 **TP0** — Accesses from an untrusted master are allowed. xxx1 **TP1** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP0** — This peripheral allows write accesses. xx1x **WP1** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP0** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP1** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW0** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW1** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.

32.8.3 Off-Platform Peripheral Access Control Registers (AIPSTZx_OPACR1)

Each of the off-platform peripherals have an Off-platform Peripheral Access Control Register (AIPSTZ_OPACR) which defines the access levels supported by the given block.

Each AIPSTZ_OPACR has the following format shown in Table 32-7

Address: Base address + 44h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	OPAC8				OPAC9				OPAC10				OPAC11				OPAC12				OPAC13				OPAC14				OPAC15			
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

AIPSTZx_OPACR1 field descriptions

Field	Description
31–28 OPAC8	Off-platform Peripheral Access Control 8 xxx0 **TP0** — Accesses from an untrusted master are allowed. xxx1 **TP1** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP0** — This peripheral allows write accesses. xx1x **WP1** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP0** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP1** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW0** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW1** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
27–24 OPAC9	Off-platform Peripheral Access Control 9 xxx0 **TP0** — Accesses from an untrusted master are allowed. xxx1 **TP1** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP0** — This peripheral allows write accesses. xx1x **WP1** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP0** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP1** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW0** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW1** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
23–20 OPAC10	Off-platform Peripheral Access Control 10 xxx0 **TP0** — Accesses from an untrusted master are allowed. xxx1 **TP1** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP0** — This peripheral allows write accesses. xx1x **WP1** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP0** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP1** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the

Table continues on the next page...

AIPSTZx_OPACR1 field descriptions (continued)

Field	Description
	master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW0** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW1** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
19–16 OPAC11	Off-platform Peripheral Access Control 11 xxx0 **TP0** — Accesses from an untrusted master are allowed. xxx1 **TP1** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP0** — This peripheral allows write accesses. xx1x **WP1** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP0** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP1** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW0** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW1** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
15–12 OPAC12	Off-platform Peripheral Access Control 12 xxx0 **TP0** — Accesses from an untrusted master are allowed. xxx1 **TP1** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP0** — This peripheral allows write accesses. xx1x **WP1** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP0** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP1** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW0** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW1** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
11–8 OPAC13	Off-platform Peripheral Access Control 13 xxx0 **TP0** — Accesses from an untrusted master are allowed. xxx1 **TP1** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP0** — This peripheral allows write accesses. xx1x **WP1** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP0** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP1** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus.

Table continues on the next page...

AIPSTZx_OPACR1 field descriptions (continued)

Field	Description
	0xxx **BW0** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW1** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
7–4 OPAC14	Off-platform Peripheral Access Control 14 xxx0 **TP0** — Accesses from an untrusted master are allowed. xxx1 **TP1** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP0** — This peripheral allows write accesses. xx1x **WP1** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP0** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP1** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW0** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW1** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
OPAC15	Off-platform Peripheral Access Control 15 xxx0 **TP0** — Accesses from an untrusted master are allowed. xxx1 **TP1** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP0** — This peripheral allows write accesses. xx1x **WP1** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP0** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP1** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW0** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW1** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.

32.8.4 Off-Platform Peripheral Access Control Registers (AIPSTZx_OPACR2)

Each of the off-platform peripherals have an Off-platform Peripheral Access Control Register (AIPSTZ_OPACR) which defines the access levels supported by the given block.

Each AIPSTZ_OPACR has the following format shown in Table 32-7

Address: Base address + 48h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	OPAC16				OPAC17				OPAC18				OPAC19				OPAC20				OPAC21				OPAC22				OPAC23			
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

AIPSTZx_OPACR2 field descriptions

Field	Description
31–28 OPAC16	Off-platform Peripheral Access Control 16 xxx0 **TP0** — Accesses from an untrusted master are allowed. xxx1 **TP1** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP0** — This peripheral allows write accesses. xx1x **WP1** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP0** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP1** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW0** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW1** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
27–24 OPAC17	Off-platform Peripheral Access Control 17 xxx0 **TP0** — Accesses from an untrusted master are allowed. xxx1 **TP1** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP0** — This peripheral allows write accesses. xx1x **WP1** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP0** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP1** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW0** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW1** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
23–20 OPAC18	Off-platform Peripheral Access Control 18 xxx0 **TP0** — Accesses from an untrusted master are allowed. xxx1 **TP1** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP0** — This peripheral allows write accesses. xx1x **WP1** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP0** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP1** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the

Table continues on the next page...

AIPSTZx_OPACR2 field descriptions (continued)

Field	Description
	master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW0** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW1** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
19–16 OPAC19	Off-platform Peripheral Access Control 19 xxx0 **TP0** — Accesses from an untrusted master are allowed. xxx1 **TP1** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP0** — This peripheral allows write accesses. xx1x **WP1** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP0** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP1** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW0** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW1** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
15–12 OPAC20	Off-platform Peripheral Access Control 20 xxx0 **TP0** — Accesses from an untrusted master are allowed. xxx1 **TP1** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP0** — This peripheral allows write accesses. xx1x **WP1** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP0** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP1** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW0** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW1** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
11–8 OPAC21	Off-platform Peripheral Access Control 21 xxx0 **TP0** — Accesses from an untrusted master are allowed. xxx1 **TP1** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP0** — This peripheral allows write accesses. xx1x **WP1** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP0** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP1** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus.

Table continues on the next page...

AIPSTZx_OPACR2 field descriptions (continued)

Field	Description
	0xxx **BW0** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW1** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
7–4 OPAC22	Off-platform Peripheral Access Control 22 xxx0 **TP0** — Accesses from an untrusted master are allowed. xxx1 **TP1** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP0** — This peripheral allows write accesses. xx1x **WP1** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP0** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP1** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW0** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW1** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
OPAC23	Off-platform Peripheral Access Control 23 xxx0 **TP0** — Accesses from an untrusted master are allowed. xxx1 **TP1** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP0** — This peripheral allows write accesses. xx1x **WP1** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP0** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP1** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW0** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW1** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.

32.8.5 Off-Platform Peripheral Access Control Registers (AIPSTZx_OPACR3)

Each of the off-platform peripherals have an Off-platform Peripheral Access Control Register (AIPSTZ_OPACR) which defines the access levels supported by the given block.

Each AIPSTZ_OPACR has the following format shown in Table 32-7

Address: Base address + 4Ch offset

Bit	31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
R W	OPAC24	OPAC25	OPAC26	OPAC27	OPAC28	OPAC29	OPAC30	OPAC31
Reset	0 1 0 0	0 1 0 0	0 1 0 0	0 1 0 0	0 1 0 0	0 1 0 0	0 1 0 0	0 1 0 0

AIPSTZx_OPACR3 field descriptions

Field	Description
31–28 OPAC24	Off-platform Peripheral Access Control 24 xxx0 **TP0** — Accesses from an untrusted master are allowed. xxx1 **TP1** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP0** — This peripheral allows write accesses. xx1x **WP1** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP0** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP1** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW0** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW1** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
27–24 OPAC25	Off-platform Peripheral Access Control 25 xxx0 **TP0** — Accesses from an untrusted master are allowed. xxx1 **TP1** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP0** — This peripheral allows write accesses. xx1x **WP1** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP0** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP1** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW0** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW1** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
23–20 OPAC26	Off-platform Peripheral Access Control 26 xxx0 **TP0** — Accesses from an untrusted master are allowed. xxx1 **TP1** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP0** — This peripheral allows write accesses. xx1x **WP1** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP0** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP1** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the

Table continues on the next page...

AIPSTZx_OPACR3 field descriptions (continued)

Field	Description
	master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW0** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW1** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
19–16 OPAC27	Off-platform Peripheral Access Control 27 xxx0 **TP0** — Accesses from an untrusted master are allowed. xxx1 **TP1** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP0** — This peripheral allows write accesses. xx1x **WP1** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP0** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP1** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW0** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW1** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
15–12 OPAC28	Off-platform Peripheral Access Control 28 xxx0 **TP0** — Accesses from an untrusted master are allowed. xxx1 **TP1** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP0** — This peripheral allows write accesses. xx1x **WP1** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP0** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP1** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW0** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW1** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
11–8 OPAC29	Off-platform Peripheral Access Control 29 xxx0 **TP0** — Accesses from an untrusted master are allowed. xxx1 **TP1** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP0** — This peripheral allows write accesses. xx1x **WP1** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP0** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP1** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus.

Table continues on the next page...

AIPSTZx_OPACR3 field descriptions (continued)

Field	Description
	0xxx **BW0** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW1** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
7–4 OPAC30	Off-platform Peripheral Access Control 30 xxx0 **TP0** — Accesses from an untrusted master are allowed. xxx1 **TP1** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP0** — This peripheral allows write accesses. xx1x **WP1** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP0** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP1** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW0** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW1** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
OPAC31	Off-platform Peripheral Access Control 31 xxx0 **TP0** — Accesses from an untrusted master are allowed. xxx1 **TP1** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP0** — This peripheral allows write accesses. xx1x **WP1** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP0** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP1** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW0** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW1** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.

32.8.6 Off-Platform Peripheral Access Control Registers (AIPSTZx_OPACR4)

Each of the off-platform peripherals have an Off-platform Peripheral Access Control Register (AIPSTZ_OPACR) which defines the access levels supported by the given block.

Each AIPSTZ_OPACR has the following format shown in Table 32-7

Address: Base address + 50h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	OPAC32				OPAC33				Reserved																							
Reset	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0

AIPSTZx_OPACR4 field descriptions

Field	Description
31–28 OPAC32	Off-platform Peripheral Access Control 32 xxx0 **TP0** — Accesses from an untrusted master are allowed. xxx1 **TP1** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP0** — This peripheral allows write accesses. xx1x **WP1** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP0** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP1** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW0** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW1** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
27–24 OPAC33	Off-platform Peripheral Access Control 33 xxx0 **TP0** — Accesses from an untrusted master are allowed. xxx1 **TP1** — Accesses from an untrusted master are not allowed. If an access is attempted by an untrusted master, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. xx0x **WP0** — This peripheral allows write accesses. xx1x **WP1** — This peripheral is write protected. If a write access is attempted, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. x0xx **SP0** — This peripheral does not require supervisor privilege level for accesses. x1xx **SP1** — This peripheral requires supervisor privilege level for accesses. The master privilege level must indicate supervisor via the hprot[1] access attribute, and the MPROTx[MPL] control bit for the master must be set. If not, the access is terminated with an error response and no peripheral access is initiated on the IPS bus. 0xxx **BW0** — Write accesses to this peripheral are not bufferable by the AIPSTZ. 1xxx **BW1** — Write accesses to this peripheral are allowed to be buffered by the AIPSTZ.
-	This field is reserved. Reserved

Chapter 33
Display and Camera Overview

33.1 Overview

The following modules are a part of the Display and Camera system:

- LCD Interface (eLCDIF): 24-bit parallel RGB LCD interface
- PXP pixel pipeline: 2D graphics and pixel/image processing engine
- Camera Sensor Interface (CSI): up to 24-bit parallel interface for image sensor

The following figure shows the high-level integration scheme of the chip display and camera system.

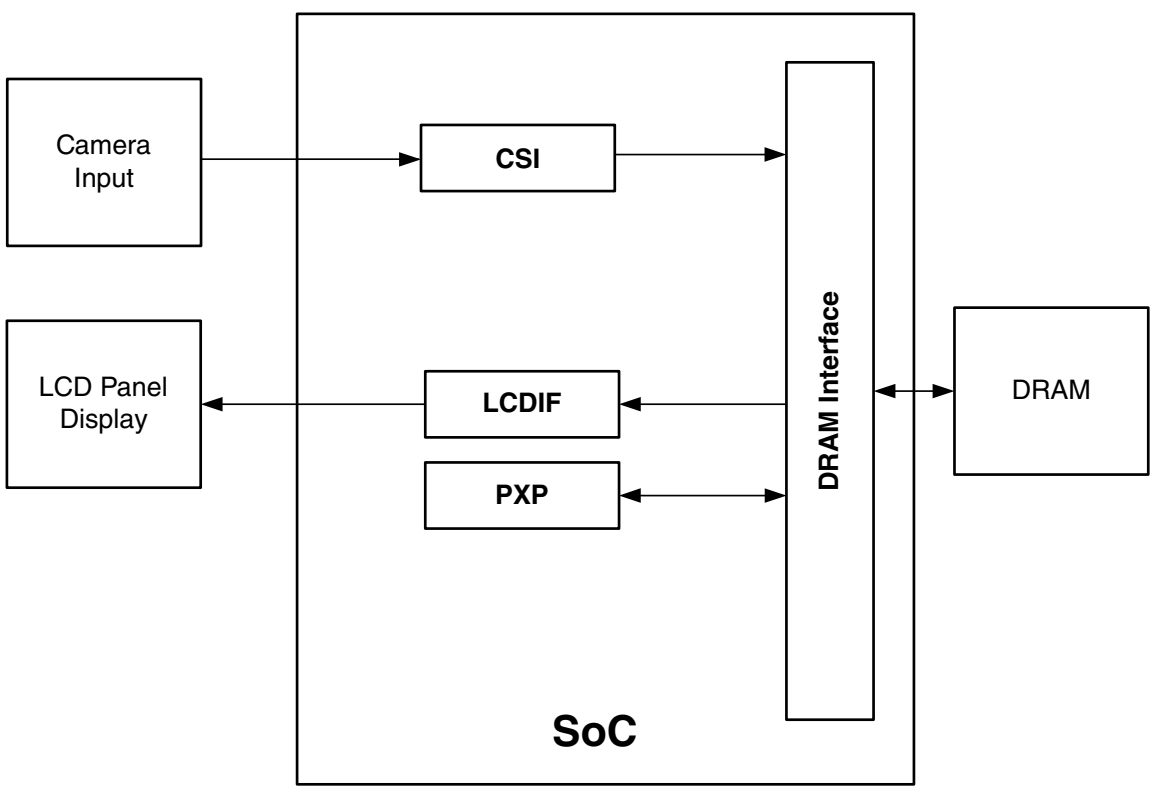


Figure 33-1. Block Diagram

33.1.1 CMOS Sensor Interface (CSI)

The CSI enables the chip to connect directly to external CMOS image sensors. CMOS image sensors are separated into two classes, dumb and smart. Dumb sensors are those that support only traditional sensor timing (Vertical SYNC, Horizontal SYNC), and output RGB, YUV, YCbCr, RAW formats, as well as Bayer and statistics data. The smart sensors support BT.656 video decoder formats and perform additional processing of the image (for example, image compression, image pre-filtering, and various data output formats).

33.1.2 Enhanced LCD Interface (eLCDIF)

The eLCDIF is a general purpose display controller that is used to drive a wide range of display devices. These displays can vary in size and capability. There are other popular displays that support moving pictures and require the RGB interface mode (called DOTCLK interface in this document) for high-speed data transfers.

The block has several major features:

- Bus master interface to source frame buffer data for display refresh and a DMA interface to manage input data transfers from the LCD requiring minimal CPU overhead.
- 8/16/18/24 bit LCD data bus support available depending on I/O mux options.
- Programmable timing and parameters for DOTCLK LCD interfaces to support a wide variety of displays.

33.1.3 Pixel Processing Pipeline (PXP)

The pixel processing pipeline is used to perform image processing on image/video buffers before sending to an LCD display.

The main features of PXP include:

- Multiple input/output format support, including YUV/RGB/Grayscale
- Supports both RGB/YUV scaling
- Supports overlay with Alpha blending
- Supports Rotation of 0, 90, 180, and 270 degrees in conjunction with vertical and horizontal flip options

Chapter 34
CMOS Sensor Interface (CSI)

34.1 Chip-specific CSI information

Table 34-1. Reference links to related information

Topic	Related module(s)	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
Interrupts, DMA Events and XBAR Assignments	-	Interrupts, DMA Events and XBAR Assignments

34.2 Overview

This chapter presents the CMOS Sensor Interface (CSI) architecture, operation principles, and programming model. The CSI enables the chip to connect directly to external CMOS image sensors.

CMOS image sensors are separated into two classes, dumb and smart. Dumb sensors are those that support only traditional sensor timing (Vertical SYNC and Horizontal SYNC) and output only RGB, YUV, and Bayer and statistics data, while smart sensors support ITU-R BT.656 video decoder formats and perform additional processing of the image (for example, image compression, image pre-filtering, and various data output formats). Statistics only work for Bayer data is 8-bit per pixel format.

34.2.1 Block Diagram

The block diagram of the CSI module is shown below:

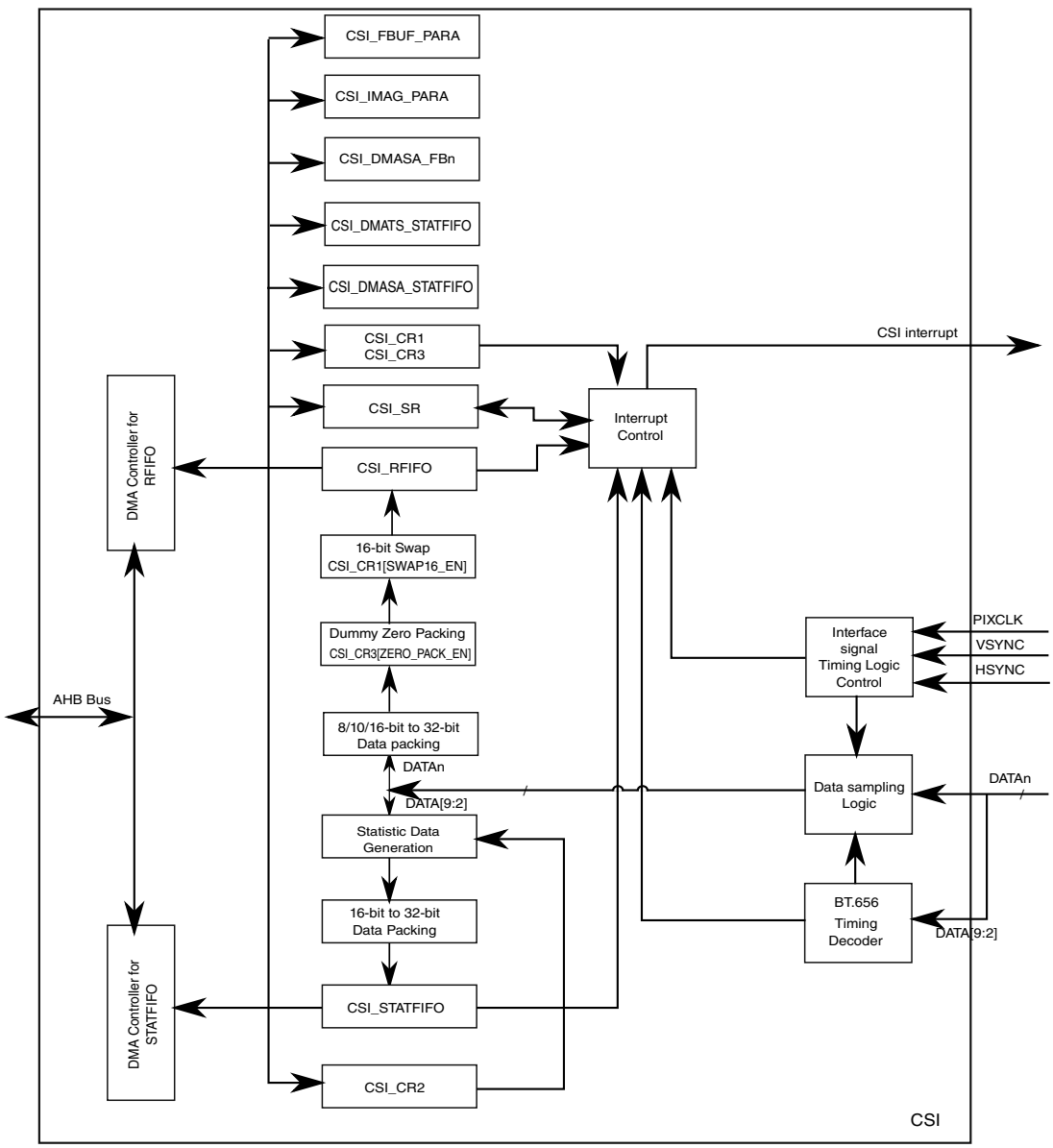


Figure 34-1. Block Diagram

34.2.2 Features

The features of the CSI include:

- Configurable interface logic to support most commonly available CMOS sensors.

- Support for ITU-R BT.656 video interface as well as traditional sensor interface.
- 8-bit / 10-bit / 16-bit data port for Bayer data input.
- Full control of 8-bit/pixel, 10-bit/pixel or 16-bit / pixel data format to 64-bit receive FIFO packing.
- Receive FIFO overrun protection mechanism.
- Embedded DMA controllers to transfer data from receive FIFO or statistic FIFO through AHB bus.
- Support 2D (data is chosen as x/y coordinate) DMA transfer from the receive FIFO to the frame buffers in the external memory.
- Support double buffering two frames in the external memory.
- Single interrupt source to interrupt controller from maskable interrupt sources: Start of Frame, End of Frame, Change of Field, FIFO full, FIFO overrun, DMA transfer done, BT.656 error and AHB bus response error.
- Configurable master clock frequency output to sensor.

34.3 Functional Description

34.3.1 Principles of Operation

The information found here describes the modes of operation of the sensor interface.

The CSI is designed to support generic sensor interface timing as well as ITU-R BT.656 video interface timing. Traditional CMOS sensors typically use VSYNC (SOF), HSYNC (BLANK), and PIXCLK signals to output Bayer, RGB or YUV data. Smart CMOS sensors, that come with on-chip imaging processing, usually support video mode transfer. They use an embedded timing codec to replace the VSYNC and HSYNC signal. The timing codec is defined by the ITU-R BT.656 standard.

The CSI can support connection with the sensor as follows.

- To connect with one 8-bit sensor, the sensor data interface should connect to CSI_DATA[9:2].
- To connect with one 10-bit sensor, the sensor data interface should connect to CSI_DATA[9:0].
- To connect with one 12-bit sensor, the sensor data interface should connect to CSI_DATA[11:0].
- To connect with one 16-bit sensor, the sensor data interface should connect to CSI_DATA[15:0].

Table 34-2. CSI input data format

Data line	YCbCr	RGB888	RGB888/ YUV4444	RGB666	RGB565	YCbCr422	YCbCr422	Generic 10 bit	ITU-R BT. 656
DATA[23]	Y[7]	R[7]		R[5]					
DATA[22]	Y[6]	R[6]		R[4]					
DATA[21]	Y[5]	R[5]		R[3]					
DATA[20]	Y[4]	R[4]		R[2]					
DATA[19]	Y[3]	R[3]		R[1]					
DATA[18]	Y[2]	R[2]		R[0]					
DATA[17]	Y[1]	R[1]		Y[5]					
DATA[16]	Y[0]	R[0]		R[4]					
DATA[15]	Cb[7]	G[7]		G[5]	R[4]	Y[7]			
DATA[14]	Cb[6]	G[6]		G[4]	R[3]	Y[6]			
DATA[13]	Cb[5]	G[5]		G[3]	R[2]	Y[5]			
DATA[12]	Cb[4]	G[4]		G[2]	R[1]	Y[4]			
DATA[11]	Cb[3]	G[3]		G[1]	R[0]	Y[3]			
DATA[10]	Cb[2]	G[2]		G[0]	G[5]	Y[2]			
DATA[9]	Cb[1]	G[1]	R/G/B[7]	G[5]	G[4]	Y[1]	Y/C[7]	Ge[9]	C/Y[7]
DATA[8]	Cb[0]	G[0]	R/G/B[6]	G[4]	G[3]	Y[0]	Y/C[6]	Ge[8]	C/Y[6]
DATA[7]	Cr[7]	B[7]	R/G/B[5]	B[5]	G[2]	C[7]	Y/C[5]	Ge[7]	C/Y[5]
DATA[6]	Cr[6]	B[6]	R/G/B[4]	B[4]	G[1]	C[6]	Y/C[4]	Ge[6]	C/Y[4]
DATA[5]	Cr[5]	B[5]	R/G/B[3]	B[3]	G[0]	C[5]	Y/C[3]	Ge[5]	C/Y[3]
DATA[4]	Cr[4]	B[4]	R/G/B[2]	B[2]	B[4]	C[4]	Y/C[2]	Ge[4]	C/Y[2]
DATA[3]	Cr[3]	B[3]	R/G/B[1]	B[1]	B[3]	C[3]	Y/C[1]	Ge[3]	C/Y[1]
DATA[2]	Cr[2]	B[2]	R/G/B[0]	B[0]	B[2]	C[2]	Y/C[0]	Ge[2]	C/Y[0]
DATA[1]	Cr[1]	B[1]		B[5]	B[1]	C[1]		Ge[1]	
DATA[0]	Cr[0]	B[0]		B[4]	B[0]	C[0]		Ge[0]	

34.3.1.1 Data Transfer with the Embedded DMA Controllers

The CSI has two embedded DMA controllers, one for the receive FIFO and the other for the statistic FIFO. It supports 2D DMA transfer from the receive FIFO to the frame buffers in the external memory and linear DMA transfer from the statistic FIFO.

NOTE

One word = 4 bytes

Double words = 8 bytes

To transfer data from the RxFIFO to the external memory, the user should set the start address in the frame buffer where the transferred data is stored, the parameters of the frame buffers, and the parameters of the image coming from the sensor. The user can have two frame buffers in the external memory. Each one will store a frame of image coming from the sensor. The embedded DMA controller will first write the frame buffer1 and then frame buffer2. These two frame buffers will be written by turns. The start address should be aligned in double words and set in the CSI_DMASA-FB1 and CSI_DMASA-FB2 registers. In the CSI_FBUF_PARA register, the user should set the stride of the frame buffer to show how many double words to skip before starting to write the next row of the image. In the CSI_IMAG_PARA register, the user should set the width and height of the image coming from the sensor. The CSI_CR3[RxFF_LEVEL] and CSI_CR3[DMA_REQ_EN_RFF] bits also need to be set before the data transfer starts. When the number of the data in the RxFIFO reaches the trigger level, a DMA request will be sent to the embedded DMA controller and the data will be read out from the RxFIFO and written through AHB bus into the external frame buffers. The burst type of transfer can be INCR4, INCR8 and INCR16 by setting CSI_CR2[DMA_BURST_TYPE_RFF] bits. INCR4/INCR8/INCR16 are AHB protocols. After all data in an image frame are transferred, the CSI_SR[DMA_TSF_DONE_FB1] or CSI_SR[DMA_TSF_DONE_FB2] bit will be set and the interrupt can be triggered if the corresponding enable bit is set in CSI_CR1 register. The CSI_CR3[DMA_REFLASH_RFF] bit can be used to activate or restart the embedded DMA controller.

The RxFIFO has the overrun protection mechanism in case the RxFIFO is overrun during data transfer. If the RxFIFO is full and more data needs to be received during the data transfer, the RxFIFO will be overwritten continuously and all 128 words of data in the RxFIFO before overrun occurred will be discarded; the corresponding 128 words memory space in the frame buffer will keep the previous values. In other words, the DMA will skip the 128 word space fragment as if data is written.

To transfer data from the statistic FIFO to the external memory, the user should set the start address of the external memory where the transferred data is stored and the total transfer sizes. The start address and the transfer sizes are all aligned in double words and should be set in the CSIDMASA-STATFIFO and CSIDMATS-STATFIFO registers. The CSI_CR3[STATFF_LEVEL] and CSI_CR3[DMA_REQ_EN_SFF] bits should also be set before the data transfer starts. When the number of the data in the CSI_STATFIFO register reaches the trigger level, a dma request will be sent to the embedded DMA controller and the data will be read out from the STATFIFO and written through AHB bus into the external memory. The burst type of transfer can be INCR4, INCR8 and INCR16 by setting CSI_CR2[DMA_BURST_TYPE_SFF] bit. After all expected data (defined by the total transfer sizes) are transferred, the CSI_SR[DMA_TSF_DONE_SFF] bit will be set and an interrupt can be triggered if the

CSI_CR1[SFF_DMA_DONE_INTEN] is enabled. The CSI_CR3[DMA_REFLASH_SFF] bit can be used to activate or re-start the embedded DMA controller.

34.3.1.2 Gated Clock Mode

VSYNC, HSYNC, and PIXCLK signals are used in gated clock mode.

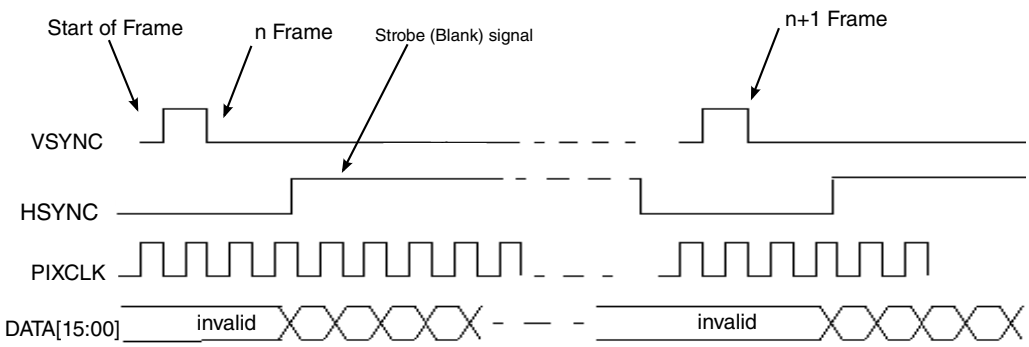


Figure 34-2. Gated Clock Mode Timing Diagram

NOTE

The CSI can be programmed to support rising/falling-edge triggered VSYNC through CSI_CR1[SOF_POL], active-high/low HSYNC using CSI_CR1[HSYNC_POL], and rising/falling-edge triggered PIXCLK through CSI_CR1[REDGE]. The configuration chosen by the user is referred to as an active edge.

A frame starts with an active edge on VSYNC, then HSYNC asserts and holds for the entire line. The Pixel clock is valid as long as HSYNC is asserted. Data is latched at the active edge of the valid pixel clocks. HSYNC deasserts at the end of line. Pixel clocks then become invalid and CSI stops receiving data from the stream. For the next line the HSYNC timing repeats. For the next frame the VSYNC timing repeats.

34.3.1.3 Non-Gated Clock Mode

In non-gated clock mode, only the VSYNC and PIXCLK signals are used; the HSYNC signal is ignored.

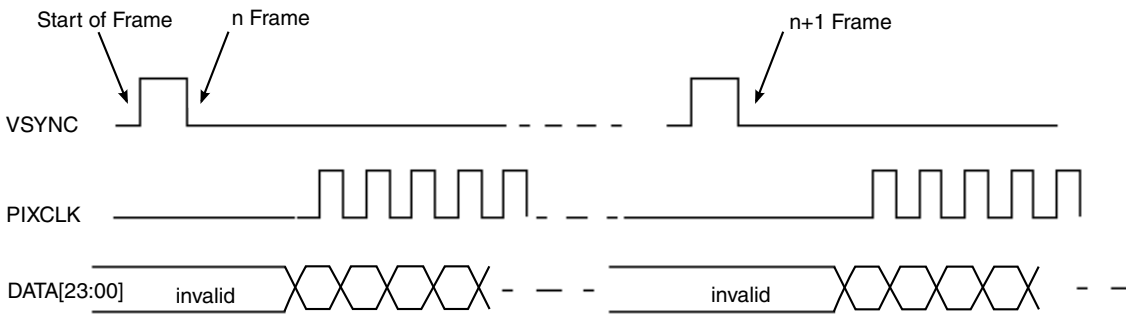


Figure 34-3. Non-Gated Clock Mode Timing Diagram

The overall timing of non-gated mode is the same as the gated-clock mode, except for the HSYNC signal. HSYNC signal is ignored by the CSI. All incoming pixel clocks are valid and cause data to be latched into RxFIFO. The PIXCLK signal is inactive (states low) until valid data is ready to be transmitted over the bus.

Figure 34-3 shows the timing of a typical sensor. Other sensors may have the slightly different timing from that shown. The CSI can be programed to support rising/falling-edge triggered VSYNC, active-high/low HSYNC, and rising/falling-edge triggered PIXCLK.

34.3.1.4 ITU-R BT.656 Interlace Mode

In interlace mode, an even field and an odd field are interlaced to create a frame. An even field consists of even-numbered lines (0,2,4,6, etc.), and an odd field consist of odd-numbered lines (1,3,5,7,etc.). Some vertical and horizontal blanking data is also added to the fields.

In BT.656 interlace mode, only the PIXCLK and CSI_DATA[9:2] signals are used. The start of frame and blank signals are replaced by a timing code which is embedded in the data stream. Each active line starts with an Start of Active Video (SAV) code and ends with an End of Active Video (EAV) code. In some cases, digital blanking is inserted in between EAV and SAV code. The CSI decodes and filters out the timing-coding from the data stream, recovering VSYNC and HSYNC signals for internal use, such as statistical block control. Data is forwarded to the data receive and packing block in a sequential manner without reordering-that is, field 1 followed by field 2. The fields must be reordered in software to get back the original image.

Change of Field (COF) interrupt is triggered upon every field change. The interrupt service routine reads the status register to check for the current field.

According to the BT.656 specification, the image must be in 625/50 PAL or 525/60 NTSC format. Data must be in YCbCr422 format, each pixel contains 2 bytes, either Y + Cr or Y + Cb. These requirements are set for TV systems. The CSI module supports PAL and NTSC format only.

The following figure describes the frame structure in PAL system, showing vertical and horizontal blanking.

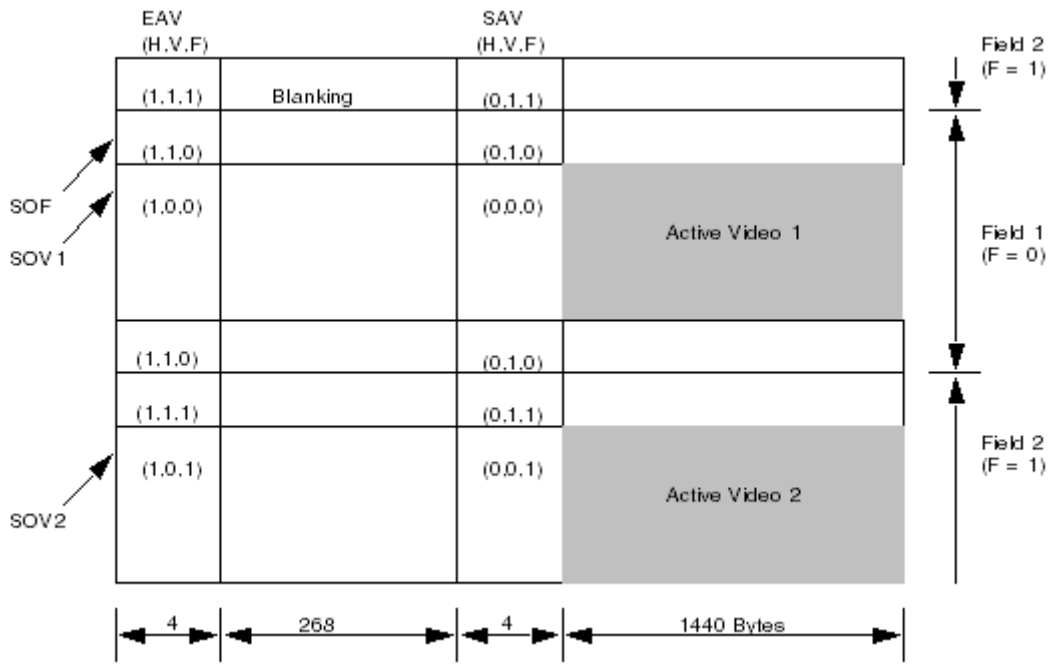


Figure 34-4. BT.656 Interlace Mode (PAL)

The following figure describes the general timing for a single line, showing SAV and EAV.

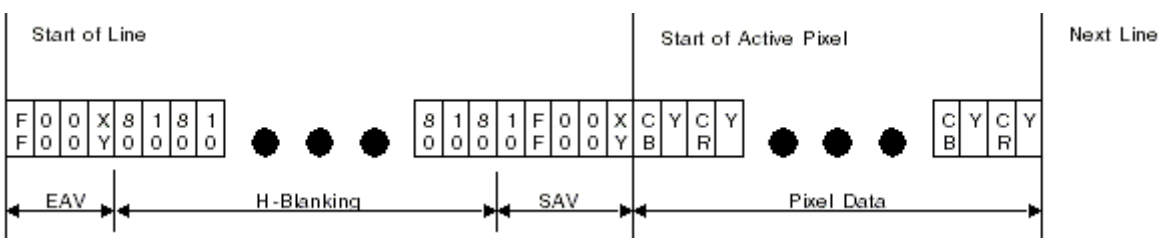


Figure 34-5. BT.656 General Line Timing

The coding tables recommended by the BT.656 specification are shown below. It is used in the BT.656 mode to decode the video stream. An interrupt is generated for SOF, which is decoded from the embedded timing codec.

Table 34-3. Coding for SAV and EAV

Data Bit Number	1st Byte 0xFF	2nd Byte 0x00	3rd Byte 0x00	4th Byte 0xXY
7 (MSB)	1	0	0	1
6	1	0	0	F
5	1	0	0	V
4	1	0	0	H
3	1	0	0	P3
2	1	0	0	P2
1	1	0	0	P1
0	1	0	0	P0

Table 34-4. Codes with Protection bits for Error Detection/Correction

F	V	H	P3	P2	P1	P0
0	0	0	0	0	0	0
0	0	1	1	1	0	1
0	1	0	1	0	1	1
0	1	1	0	1	1	0
1	0	0	0	1	1	1
1	0	1	1	0	1	0
1	1	0	1	1	0	0
1	1	1	0	0	0	1

Table 34-5. Representations by F-Bit

F-Bit	Representations
0	ODD FIELD (FIELD 1)
1	EVEN FIELD (FIELD 2)

34.3.1.5 ITU-R BT.656 Progressive Mode

For a CMOS camera system of VGA or CIF resolution, strict adherence to the interlace requirements stated in the BT.656 standard is not required.

The image is considered to have only 1 active field which is scanned in a progressive manner. This active field is regarded as field 1 and the F-bit in the timing codec is ignored by the decoder. Most sensors support BT.656 timing in this mode (progressive) by default.

The following figure shows the typical flow of progressive mode.

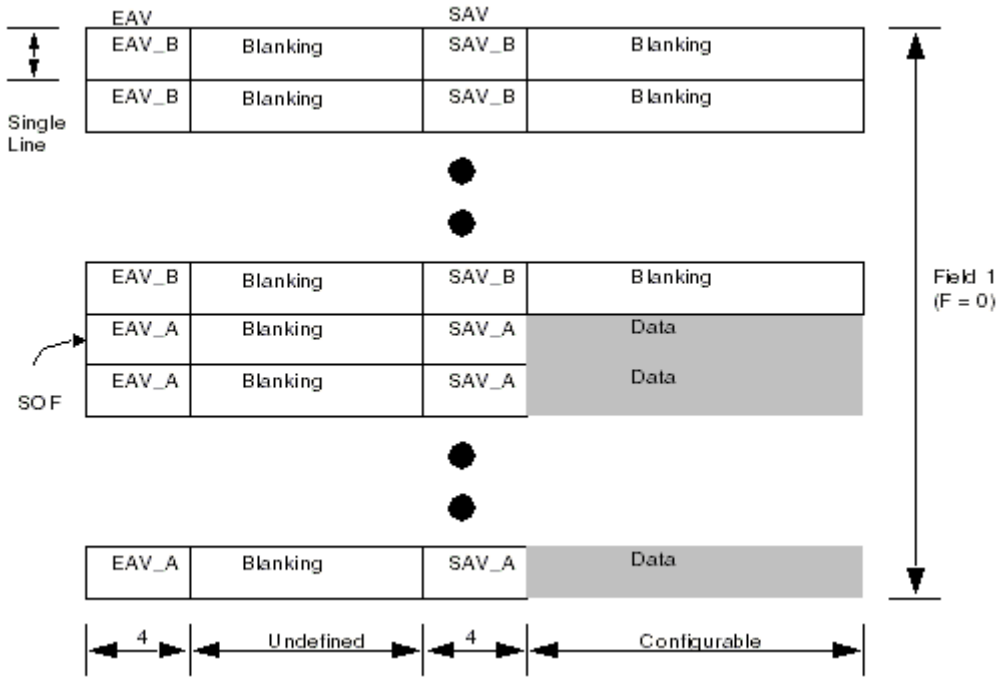


Figure 34-6. BT.656 Progressive Mode (General Case)

An interrupt is generated for SOF but not for COF. In the general case, when SOF information is retrieved from the embedded coding, it is known as internal VSYNC mode. In other cases, when the VSYNC signal is provided by the sensor, it is known as external VSYNC mode. The CSI can be operated in internal or external VSYNC mode.

34.3.1.6 Error Correction for ITU-R BT.656 Coding

According to the algorithm for BT.656 coding, protection bits in the SAV and EAV are encoded in the way that allows a 1-bit error to be corrected, or a 2-bit error to be detected by the decoder. This feature is supported by the interlace mode BT.656 decoder in CSI.

For the 1-bit error case, users can select the error to be corrected automatically, or simply shown as a status flag instead. For the 2-bit error case, because the decoder is unable to make a correction, the error would be shown as a status flag only.

An interrupt can be generated upon the detection of an error. This signal can be enabled or disabled without affecting the operation of the status bit.

34.3.2 Data Packing Style

Careful attention to endianess is needed given the different port sizes at different stages of the image capture path.

To enable flexible packing of image data before storage in the FIFOs, the CSI module can swap data fields by use of the CSI_CR1[PACK_DIR] and CSI_CR1[SWAP16_EN]).

The CSI module accepts 8-bit, 10-bit or 16-bit data from the sensor by configuring CSI_CR1[PIXEL_BIT] and CSI_CR3[SENSOR_16BITS]. The input data is packed according to the setting of CSI_CR1[PACK_DIR] bit. The packed data is stored in the RX FIFO according to the setting of the CSI_CR1[SWAP16_EN].

For 10-bit per pixel data format, each pixel is expanded to 16 bits by appending 6 zeros bits to the most significant bit.

34.3.2.1 RX FIFO Path

34.3.2.1.1 Bayer Data

Bayer data is a type of raw data from the image sensor. This byte-wide data must be converted to the RGB space or YUV space by software. The data path for Bayer data is from the CSI to memory. If the system is in little endian, then the CSI_CR1[PACK_DIR] bit should be set to 0. 8-bit data format from a sensor is packed to 64 bits as P7.P6.P5.P4.P3.P2.P1.P0, where P0 is the pixel coming in time slot 0 (first data) and P3 is the pixel coming in time slot 3 . When the data is addressed as bytes by software, P0 is transferred first, P1 is transferred next, and so on. 10-bit data format is packed to 64 bits as 000000.P3.000000.P2.000000.P1.000000.P0, where P0 is the 10-bit data coming in time slot 0 (first pixel) and P3 is the 10-bit data coming in time slot 3 (fourth pixel). 16-bit data is packed to 64 bits as P3.P2.P1.P0, where P0-P3 are 16-bit data.

34.3.2.1.2 RGB565 Data

RGB565 data is processed data from the image sensor, which can be put directly into the display buffer. The data is 16 bits wide. The data path is from CSI to memory. It could be directed to the display controller from the memory. On the sensor side, data must be transmitted as P0 first, followed by P1, and so on. For each pixel, whether the MSB or LSB is sent first depends on the endianness of the sensor. Data is 16 bits wide with the MSB labeled RG, and the LSB labeled GB. P0 is represented as RG0 and GB0.

CSI receives data in one of the following sequence:

- RG0, GB0, RG1, GB1, while RG0 comes out at time slot 0 (first data), and GB1 comes out at time slot 3 (last data)
- GB0, RG0, GB1, RG1

Using the first sequence as an example, and assuming the system is running in little endian, the data is presented as:

- 8-bit data from sensor: RG0, GB0, RG1, GB1, …
- 64-bit data before storage in the CSI RX FIFO (PACK_DIR bit = 1): RG0GB0RG1GB1RG2GB2RG3GB3
- 64-bitdata in CSI RX FIFO (SWAP16_EN bit enabled): RG3GB3RG2GB2RG1GB1RG0GB0
- 64-bit transfer to system memory: RG3GB3RG2GB2RG1GB1RG0GB0
- 16-bit read by display controller: RG0GB0, RG1GB1

34.3.2.1.3 RGB888 Data

This is another kind of processed data from image sensor, which can be used for further image processing directly. Each of the data consist of 8-bit Red, 8-bit Green, and 8-bit Blue data. An example of timing scheme is shown in the following figure.

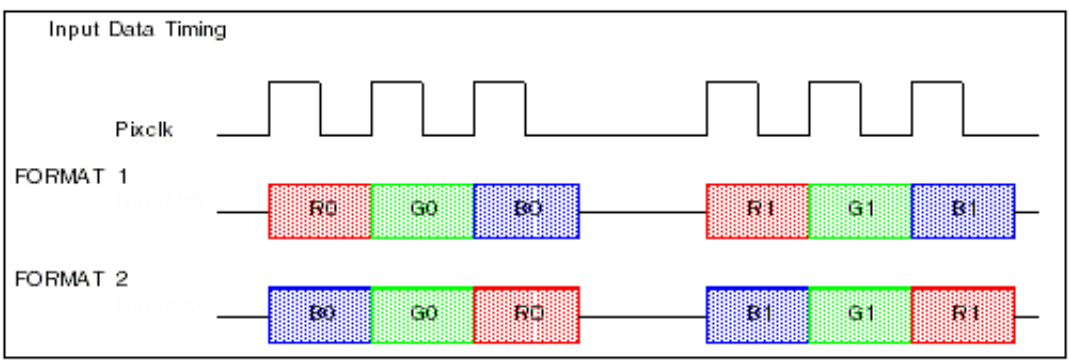


Figure 34-7. Sample Timing Diagram for RGB888 8 bits/cycle Data

An optional scheme to pack a dummy byte is provided. For every group of 3 bytes data, a dummy zero is packed to form a 32-bit word as shown in the following figure. The dummy zero can be packed at the LSB position or MSB position. Using RGB888A_FORMAT_SEL in CSI_CSICR18[18] to determine to put the dummy bytes packed at LSB or MSB position.

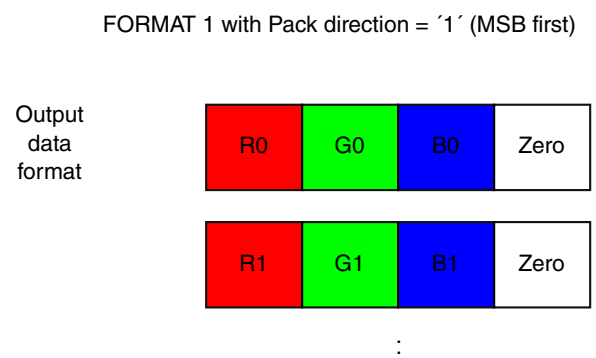


Figure 34-8. Dummy Byte Packing Scheme

34.3.2.2 STAT FIFO Path

Statistics only works for Bayer data in 8-bit per pixel format. It generates 16-bit statistical output from the 8-bit Bayer input (CSI_DATA[13:6]). The outputs are Sum of Green (G), Sum of Red (R), Sum of Blue (B), and Auto Focus (F). Each output is 16-bits wide.

The settings of CSI_CR1[PACK_DIR] and CSI_CR1[SWAP16_EN] bits have no effect on the input path. The CSI_CR1[PACK_DIR] only controls how the 16-bit stat output is packed into the 32-bit STAT FIFO.

When the CSI_CR1[PACK_DIR] = 1, the stat data is packed as:

First 32-bit: RG

Second 32-bit: BF

…

When the CSI_CR1[PACK_DIR] = 0, the stat data is packed as:

First 32-bit: GR

Second 32-bit: FB

...

The diagram shows how 16-bit data stat output is packed into 32-bit STATFIFO register:

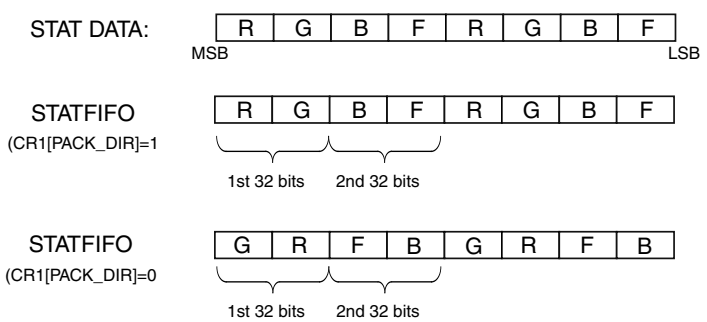


Figure 34-9. Data Packing

34.3.3 Clocks

The following table describes the clock sources for CSI. Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 34-6. CSI Clocks

Clock name	Clock Root	Description
csi_hclk	ipg_clk_root	Module clock
ipg_clk	ipg_clk_root	Peripheral clock
ipg_clk_s	ipg_clk_root	Peripheral access clock
ipg_clk_s_raw	ipg_clk_root	Peripheral raw data clock

34.3.4 Reset

CSI has one hardware reset. When set, it will reset all the logic and registers to their default values.

To re-enable CSI without using the hardware reset, follow the steps below:

1. Disable CSI by setting CSI_CR18[CSI_ENABLE]=0
2. Reflash the DMA controller for RFIFO by setting CSI_CR3[DMA_REFLASH_RFF]=1
3. Enable CSI by setting CSI_CR18[CSI_ENABLE]=1

34.3.5 Interrupt Generation

The information found here describes CSI events that generate interrupts.

34.3.5.1 Start Of Frame Interrupt (SOF_INT)

The source of an SOF interrupt is dependent on the mode of operation.

In traditional mode, VSYNC signal is taken from sensor and SOF_INT is generated at the rising or falling edge (programmable) of VSYNC.

In BT.656 interlace mode, the SOF interrupt information is retrieved from the embedded coding and SOF_INT is generated.

In BT.656 progressive mode, there are two sources of an SOF interrupt:

- In *internal* VSYNC mode, SOF is retrieved from the embedded coding.
- In *external* VSYNC mode, VSYNC is taken from the sensor and SOF is generated at the rising edge of VSYNC.

34.3.5.2 End Of Frame Interrupt (EOF_INT)

An EOF interrupt is generated when the frame ends and the complete frame data in RXFIFO is read.

The EOF event triggering works with the RX count register (CSIRXCNT). Software sets the RX count register to the frame size (in words). The CSI RX logic then counts the number of pixel data being received and compares it with the RX count. If the preset value is reached, an EOF interrupt is generated and the data in the RXFIFO are read. If a SOF event is detected before this happens, the EOF interrupt is not generated.

34.3.5.3 Change Of Field Interrupt (COF_INT)

The Change of Field interrupt is only valid in BT.656 Interlace mode. The COF interrupt is generated when the field toggles, either from field 1 to field 2, or field 2 to field 1.

Software should first check CSI_SR[COF_INT] bit before checking that F1_INT or F2_INT is turned on.

In PAL systems, the field changes at the beginning of the frame and coincides with SOF. For the first field, a COF interrupt is not generated, only an SOF. The COF interrupt is generated for the second field.

34.3.5.4 BT.656 Error Interrupt (ECC_INT)

The BT.656 Error Interrupt is only valid for BT.656 Interlace mode. An ECC interrupt is generated when an error is found on the SAV or EAV codes in the incoming stream. When this happens, the CSI_SR[ECC_INT] status bit is set.

34.3.5.5 RxFIFO Full Interrupt (RxFF_INT)

A RxFIFO full interrupt is generated when the number of data in RXFIFO reaches the water mark defined by RxFF_LEVEL in CSICR3.

34.3.5.6 Statistic FIFO Full Interrupt (STATFF_INT)

A StatFIFO full interrupt is generated when the number of data in STATFIFO reaches the water mark defined by STATFF_LEVEL in CSICR3.

34.3.5.7 RxFIFO Overrun Interrupt (RFF_OR_INT)

A RxFIFO Overrun interrupt is generated when the RxFIFO has 128 words data and more data is being written in.

34.3.5.8 Statistic FIFO Overrun Interrupt (SFF_OR_INT)

A StatFIFO Overrun interrupt is generated when the STATFIFO has 64 words data and more data is being written in.

34.3.5.9 Frame Buffer1 DMA Transfer Done Interrupt (DMA_TSF_DONE_FB1)

A DMA transfer done interrupt of frame buffer1 is generated when one frame of data are transferred from RxFIFO to the frame buffer1 in the external memory.

34.3.5.10 Frame Buffer2 DMA Transfer Done Interrupt (DMA_TSF_DONE_FB2)

A DMA transfer done interrupt of frame buffer2 is generated when one frame of data are transferred from RxFIFO to the frame buffer2 in the external memory.

34.3.5.11 Statistic FIFO DMA Transfer Done Interrupt (DMA_TSF_DONE_SFF)

A StatFIFO DMA transfer done interrupt is generated when all the data are transferred from StatFIFO to the external memory. The transfer size is defined in the STATFIFO DMA Transfer Size Register.

34.3.5.12 AHB Bus Response Error Interrupt (HRESP_ERR_INT)

An AHB Bus response error interrupt is generated when a bus error is detected.

This error can occur if an address is wrong. For more details, refer to AHB protocol.

34.4 External Signals

The table below describes the external signals for the CSI. The external signals are tied between the CSI module and an external CMOS sensor.

Table 34-7. CSI External Signals

Signal	Description	Pad	Mode	Direction
CSI_DATA00	Data Sensor Signal, part of 16-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	GPIO_B1_10	ALT2	I
CSI_DATA01	Data Sensor Signal, part of 16-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	GPIO_B1_11	ALT2	I
CSI_DATA02	Data Sensor Signal, part of 16-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	GPIO_AD_B1_15 pin 27	ALT4	I
		GPIO_AD_B0_11	ALT4	
CSI_DATA03	Data Sensor Signal, part of 16-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	GPIO_AD_B1_14 pin 26	ALT4	I
		GPIO_AD_B0_10	ALT4	
CSI_DATA04	Data Sensor Signal, part of 16-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	GPIO_AD_B1_13 pin 39 (T4.1)	ALT4	I
		GPIO_AD_B0_09	ALT4	
CSI_DATA05	Data Sensor Signal, part of 16-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	GPIO_AD_B1_12 pin 38 (T4.1)	ALT4	I
		GPIO_AD_B0_08	ALT4	
CSI_DATA06	Data Sensor Signal, part of 16-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	GPIO_AD_B1_11 pin 21	ALT4	I
		GPIO_AD_B0_07	ALT4	
CSI_DATA07	Data Sensor Signal, part of 16-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	GPIO_AD_B1_10 pin 20	ALT4	I
		GPIO_AD_B0_06	ALT4	
CSI_DATA08	Data Sensor Signal, part of 16-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	GPIO_AD_B1_09 pin 23	ALT4	I
		GPIO_AD_B0_05	ALT4	
CSI_DATA09	Data Sensor Signal, part of 16-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	GPIO_AD_B1_08 pin 22	ALT4	I
		GPIO_AD_B0_04	ALT4	
CSI_DATA10	Data Sensor Signal, part of 16-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	GPIO_B1_09	ALT2	I
CSI_DATA11	Data Sensor Signal, part of 16-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	GPIO_B1_08	ALT2	I
CSI_DATA12	Data Sensor Signal, part of 16-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	GPIO_B1_07	ALT2	I

Table continues on the next page...

Table 34-7. CSI External Signals (continued)

Signal	Description	Pad	Mode	Direction
CSI_DATA13	Data Sensor Signal, part of 16-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	GPIO_B1_06	ALT2	I
CSI_DATA14	Data Sensor Signal, part of 16-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	GPIO_B1_05	ALT2	I
CSI_DATA15	Data Sensor Signal, part of 16-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	GPIO_B1_04	ALT2	I
CSI_DATA16	Data Sensor Signal, part of 16-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	GPIO_EMC_37 pin 30	ALT4	I
CSI_DATA17	Data Sensor Signal, part of 16-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	GPIO_EMC_36 pin 31	ALT4	I
CSI_DATA18	Data Sensor Signal, part of 16-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	GPIO_EMC_35	ALT4	I
CSI_DATA19	Data Sensor Signal, part of 16-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	GPIO_EMC_34	ALT4	I
CSI_DATA20	Data Sensor Signal, part of 16-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	GPIO_EMC_33	ALT4	I
CSI_DATA21	Data Sensor Signal, part of 16-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	GPIO_EMC_32 pin 28	ALT4	I
CSI_DATA22	Data Sensor Signal, part of 16-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	GPIO_EMC_31 pin 29	ALT4	I
CSI_DATA23	Data Sensor Signal, part of 16-bit Sensor Data Bus (Bayer, YUV, YCrCb, RGB)	GPIO_EMC_30	ALT4	I
CSI_FIELD	CSI Field Signal	GPIO_EMC_38	ALT4	I
CSI_HSYNC	Horizontal Sync (Blank Signal)	GPIO_B1_14	ALT2	I
		GPIO_AD_B1_07 pin 16	ALT4	
		GPIO_AD_B0_15	ALT4	
CSI_MCLK	CMOS Sensor Master Clock *Note: MCLK is provided by the CCM module directly, not from the CSI module itself	GPIO_B1_15	ALT2	O
		GPIO_AD_B1_05 pin 41 (T4.1)	ALT4	
CSI_PIXCLK	Pixel Clock *Note: csi_hclk needs to be 10% faster (or more) than the Pixel Clock	GPIO_B1_12 pin 35 (T4.1)	ALT2	I
		GPIO_AD_B1_04 pin 40 (T4.1)	ALT4	
CSI_VSYNC	Vertical Sync (Start Of Frame)	GPIO_B1_13 pin 34 (T4.1)	ALT2	I
		GPIO_AD_B1_06 pin 17	ALT4	

Table continues on the next page...

Table 34-7. CSI External Signals (continued)

Signal	Description	Pad	Mode	Direction
		GPIO_AD_B0_14	ALT4	

NOTE

DATA[1:0] cannot be used for 8-bit bus.

34.5 Initialization

To initialize CSI, follow the steps below:

1. Enable the external sensor
2. Configure the CSI registers
3. Enable CSI by setting CSI_CR18[CSI_ENABLE]=1

The steps 1 and 2 can be interchanged.

34.6 CSI Memory Map/Register Definition

All the 32-bit registers of the CSI module are summarized in the Memory Map below:

CSI memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
402B_C000	CSI Control Register 1 (CSI_CR1)	32	R/W	4000_0800h	34.6.1/1836
402B_C004	CSI Control Register 2 (CSI_CR2)	32	R/W	0000_0000h	34.6.2/1840
402B_C008	CSI Control Register 3 (CSI_CR3)	32	R/W	0000_0000h	34.6.3/1842
402B_C00C	CSI Statistic FIFO Register (CSI_STATFIFO)	32	R	0000_0000h	34.6.4/1844
402B_C010	CSI RX FIFO Register (CSI_RFIFO)	32	R	0000_0000h	34.6.5/1844
402B_C014	CSI RX Count Register (CSI_RXCNT)	32	R/W	0000_9600h	34.6.6/1845
402B_C018	CSI Status Register (CSI_SR)	32	R/W	8000_4000h	34.6.7/1846
402B_C020	CSI DMA Start Address Register - for STATFIFO (CSI_DMASA_STATFIFO)	32	R/W	0000_0000h	34.6.8/1849
402B_C024	CSI DMA Transfer Size Register - for STATFIFO (CSI_DMATS_STATFIFO)	32	R/W	0000_0000h	34.6.9/1849
402B_C028	CSI DMA Start Address Register - for Frame Buffer1 (CSI_DMASA_FB1)	32	R/W	0000_0000h	34.6.10/ 1850

Table continues on the next page...

CSI memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
402B_C02C	CSI DMA Transfer Size Register - for Frame Buffer2 (CSI_DMASA_FB2)	32	R/W	0000_0000h	34.6.11/ 1851
402B_C030	CSI Frame Buffer Parameter Register (CSI_FBUF_PARA)	32	R/W	0000_0000h	34.6.12/ 1851
402B_C034	CSI Image Parameter Register (CSI_IMAG_PARA)	32	R/W	0000_0000h	34.6.13/ 1852
402B_C048	CSI Control Register 18 (CSI_CR18)	32	R/W	0002_D000h	34.6.14/ 1853
402B_C04C	CSI Control Register 19 (CSI_CR19)	32	R/W	0000_0000h	34.6.15/ 1855

34.6.1 CSI Control Register 1 (CSI_CR1)

This register controls the sensor interface timing and interrupt generation. The interrupt enable bits in this register control the interrupt signals and the status bits. That means status bits will only function when the corresponding interrupt bits are enabled.

Address: 402B_C000h base + 0h offset = 402B_C000h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SWAP16_EN	EXT_VSYNC	EOF_INT_EN	PrP_IF_EN	CCIR_MODE	COF_INT_EN	SF_OR_INTEN	RF_OR_INTEN	Reserved	SFF_DMA_DONE_INTEN	STATFF_INTEN	FB2_DMA_DONE_INTEN	FB1_DMA_DONE_INTEN	RXFF_INTEN	SOF_POL	SOF_INTEN
W																
Reset	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				HSYNC_POL	CCIR_EN	Reserved	FCC	PACK_DIR	CLR_STATFIFO	CLR_RXFIFO	GCLK_MODE	INV_DATA	INV_PCLK	REDGE	PIXEL_BIT
W																
Reset	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0

CSI_CR1 field descriptions

Field	Description
31 SWAP16_EN	SWAP 16-Bit Enable. This bit enables the swapping of 16-bit data. Data is packed from 8-bit or 10-bit to 32-bit first (according to the setting of PACK_DIR) and then swapped as 16-bit words before being put into the RX FIFO. The action of the bit only affects the RX FIFO and has no affect on the STAT FIFO. **NOTE:** Example of swapping enabled: Data input to FIFO = 0x11223344 Data in RX FIFO = 0x 33441122 **NOTE:** Example of swapping disabled: Data input to FIFO = 0x11223344 Data in RX FIFO = 0x11223344 0 Disable swapping 1 Enable swapping
30 EXT_VSYNC	External VSYNC Enable. This bit controls the operational VSYNC mode. **NOTE:** This only works when the CSI is in BT.656 progressive mode. 0 Internal VSYNC mode 1 External VSYNC mode
29 EOF_INT_EN	End-of-Frame Interrupt Enable. This bit enables and disables the EOF interrupt. 0 EOF interrupt is disabled. 1 EOF interrupt is generated when RX count value is reached.
28 PrP_IF_EN	CSI-PrP Interface Enable. This bit controls the CSI to PrP bus. When enabled the RxFIFO is detached from the AHB bus and connected to PrP. All CPU reads or DMA accesses to the RxFIFO register are ignored. All CSI interrupts are also masked. 0 CSI to PrP bus is disabled 1 CSI to PrP bus is enabled
27 CCIR_MODE	BT.656 (formerly CCIR656) Mode Select. This bit controls the BT.656 mode of operation. This bit only works in BT.656 interface mode. 0 Progressive mode is selected 1 Interlace mode is selected
26 COF_INT_EN	Change Of Image Field (COF) Interrupt Enable. This bit enables the COF interrupt. This bit works only in BT.656 interlace mode which is when CCIR_EN = 1 and CCIR_MODE = 1. 0 COF interrupt is disabled 1 COF interrupt is enabled
25 SF_OR_INTEN	STAT FIFO Overrun Interrupt Enable. This bit enables the STATFIFO overrun interrupt. 0 STATFIFO overrun interrupt is disabled 1 STATFIFO overrun interrupt is enabled
24 RF_OR_INTEN	RxFIFO Overrun Interrupt Enable. This bit enables the RX FIFO overrun interrupt. 0 RxFIFO overrun interrupt is disabled 1 RxFIFO overrun interrupt is enabled

Table continues on the next page...

CSI_CR1 field descriptions (continued)

Field	Description
23 Reserved	This field is reserved. This bit should read 0.
22 SFF_DMA_ DONE_INTEN	STATFIFO DMA Transfer Done Interrupt Enable. This bit enables the interrupt of STATFIFO DMA transfer done. 0 STATFIFO DMA Transfer Done interrupt disable 1 STATFIFO DMA Transfer Done interrupt enable
21 STATFF_INTEN	STATFIFO Full Interrupt Enable. This bit enables the STAT FIFO interrupt. 0 STATFIFO full interrupt disable 1 STATFIFO full interrupt enable
20 FB2_DMA_ DONE_INTEN	Frame Buffer2 DMA Transfer Done Interrupt Enable. This bit enables the interrupt of Frame Buffer2 DMA transfer done. 0 Frame Buffer2 DMA Transfer Done interrupt disable 1 Frame Buffer2 DMA Transfer Done interrupt enable
19 FB1_DMA_ DONE_INTEN	Frame Buffer1 DMA Transfer Done Interrupt Enable. This bit enables the interrupt of Frame Buffer1 DMA transfer done. 0 Frame Buffer1 DMA Transfer Done interrupt disable 1 Frame Buffer1 DMA Transfer Done interrupt enable
18 RXFF_INTEN	RxFIFO Full Interrupt Enable. This bit enables the RxFIFO full interrupt. 0 RxFIFO full interrupt disable 1 RxFIFO full interrupt enable
17 SOF_POL	SOF Interrupt Polarity. This bit controls the condition that generates an SOF interrupt. 0 SOF interrupt is generated on SOF falling edge 1 SOF interrupt is generated on SOF rising edge
16 SOF_INTEN	Start Of Frame (SOF) Interrupt Enable. This bit enables the SOF interrupt. 0 SOF interrupt disable 1 SOF interrupt enable
15–12 Reserved	This field is reserved.
11 HSYNC_POL	HSYNC Polarity Select. This bit controls the polarity of HSYNC. This bit only works in gated-clock-that is, GCLK_MODE = 1 and CCIR_EN = 0. 0 HSYNC is active low 1 HSYNC is active high
10 CCIR_EN	BT.656 Interface Enable. This bit selects the type of interface used. 0 Traditional interface is selected. 1 BT.656 interface is selected.
9 Reserved	This field is reserved.
8 FCC	FIFO Clear Control. This bit determines how the RXFIFO and STATFIFO are cleared. When Synchronous FIFO clear is selected the RXFIFO and STATFIFO are cleared, and STAT block is reset, on every SOF. FIFOs and STAT block restarts immediately after reset. For information on the operation when

Table continues on the next page...

CSI_CR1 field descriptions (continued)

Field	Description
	Asynchronous FIFO clear is selected, refer to the descriptions for the CLR_RXFIFO and CLR_STATFIFO bits. 0 Asynchronous FIFO clear is selected. 1 Synchronous FIFO clear is selected.
7 PACK_DIR	Data Packing Direction. This bit Controls how 8-bit/10-bit image data is packed into 32-bit RX FIFO, and how 16-bit statistical data is packed into 32-bit STAT FIFO. 0 Pack from LSB first. For image data, 0x11, 0x22, 0x33, 0x44, it will appear as 0x44332211 in RX FIFO. For stat data, 0xAAAA, 0xBBBB, it will appear as 0xBBBBAAAA in STAT FIFO. 1 Pack from MSB first. For image data, 0x11, 0x22, 0x33, 0x44, it will appear as 0x11223344 in RX FIFO. For stat data, 0xAAAA, 0xBBBB, it will appear as 0xAAAABBBB in STAT FIFO.
6 CLR_STATFIFO	Asynchronous STATFIFO Clear. This bit clears the STATFIFO and Reset STAT block. This bit works only in async FIFO clear mode-that is, FCC = 0. Otherwise this bit is ignored. Writing 1 will clear STATFIFO and reset STAT block immediately, STATFIFO and STAT block then wait and restart after the arrival of next SOF. The bit is restored to 0 automatically after finish. Normally reads 0.
5 CLR_RXFIFO	Asynchronous RXFIFO Clear. This bit clears the RXFIFO. This bit works only in async FIFO clear mode-that is, FCC = 0. Otherwise this bit is ignored. Writing 1 clears the RXFIFO immediately, RXFIFO restarts immediately after that. The bit is restored to 0 automatically after finish. Normally reads 0.
4 GCLK_MODE	Gated Clock Mode Enable. Controls if CSI is working in gated or non-gated mode. This bit works only in traditional mode-that is, CCIR_EN = 0. Otherwise this bit is ignored. 0 Non-gated clock mode. All incoming pixel clocks are valid. HSYNC is ignored. 1 Gated clock mode. Pixel clock signal is valid only when HSYNC is active.
3 INV_DATA	Invert Data Input. This bit enables or disables internal inverters on the data lines. 0 CSI_D[7:0] data lines are directly applied to internal circuitry 1 CSI_D[7:0] data lines are inverted before applied to internal circuitry
2 INV_PCLK	Invert Pixel Clock Input. This bit determines if the Pixel Clock (CSI_PIXCLK) is inverted before it is applied to the CSI module. 0 CSI_PIXCLK is directly applied to internal circuitry 1 CSI_PIXCLK is inverted before applied to internal circuitry
1 REDGE	Valid Pixel Clock Edge Select. Selects which edge of the CSI_PIXCLK is used to latch the pixel data. 0 Pixel data is latched at the falling edge of CSI_PIXCLK 1 Pixel data is latched at the rising edge of CSI_PIXCLK
0 PIXEL_BIT	Pixel Bit. This bit indicates the bayer data width for each pixel. This bit should be configured before activating or re-starting the embedded DMA controller. 0 8-bit data for each pixel 1 10-bit data for each pixel

34.6.2 CSI Control Register 2 (CSI_CR2)

This register provides the statistic block with data about which live view (sensor input) resolution is being used, and the starting sensor pixel of the Bayer pattern. It also contains the horizontal and vertical count used to determine the number of pixels to skip between the 64 x 64 blocks of statistics when generating statistics on live view image that are greater than 512 x 384.

Address: 402B_C000h base + 4h offset = 402B_C004h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DMA_BURST_TYPE_RFF		DMA_BURST_TYPE_SFF		Reserved	DRM	AFS		SCE	Reserved		BTS		LVRM		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	VSC								HSC							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CSI_CR2 field descriptions

Field	Description
31–30 DMA_BURST_TYPE_RFF	Burst Type of DMA Transfer from RxFIFO. Selects the burst type of DMA transfer from RxFIFO. X0 INCR8 01 INCR4 11 INCR16
29–28 DMA_BURST_TYPE_SFF	Burst Type of DMA Transfer from STATFIFO. Selects the burst type of DMA transfer from STATFIFO. X0 INCR8 01 INCR4 11 INCR16

Table continues on the next page...

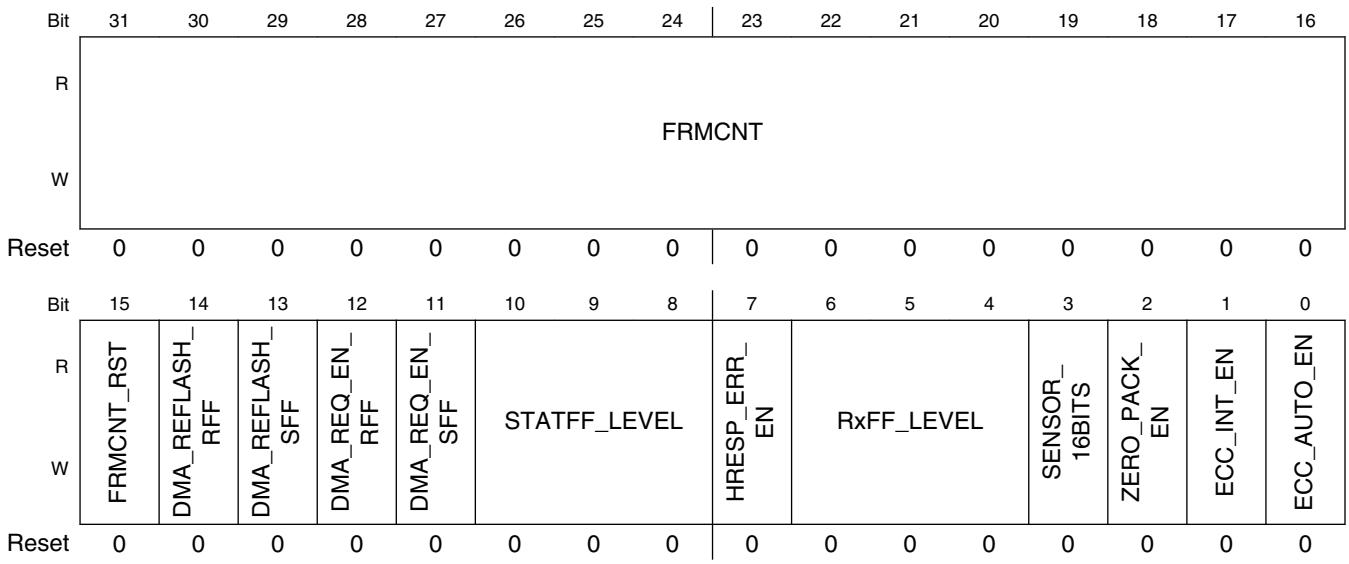
CSI_CR2 field descriptions (continued)

Field	Description
27 Reserved	This field is reserved. This bit should read 0
26 DRM	Double Resolution Mode. Controls size of statistics grid. 0 Stats grid of 8 x 6 1 Stats grid of 8 x 12
25–24 AFS	Auto Focus Spread. Selects which green pixels are used for auto-focus. 00 Abs Diff on consecutive green pixels 01 Abs Diff on every third green pixels 1x Abs Diff on every four green pixels
23 SCE	Skip Count Enable. Skip count is the number of pixels to skip between the 64x64 blocks of statistics. 0 Skip count disable 1 Skip count enable
22–21 Reserved	This field is reserved. These bits should read 0.
20–19 BTS	Bayer Tile Start. Controls the Bayer pattern starting point. 00 GR 01 RG 10 BG 11 GB
18–16 LVRM	Live View Resolution Mode. Selects the grid size used for live view resolution. 0 512 x 384 1 448 x 336 2 384 x 288 3 384 x 256 4 320 x 240 5 288 x 216 6 400 x 300
15–8 VSC	Vertical Skip Count. Contains the number of rows to skip. SCE must be 1, otherwise VSC is ignored. 0-255 Number of rows to skip minus 1
HSC	Horizontal Skip Count. Contains the number of pixels to skip. SCE must be 1, otherwise HSC is ignored. 0-255 Number of pixels to skip minus 1

34.6.3 CSI Control Register 3 (CSI_CR3)

This read/write register acts as an extension of the functionality of the CSI Control register 1, adding additional control and features.

Address: 402B_C000h base + 8h offset = 402B_C008h



CSI_CR3 field descriptions

Field	Description
31–16 FRMCNT	Frame Counter. This is a 16-bit Frame Counter (Wraps around automatically after reaching the maximum)
15 FRMCNT_RST	Frame Count Reset. Resets the Frame Counter. (Cleared automatically after reset is done) 0 Do not reset 1 Reset frame counter immediately
14 DMA_REFLASH_RFF	Reflash DMA Controller for RxFIFO. This bit reflash the embedded DMA controller for RxFIFO. It should be reflashed before the embedded DMA controller starts to work. (Cleared automatically after reflashing is done) 0 No reflashing 1 Reflash the embedded DMA controller
13 DMA_REFLASH_SFF	Reflash DMA Controller for STATFIFO. This bit reflash the embedded DMA controller for STATFIFO. It should be reflashed before the embedded DMA controller starts to work. (Cleared automatically after reflashing is done) 0 No reflashing 1 Reflash the embedded DMA controller

Table continues on the next page...

CSI_CR3 field descriptions (continued)

Field	Description
12 DMA_REQ_EN_RFF	DMA Request Enable for RxFIFO. This bit enables the dma request from RxFIFO to the embedded DMA controller. 0 Disable the dma request 1 Enable the dma request
11 DMA_REQ_EN_SFF	DMA Request Enable for STATFIFO. This bit enables the dma request from STATFIFO to the embedded DMA controller. 0 Disable the dma request 1 Enable the dma request
10–8 STATFF_LEVEL	STATFIFO Full Level. When the number of data in STATFIFO reach this level, STATFIFO full interrupt is generated, or STATFIFO DMA request is sent. 000 4 Double words 001 8 Double words 010 12 Double words 011 16 Double words 100 24 Double words 101 32 Double words 110 48 Double words 111 64 Double words
7 HRESP_ERR_EN	Hresponse Error Enable. This bit enables the hresponse (AHB protocol standard) error interrupt. 0 Disable hresponse error interrupt 1 Enable hresponse error interrupt
6–4 RxFF_LEVEL	**RxFIFO Full Level**. When the number of data in RxFIFO reaches this level, a RxFIFO full interrupt is generated, or an RXFIFO DMA request is sent. If the number of data is smaller, it will send request to bus with small burst size. 000 4 Double words 001 8 Double words 010 16 Double words 011 24 Double words 100 32 Double words 101 48 Double words 110 64 Double words 111 96 Double words
3 SENSOR_16BITS	16-bit Sensor Mode. This bit indicates one 16-bit sensor connected to the 16-bit data ports. This bit should be set if there is one 16-bit sensor connected. This bit should be configured before re-flashing or restarting the embedded DMA controller. 0 Only one 8-bit sensor is connected. 1 One 16-bit sensor is connected.
2 ZERO_PACK_EN	Dummy Zero Packing Enable. This bit causes a dummy zero to be packed with every 3 incoming bytes, forming a 32-bit word. The dummy zero is always packed to the LSB position. This packing function is only available in 8-bit/pixel mode. 0 Zero packing disabled 1 Zero packing enabled

Table continues on the next page...

CSI_CR3 field descriptions (continued)

Field	Description
1 ECC_INT_EN	Error Detection Interrupt Enable. This bit enables and disables the error detection interrupt. This feature only works in BT.656 interlace mode. 0 No interrupt is generated when error is detected. Only the status bit ECC_INT is set. 1 Interrupt is generated when error is detected.
0 ECC_AUTO_EN	Automatic Error Correction Enable. This bit enables and disables the automatic error correction. If an error occurs and error correction is disabled only the ECC_INT status bit is set. This feature only works in BT. 656 interlace mode. 0 Auto Error correction is disabled. 1 Auto Error correction is enabled.

34.6.4 CSI Statistic FIFO Register (CSI_STATFIFO)

The StatFIFO is a read-only register containing statistic data from the sensor. Writing to this register has no effect.

Address: 402B_C000h base + Ch offset = 402B_C00Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	STAT																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CSI_STATFIFO field descriptions

Field	Description
STAT	Static data from sensor

34.6.5 CSI RX FIFO Register (CSI_RFIFO)

This read-only register contains received image data. Writing to this register has no effect.

Address: 402B_C000h base + 10h offset = 402B_C010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IMAGE																															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CSI_RFIFO field descriptions

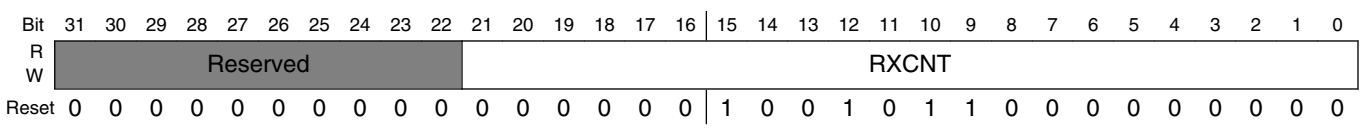
Field	Description
IMAGE	Received image data

34.6.6 CSI RX Count Register (CSI_RXCNT)

This register works for End of Frame (EOF) interrupt generation while width/height is used for DMA done. It should be set to the number of words to receive that would generate an EOF interrupt.

There is an internal counter that counts the number of words read from the RX FIFO. Whenever the RX FIFO is being read, by either the CPU or the embedded DMA controller, the counter value is updated and compared with this register. If the values match, then an EOF interrupt is triggered.

Address: 402B_C000h base + 14h offset = 402B_C014h



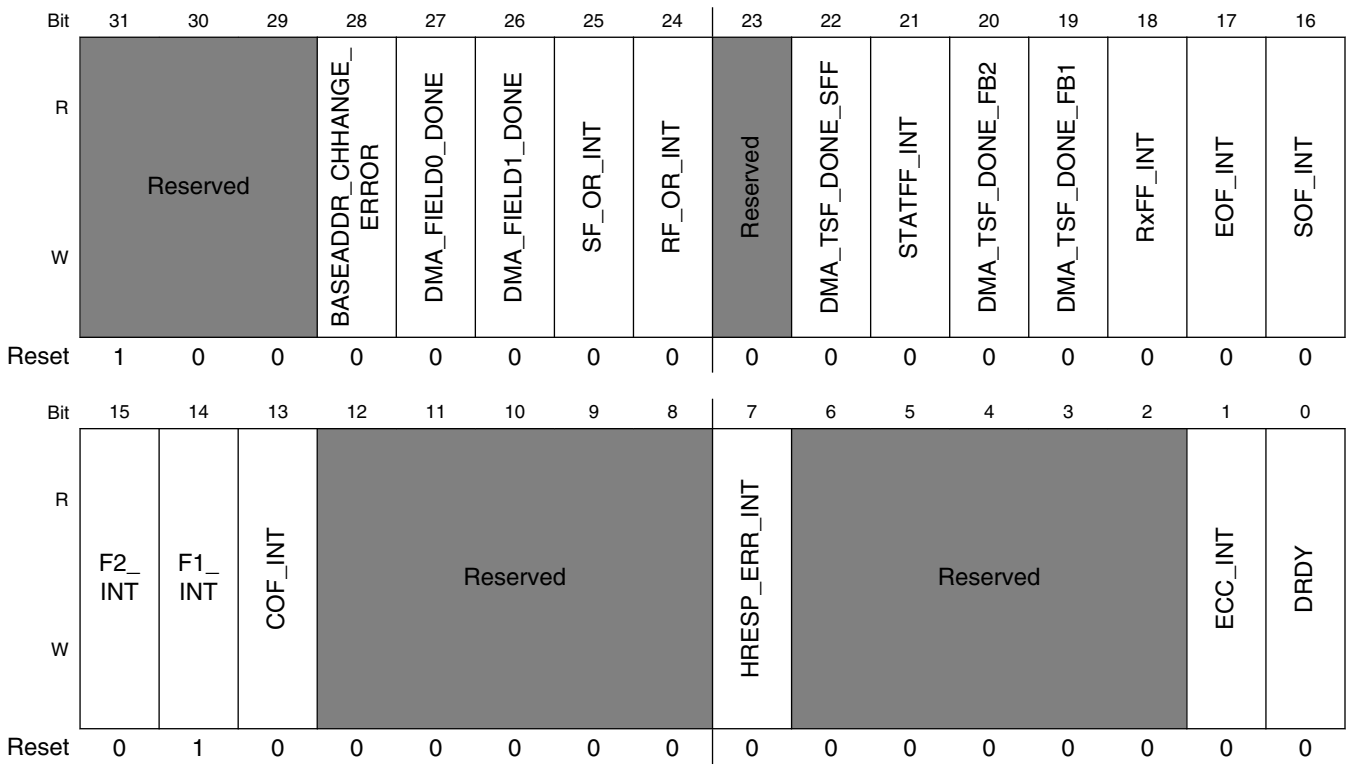
CSI_RXCNT field descriptions

Field	Description
31–22 Reserved	This field is reserved. These bits should read 0.
RXCNT	RxFIFO Count. This 22-bit counter for RXFIFO is updated each time the RXFIFO is read by CPU or DMA.This counter should be set to the expected number of words to receive that would generate an EOF interrupt.

34.6.7 CSI Status Register (CSI_SR)

This read/write register shows sensor interface status, and which kind of interrupt is being generated. The corresponding interrupt bits must be set for the status bit to function. Status bits will function normally even if the corresponding interrupt enable bits are not enabled.

Address: 402B_C000h base + 18h offset = 402B_C018h



CSI_SR field descriptions

Field	Description
31–29 Reserved	This field is reserved.
28 BASEADDR_CHHANGE_ERROR	When using base address switching enable, this bit will be 1 when switching occur before DMA complete. This bit will be clear by writing 1. When this interrupt happens, follow the steps listed below. 1. Unassert the CSI enable, CSIx_CSICR18 bit31, 2. Reflash the DMA, assert the CSIX_CSICR3 bit 14, 3. Assert the CSI enable, CSIx_CSICR18 bit31.
27 DMA_FIELD0_DONE	When DMA field 0 is complete, this bit will be set to 1(clear by writing 1).

Table continues on the next page...

CSI_SR field descriptions (continued)

Field	Description
26 DMA_FIELD1_DONE	When DMA field 1 is complete, this bit will be set to 1(clear by writing 1).
25 SF_OR_INT	STATFIFO Overrun Interrupt Status. Indicates the overflow status of the STATFIFO register. (Cleared by writing 1) 0 STATFIFO has not overflowed. 1 STATFIFO has overflowed.
24 RF_OR_INT	RxFIFO Overrun Interrupt Status. Indicates the overflow status of the RxFIFO register. (Cleared by writing 1) 0 RXFIFO has not overflowed. 1 RXFIFO has overflowed.
23 Reserved	This field is reserved. This bit should read 0.
22 DMA_TSF_DONE_SFF	DMA Transfer Done from StatFIFO. Indicates that the dma transfer from StatFIFO is completed. It can trigger an interrupt if the corresponding enable bit is set in CSICR1. This bit can be cleared by writting 1 or reflashing the StatFIFO dma controller in CSICR3.(Cleared by writing 1) 0 DMA transfer is not completed. 1 DMA transfer is completed.
21 STATFF_INT	STATFIFO Full Interrupt Status. Indicates the number of data in the STATFIFO reaches the trigger level. (this bit is cleared automatically by reading the STATFIFO) 0 STATFIFO is not full. 1 STATFIFO is full.
20 DMA_TSF_DONE_FB2	DMA Transfer Done in Frame Buffer2. Indicates that the DMA transfer from RxFIFO to Frame Buffer2 is completed. It can trigger an interrupt if the corresponding enable bit is set in CSICR1. This bit can be cleared by by writting 1 or reflashing the RxFIFO dma controller in CSICR3. (Cleared by writing 1) 0 DMA transfer is not completed. 1 DMA transfer is completed.
19 DMA_TSF_DONE_FB1	DMA Transfer Done in Frame Buffer1. Indicates that the DMA transfer from RxFIFO to Frame Buffer1 is completed. It can trigger an interrupt if the corresponding enable bit is set in CSICR1. This bit can be cleared by by writting 1 or reflashing the RxFIFO dma controller in CSICR3. (Cleared by writing 1) 0 DMA transfer is not completed. 1 DMA transfer is completed.
18 RxFF_INT	RXFIFO Full Interrupt Status. Indicates the number of data in the RxFIFO reaches the trigger level. (this bit is cleared automatically by reading the RxFIFO) 0 RxFIFO is not full. 1 RxFIFO is full.
17 EOF_INT	End of Frame (EOF) Interrupt Status. Indicates when EOF is detected. (Cleared by writing 1) 0 EOF is not detected. 1 EOF is detected.
16 SOF_INT	Start of Frame Interrupt Status. Indicates when SOF is detected. (Cleared by writing 1)

Table continues on the next page...

CSI_SR field descriptions (continued)

Field	Description
	0 SOF is not detected. 1 SOF is detected.
15 F2_INT	BT.656 Field 2 Interrupt Status. Indicates the presence of field 2 of video in BT.656 mode. (Cleared automatically when current field does not match) **NOTE:** Only works in BT.656 Interlace mode. 0 Field 2 of video is not detected 1 Field 2 of video is about to start
14 F1_INT	BT.656 Field 1 Interrupt Status. Indicates the presence of field 1 of video in BT.656 mode. (Cleared automatically when current field does not match) **NOTE:** Only works in BT.656 Interlace mode. 0 Field 1 of video is not detected. 1 Field 1 of video is about to start.
13 COF_INT	Change Of Field Interrupt Status. Indicates that a change of the video field has been detected. Only works in BT.656 Interlace mode. Software should read this bit first and then dispatch the new field from F1_INT and F2_INT. (Cleared by writing 1) 0 Video field has no change. 1 Change of video field is detected.
12–8 Reserved	This field is reserved. These bits should read 0.
7 HRESP_ERR_INT	Hresponse Error Interrupt Status. Indicates that a hresponse (AHB protocol standard) error has been detected. (Cleared by writing 1) 0 No hresponse error. 1 Hresponse error is detected.
6–2 Reserved	This field is reserved. These bits should read 0.
1 ECC_INT	BT.656 Error Interrupt. This bit indicates an error has occurred. This only works in BT.656 Interlace mode. (Cleared by writing 1) 0 No error detected 1 Error is detected in BT.656 coding
0 DRDY	RXFIFO Data Ready. Indicates the presence of data that is ready for transfer in the RxFIFO. (Cleared automatically by reading FIFO) 0 No data (word) is ready 1 At least 1 datum (word) is ready in RXFIFO.

34.6.8 CSI DMA Start Address Register - for STATFIFO (CSI_DMASA_STATFIFO)

This register provides the start address for the embedded DMA controller of STATFIFO. The embedded DMA controller will read data from STATFIFO and write it to the external memory from the start address. This register should be configured before activating or restarting the embedded DMA controller.

Address: 402B_C000h base + 20h offset = 402B_C020h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DMA_START_ADDR_SFF															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DMA_START_ADDR_SFF														Reserved	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CSI_DMASA_STATFIFO field descriptions

Field	Description
31–2 DMA_START_ADDR_SFF	DMA Start Address for STATFIFO. Indicates the start address to write data. The embedded DMA controller will read data from STATFIFO and write it from this address through AHB bus. **NOTE:** The address should be double words aligned.
Reserved	This field is reserved. These bits should read 0.

34.6.9 CSI DMA Transfer Size Register - for STATFIFO (CSI_DMATS_STATFIFO)

This register provides the total transfer size for the embedded DMA controller of STATFIFO. This register should be configured before activating or restarting the embedded DMA controller.

Address: 402B_C000h base + 24h offset = 402B_C024h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DMA_TSF_SIZE_SFF																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CSI_DMATS_STATFIFO field descriptions

Field	Description
DMA_TSF_SIZE_SFF	DMA Transfer Size for STATFIFO. Indicates how many words to be transfered by the embedded DMA controller. **NOTE:** The size should be double words aligned.

34.6.10 CSI DMA Start Address Register - for Frame Buffer1 (CSI_DMASA_FB1)

This register provides the start address in the frame buffer1 for the embedded DMA controller of RxFIFO. The embedded DMA controller will read data from RxFIFO and write it to the frame buffer1 from the start address. This register should be configured before activating or restarting the embedded DMA controller.

Address: 402B_C000h base + 28h offset = 402B_C028h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DMA_START_ADDR_FB1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DMA_START_ADDR_FB1														Reserved	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CSI_DMASA_FB1 field descriptions

Field	Description
31–2 DMA_START_ADDR_FB1	DMA Start Address in Frame Buffer1. Indicates the start address to write data. The embedded DMA controller will read data from RxFIFO and write it from this address through AHB bus. **NOTE:** The address should be double words aligned.
Reserved	This field is reserved. These bits should read 0.

34.6.11 CSI DMA Transfer Size Register - for Frame Buffer2 (CSI_DMASA_FB2)

This register provides the start address in the frame buffer2 for the embedded DMA controller of RxFIFO. The embedded DMA controller will read data from RxFIFO and write it to the frame buffer2 from the start address. This register should be configured before activating or restarting the embedded DMA controller.

Address: 402B_C000h base + 2Ch offset = 402B_C02Ch

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	DMA_START_ADDR_FB2															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DMA_START_ADDR_FB2														Reserved	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CSI_DMASA_FB2 field descriptions

Field	Description
31–2 DMA_START_ADDR_FB2	DMA Start Address in Frame Buffer2. Indicates the start address to write data. The embedded DMA controller will read data from RxFIFO and write it from this address through AHB bus. **NOTE:** The address should be double words aligned.
Reserved	This field is reserved. These bits should read 0.

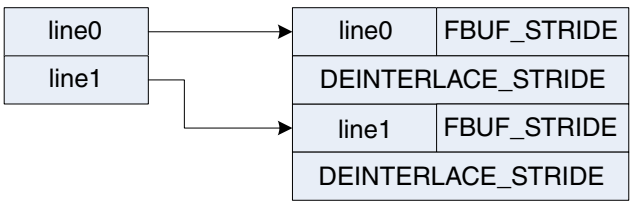
34.6.12 CSI Frame Buffer Parameter Register (CSI_FBUF_PARA)

This register provides the stride of the frame buffer to show how many words to skip before starting to write the next row of the image. The width of the frame buffer minus the width of the image is the stride. This register should be configured before activating or restarting the embedded DMA controller.

Address: 402B_C000h base + 30h offset = 402B_C030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DEINTERLACE_STRIDE																FBUF_STRIDE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CSI_FBUF_PARA field descriptions

Field	Description
31–16 DEINTERLACE_ STRIDE	DEINTERLACE_STRIDE is only used in the deinterlace mode. If line stride feature is supported in deinterlace mode, FBUF_STRIDE and DEINTERLACE_STRIDE need to be configured at the same time. DEINTERLACE_STRIDE is configured the same as line width. 
FBUF_STRIDE	Frame Buffer Parameter. Indicates the stride of the frame buffer. The width of the frame buffer(in double words) minus the width of the image(in double words) is the stride. The stride should be double words aligned. The embedded DMA controller will skip the stride before starting to write the next row of the image. **NOTE:** Double words is equivalent to 8 bytes.

34.6.13 CSI Image Parameter Register (CSI_IMAG_PARA)

This register provides the width and the height of the image from the sensor. The width and height should be aligned in pixel. The width of the image multiplied by the height is the total pixel size that will be transferred in a frame by the embedded DMA controller. This register should be configured before activating or restarting the embedded DMA controller.

Address: 402B_C000h base + 34h offset = 402B_C034h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IMAGE_WIDTH																IMAGE_HEIGHT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

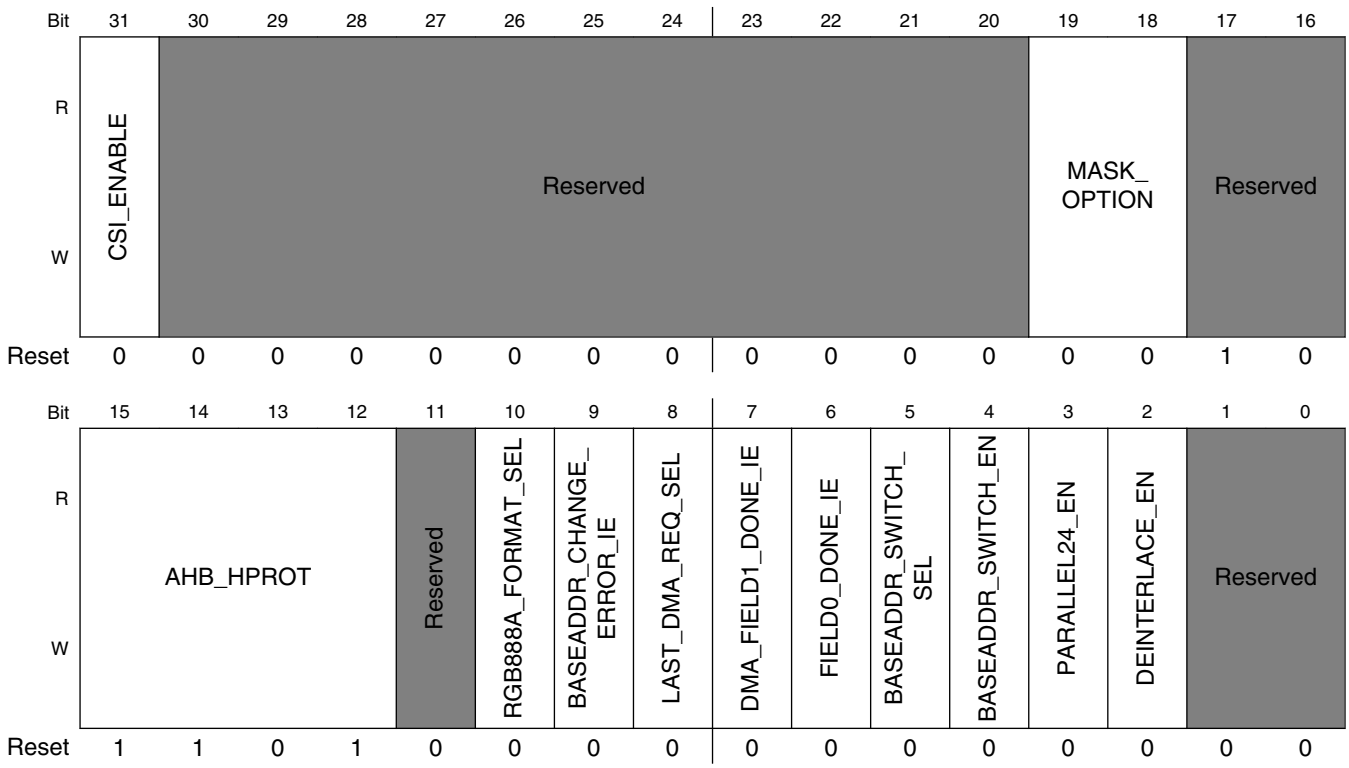
CSI_IMAG_PARA field descriptions

Field	Description
31–16 IMAGE_WIDTH	This field indicates the number of active pixel cycles per line. For 1 cycle/pixel, it equals to IMAGE_WIDTH, and for 2 cycles/pixel it equals to (IMAGE_WIDTH *2). For 24bpp pixel transferred on 8-bit data bus (3 cycle/pixel), it equals to image width. If the input data from the sensor is 8-bit/pixel format, the IMAGE_WIDTH should be a multiple of 8 pixels. If the input data from the sensor is 10-bit/pixel or 16-bit/pixel format, the IMAGE_WIDTH should be a multiple of 4 pixels.
IMAGE_HEIGHT	Image Height. Indicates how many pixels in a column of the image from the sensor.

34.6.14 CSI Control Register 18 (CSI_CR18)

This read/write register acts as an extension of the functionality of the CSI Control register 18

Address: 402B_C000h base + 48h offset = 402B_C048h



CSI_CR18 field descriptions

Field	Description
31 CSI_ENABLE	CSI global enable signal. Only when this bit is 1, CSI can start to receive the data and store to memory.
30–20 Reserved	This field is reserved.
19–18 MASK_OPTION	These bits used to choose the method to mask the CSI input. 00 Writing to memory (OCRAM or external DDR) from first completely frame, when using this option, the CSI_ENABLE should be 1. 01 Writing to memory when CSI_ENABLE is 1. 02 Writing to memory from second completely frame, when using this option, the CSI_ENABLE should be 1. 03 Writing to memory when data comes in, not matter the CSI_ENABLE is 1 or 0.
17–16 Reserved	This field is reserved.

Table continues on the next page...

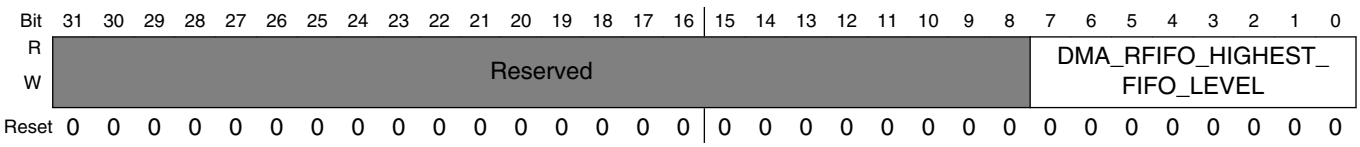
CSI_CR18 field descriptions (continued)

Field	Description
15–12 AHB_HPROT	Hprot value in AHB bus protocol.
11 Reserved	This field is reserved.
10 RGB888A_ FORMAT_SEL	Output is 32-bit format. 0 {8'h0, data[23:0]} 1 {data[23:0], 8'h0}
9 BASEADDR_ CHANGE_ ERROR_IE	Base address change error interrupt enable signal. 0 Interrupt disabled 1 Interrupt enabled
8 LAST_DMA_ REQ_SEL	Choosing the last DMA request condition. If fifo_full_level > hburst_length, please select hburst_length. Otherwise, select fifo_full_length. 0 fifo_full_level 1 hburst_length
7 DMA_FIELD1_ DONE_IE	When in interlace mode, field 1 done interrupt enable. 0 Interrupt disabled 1 Interrupt enabled
6 FIELD0_DONE_ IE	In interlace mode, field 0 means interrupt enabled. 0 Interrupt disabled 1 Interrupt enabled
5 BASEADDR_ SWITCH_SEL	CSI 2 base addresses switching method. When using this bit, BASEADDR_SWITCH_EN is 1. 0 Switching base address at the edge of the vsync 1 Switching base address at the edge of the first data of each frame
4 BASEADDR_ SWITCH_EN	When this bit is enabled, CSI DMA will switch the base address according to BASEADDR_SWITCH_SEL rather than automatically by DMA completed.
3 PARALLEL24_ EN	Enable bit for Parallel RGB888/YUV444 24bit input 0 Input is disabled 1 Input is enabled
2 DEINTERLACE_ EN	This bit is used to select the output method When input is standard BT.656 video. 0 Deinterlace disabled 1 Deinterlace enabled
Reserved	This field is reserved.

34.6.15 CSI Control Register 19 (CSI_CR19)

This read/write register acts as an extension of the functionality of the CSI Control register 19

Address: 402B_C000h base + 4Ch offset = 402B_C04Ch



CSI_CR19 field descriptions

Field	Description
31–8 Reserved	This field is reserved.
DMA_RFIFO_ HIGHEST_FIFO_ LEVEL	This byte stores the highest FIFO level achieved by CSI FIFO timely and will be clear by writing 8'ff to it.

Chapter 35
Enhanced LCD Interface (eLCDIF)

35.1 Chip-specific eLCDIF information

Table 35-1. Reference links to related information

Topic	Related module(s)	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
Interrupts, DMA Events and XBAR Assignments	-	Interrupts, DMA Events and XBAR Assignments

35.2 Overview

The enhanced Liquid Crystal Display Interface (eLCDIF) is a general purpose display controller

The eLCDIF block supports the following:

- Displays that support moving pictures and require the RGB interface mode (DOTCLK interface).

35.2.1 Block Diagram

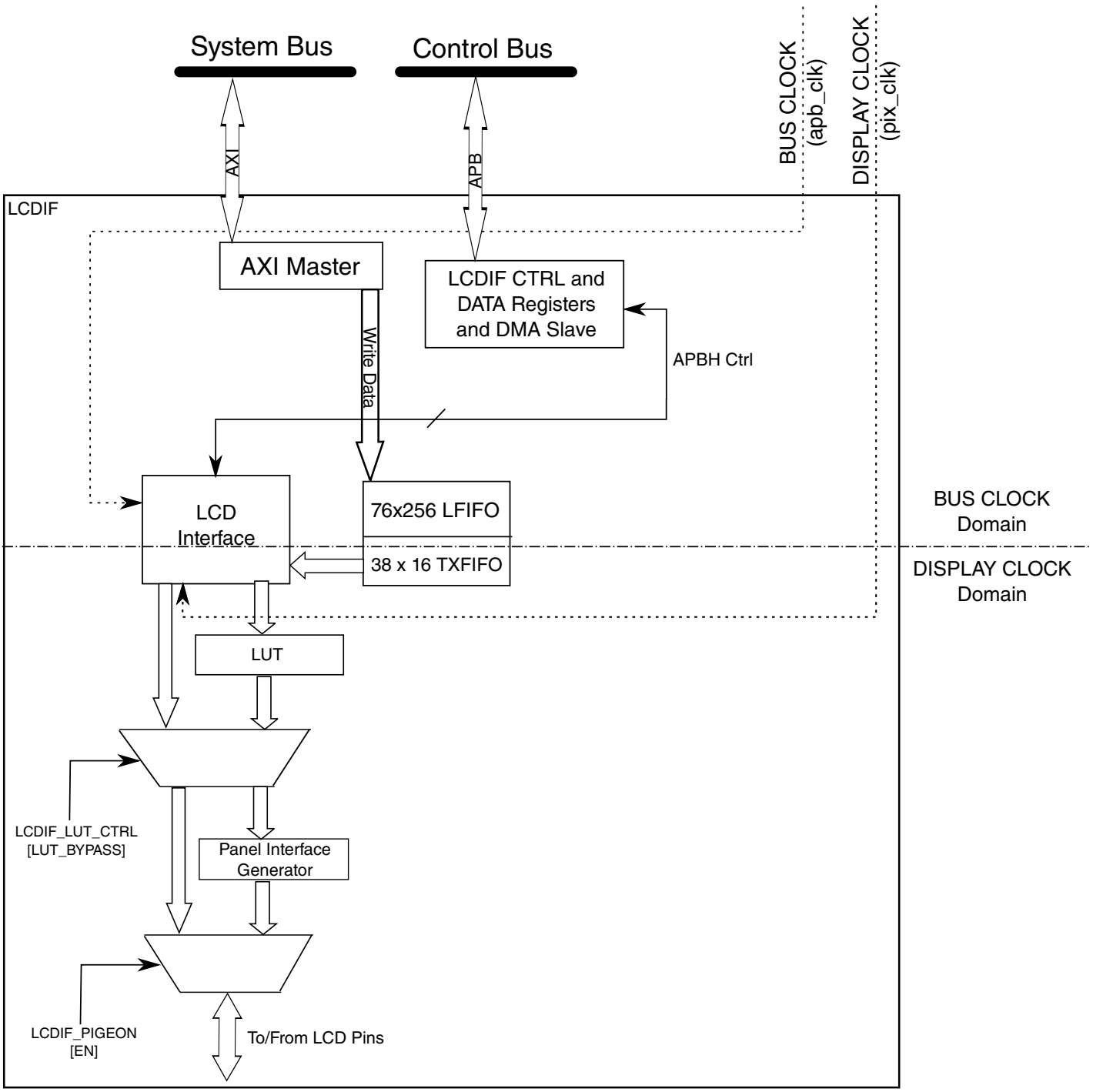


Figure 35-1. Top-Level Block Diagram of eLCDIF subsystem

35.2.2 Features

The eLCDIF provides fully programmable functionality to supported interfaces:

- Bus master interface to source frame buffer data for display refresh. This interface can also be used to drive data for "Smart" displays.
- DMA interface to manage data transfers between "Smart" displays and SoC.
- 8/16/18/24 bit LCD data bus support available depending on I/O mux options.
- Programmable timing and parameters for DOTCLK LCD interfaces.

35.3 Functional Description

The sections listed below describe the pipeline for the eLCDIF interface:

- Bus Interface Mechanisms
- Write Data Path
- Initialization

The differences for each modes are described in the sections listed below:

- DOTCLK Interface

LCDIF pin usage by interface mode is described in eLCDIF Pin Usage by Interface Mode.

35.3.1 Bus Interface Mechanisms

The eLCDIF module has memory-mapped control, data and status registers. It provides several interfaces to transfer data between the display and SoC.

The bus master interface is used to initiate the requests to transfer data from external memory to the display. It is completely autonomous, or no CPU intervention is required, to manage the cyclical nature of refreshing standard display types.

The DMA interface requires the SoC integrated APBDMA engine to manage data transfers on behalf of the eLCDIF. APBDMA descriptor chains are created to manage control and data flow between the eLCDIF and external display. This interface is typically used for "Smart" display types that integrate their own frame buffer and control interface. The APBDMA engine is used to reduce the CPU compute requirements to complete the display solution.

The host CPU executes display drivers to manage the display solution. The following sections describe the system bus interface mechanisms.

35.3.1.1 Bus Master Operation in Write/Display Modes

The eLCDIF block has a bus master interface that initiates requests for data to drive the display. The LCDIF_CTRL[MASTER] bit must be set to 1 to enable the bus master interface. Software should program all control registers required to transfer the frame sequence.

The DOTCLK mode is used to refresh the display at the desired refresh rate and resolution, and drive displays that don't integrate a display buffer memory. When the display is refreshed, the eLCDIF will automatically update the LCDIF_CUR_BUF register with the value in LCDIF_NEXT_BUF at the end of current frame and start fetching the next frame from the new address. If the LCDIF_NEXT_BUF register was not updated within a frame refresh cycle, eLCDIF will keep transmitting the last frame until a new value is programmed into that register.

35.3.1.2 System Bus Master Performance

The performance of the eLCDIF block can be controlled by changing the burst length and the outstanding cycle issuing capability depending on the memory bandwidth requirements. Two fields in the LCDIF_CTRL2 register will throttle system memory requests. The LCDIF_CTRL2[OUTSTANDING_REQS] bitfield will control how many requests the eLCDIF can have in flight on any given clock cycle. This should be programmed based on the expected system bus latency for returned read data. Also, the LCDIF_CTRL2[BURST_LEN_8] bit will set the number of 64 bit words requested for each eLCDIF system bus request to either 8 or 16 QWORDS. Generally, 4 outstanding requests of length 16 will provide enough performance to drive any standard display resolution. These configuration bits are intended to change the access pattern of the eLCDIF to optimize system bus throughput when other system masters will contend for system memory resources.

35.3.1.3 DMA Operation in MPU Read Mode

Data can be transferred between an external display and the SoC using the APBDMA interface. This mode is enabled by setting the LCDIF_CTRL[MASTER] bit to 0 and is typically used with "Smart" displays. The APBDMA module executes DMA descriptor chains to process the desired data flow sequence. The data can be either frame buffer data or display control data.

35.3.2 Write Data Path

eLCDIF supports raster based frame buffers and there is no support for tiled buffers.

There are several options to accommodate endianness of display buffers in memory before the data is processed for the external display. The LCDIF_CTRL[INPUT_DATA_SWIZZLE] field provides the following options for data word multiplexing:

```
00 (0): No swizzle (little-endian)
01 (1): Swap bytes 0 and 3, swap bytes 1 and 2 (big-endian)
10 (2): Swap half-words
11 (3): Swap bytes within each half-word
```

The LCDIF_CTRL[WORD_LENGTH] field indicates the input data/pixel format. LCDIF_TRANSFER_COUNT register denotes how much data is contained in each frame. The LCDIF_TRANSFER_COUNT[H_COUNT] field indicates the number of pixels per line and LCDIF_TRANSFER_COUNT[V_COUNT] indicates the total number of lines per frame. The LCDIF_CTRL1[BYTE_PACKING_FORMAT] field can be used to specify which bytes within the 32-bit word are going to be valid. For example, if the entire 32-bit word is valid, LCDIF_CTRL1[BYTE_PACKING_FORMAT] should be set to 0xF, if only lower 3 bytes of each word in the frame buffer are valid, then LCDIF_CTRL1[BYTE_PACKING_FORMAT] should be set to 0x7.

The LCDIF_CTRL[LCD_DATABUS_WIDTH] field suggests the width of the bus going to the display controller. There is an option to source all 32 bits of the input word and transfer it to the output I/O display interface. If the LCDIF_CTRL[LCD_DATABUS_WIDTH] is not the same as LCDIF_CTRL[WORD_LENGTH], eLCDIF will perform RGB to RGB color space conversion. For example, if the input frame has fewer bits per pixel than the display, as in a 16 bpp input frame going to 24 bpp LCD, eLCDIF will pad the MSBs of each color to the LSBs of the same color for each pixel. If the input frame has more bits per pixel than the display, for example, 24 bpp input frame going to 16 bpp LCD, eLCDIF will drop the LSBs of each color channel to convert to the lower color depth. eLCDIF also has the capability to support delta pixel displays by swizzling the R, G and B colors of each pixel in the odd and even lines of the frame separately by programming the LCDIF_CTRL2[ODD_LINE_PATTERN] and the LCDIF_CTRL2[EVEN_LINE_PATTERN] bit fields. This operation occurs after the RGB-to-RGB color space conversion operation.

The following list shows how the different input/output combinations can be obtained:

- LCDIF_CTRL[WORD_LENGTH]=1 indicates that the input is 8-bit data. Any combination of LCDIF_CTRL1[BYTE_PACKING_FORMAT] is permissible.

Limitation: LCDIF_TRANSFER_COUNT[H_COUNT] must be a multiple of the sum of BYTE_PACKING_FORMAT [3], BYTE_PACKING_FORMAT [2], BYTE_PACKING_FORMAT [1] and BYTE_PACKING_FORMAT [0]. LCDIF_CTRL[LCD_DATABUS_WIDTH] must be 1, indicating an 8-bit data bus.

- LCDIF_CTRL[WORD_LENGTH]=0 implies the input frame buffer is RGB 16 bits per pixel. LCDIF_CTRL[DATA_FORMAT_16_BIT] field determines the pixels are RGB 555 or RGB 565.

  Limitation: LCDIF_CTRL1[BYTE_PACKING_FORMAT] should be 0x3 or 0xC if there is only one pixel per word. If there are two pixels per word, it should be 0xF and LCDIF_TRANSFER_COUNT[H_COUNT] will be restricted to be a multiple of 2 pixels.

- LCDIF_CTRL[WORD_LENGTH]=2 indicates that input frame buffer is RGB 18 bits per pixel, that is, RGB 666. The valid RGB values can be left-aligned or right-aligned within a 32-bit word. The alignment of the valid 18 bits within a word is indicated by the LCDIF_CTRL[DATA_FORMAT_18_BIT] bit.

  Limitation: LCDIF_CTRL1[BYTE_PACKING_FORMAT] can be 0x7, 0xE or 0xF. Packed pixels are not supported in this case.
  LCDIF_TRANSFER_COUNT[H_COUNT] can be any number.

- LCDIF_CTRL[WORD_LENGTH]=3 indicates that the input frame-buffer is RGB 24 bits per pixel (RGB 888). If LCDIF_CTRL1[BYTE_PACKING_FORMAT] is 0x7, it indicates that there is only one pixel per 32-bit word and there is no restriction on LCDIF_TRANSFER_COUNT[H_COUNT]. This is also the option that provides 32 bit output depending on the I/O muxing options available. The fourth byte, or bits [31:24], and connected to the I/Os if this muxing is available in the chip package.

  Limitation: If LCDIF_CTRL1[BYTE_PACKING_FORMAT] is 0xF, it indicates that the pixels are packed, that is, there are 4 pixels in 3 words or 12 bytes and LCDIF_TRANSFER_COUNT[H_COUNT] must be a multiple of 4 pixels.

After the RGB to RGB color space conversions, there is one more opportunity to swizzle the data before sending it out to the display or the encoder. This can be done with the LCDIF_CTRL[CSC_DATA_SWIZZLE] field, and it provides the same options as the LCDIF_CTRL[INPUT_DATA_SWIZZLE] register.

Finally, there is an option to shift the output data before sending it out to the display. This is done based on the LCDIF_CTRL[SHIFT_DIR] and LCDIF_CTRL[SHIFT_NUM_BITS] fields.

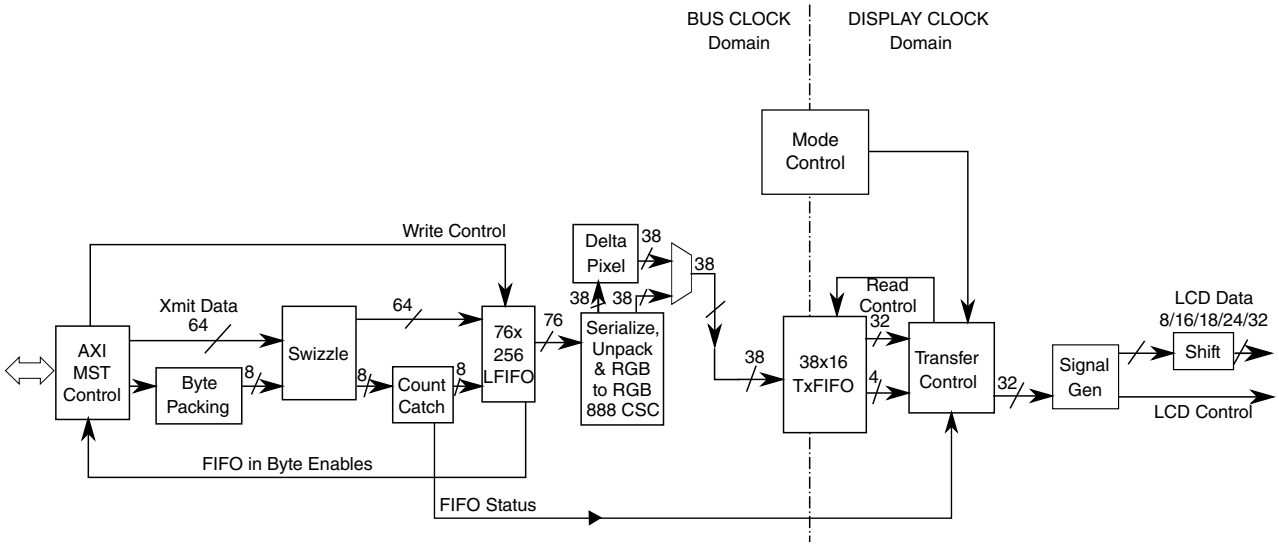


Figure 35-2. General Operations in Write Data Path

The examples in the following figures illustrate some different combinations of register programming for write mode. Assume that the data transferred over the system bus within a 32 bit word is organized as {A7-A0, B7-B0, C7-C0, D7-D0} in 8-bit mode and {A15-A0, B15-B0} in 16-bit mode.

In this example, all 32 bits of the input word are transferred out over an 8 bit display bus. Each byte within the 32 bit word is shifted to the right with zeros appended to bits D[7:6]. The input data bits [7:2] are shifted to the right by 2 bits and presented on the D[5:0].

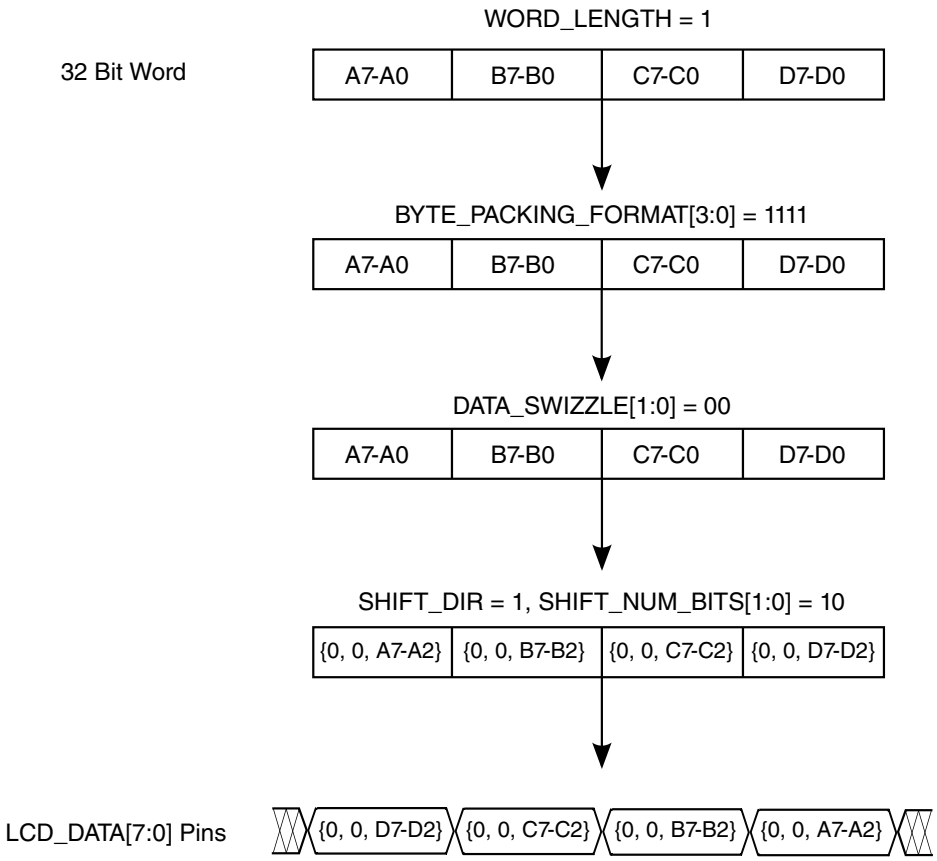


Figure 35-3. Register programming for write mode

In this 8 bit display interface example, one byte of the input word is deleted and not transferred over the external 8 bit display interface. This mode could be used to transfer 24bpp pixels over the 8 bit interface. In this case, the 4th unused byte is not transferred.

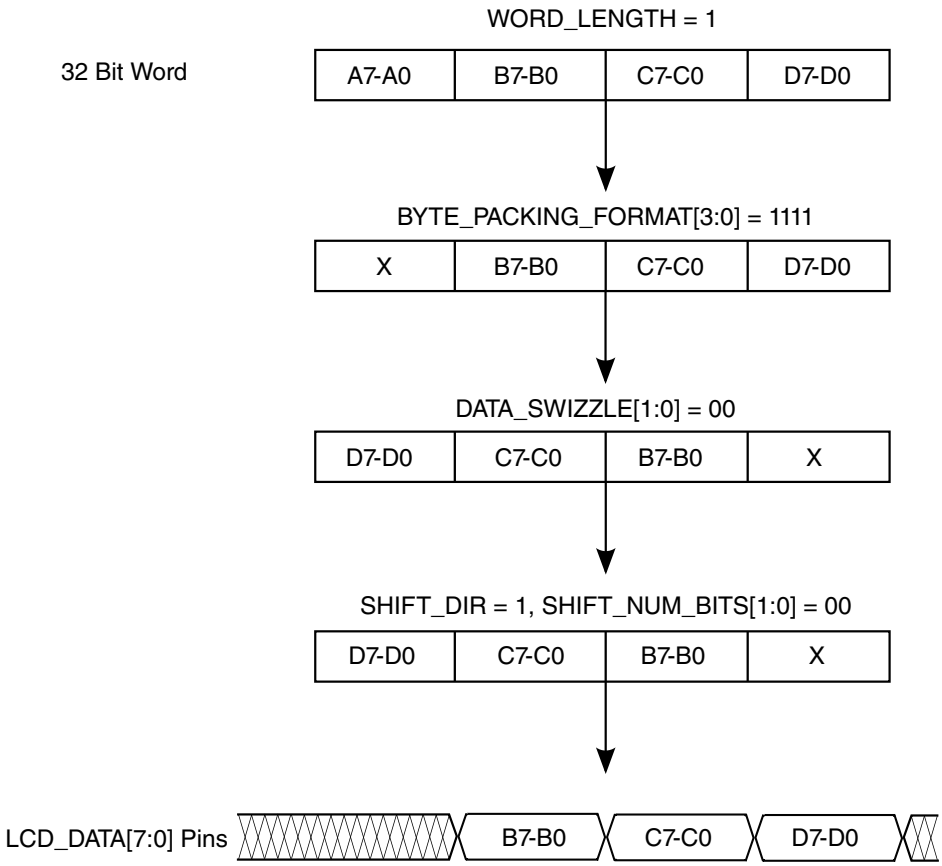


Figure 35-4. Register programming for write mode

The following example uses a 16 bit display interface. Each 16 bit half word is shifted to the right by two bits with zeros appended to the most significant two bits.

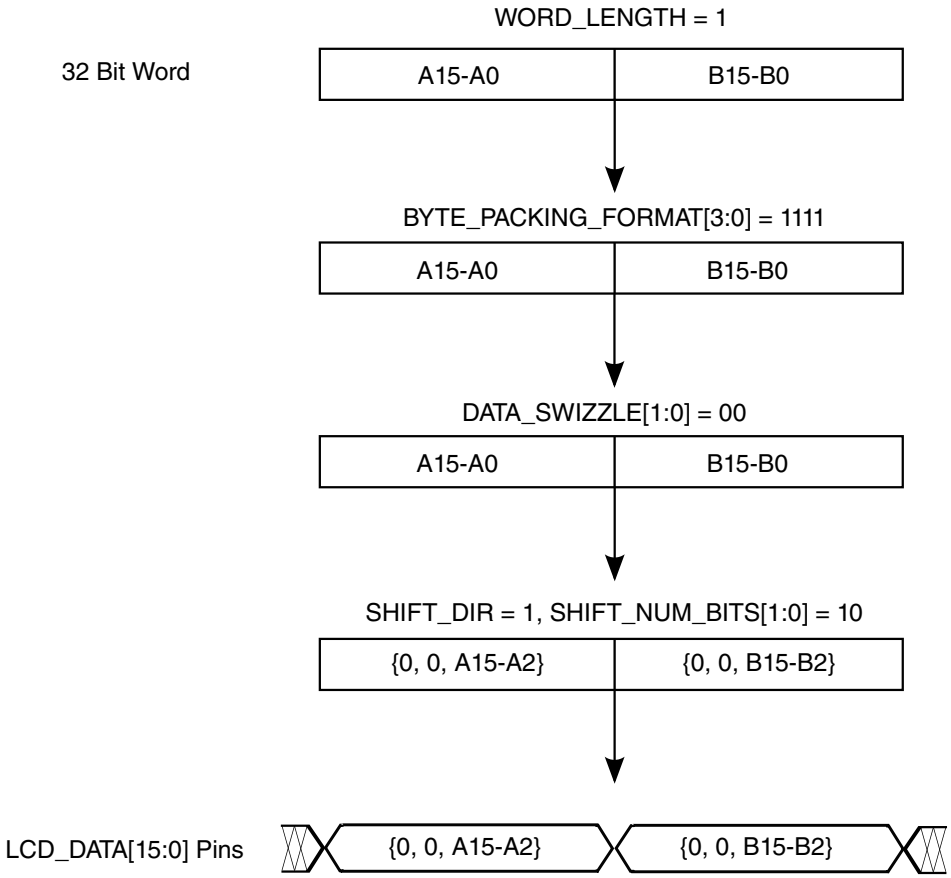


Figure 35-5. Register programming for write mode

This example indicates how an unpacked frame buffer can sourced for display. Only a single 16 bit half word within the 32 bit word is tranferred out via the 16 display bus.

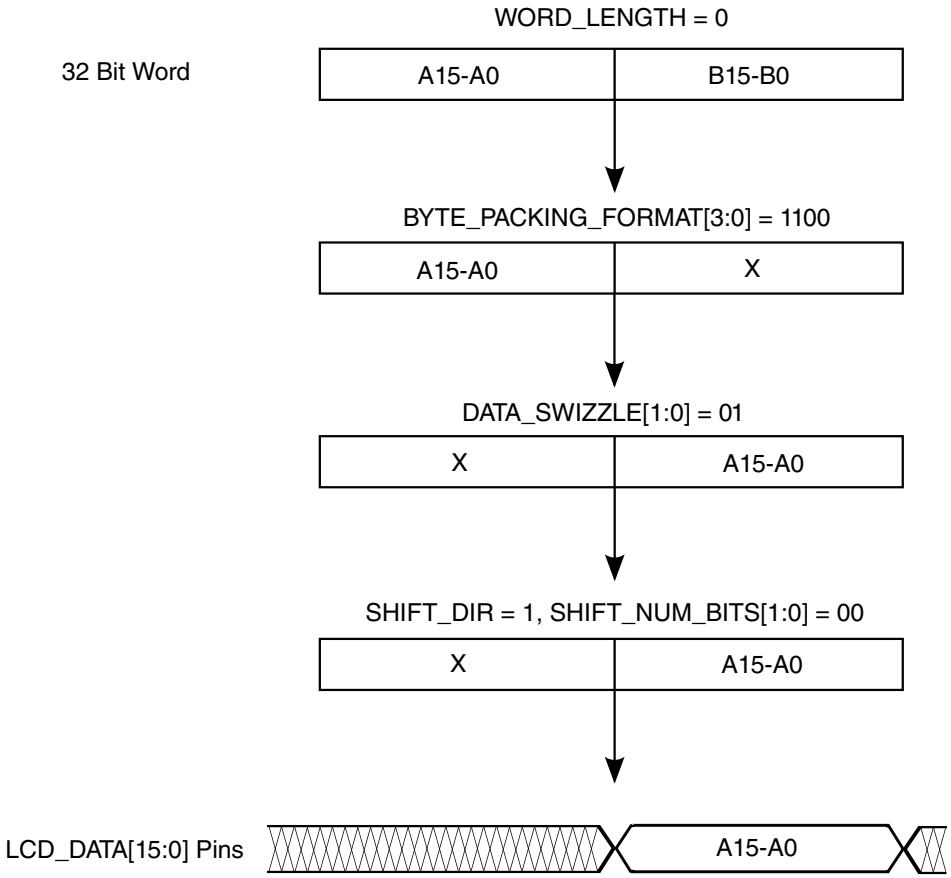


Figure 35-6. Register programming for write mode

35.3.3 DOTCLK Interface

The DOTCLK interface is another mode used in moving picture displays.

It includes the VSYNC, HSYNC, DOTCLK and (optional) ENABLE signals. The interface is popularly called the RGB interface if the ENABLE signal is present.

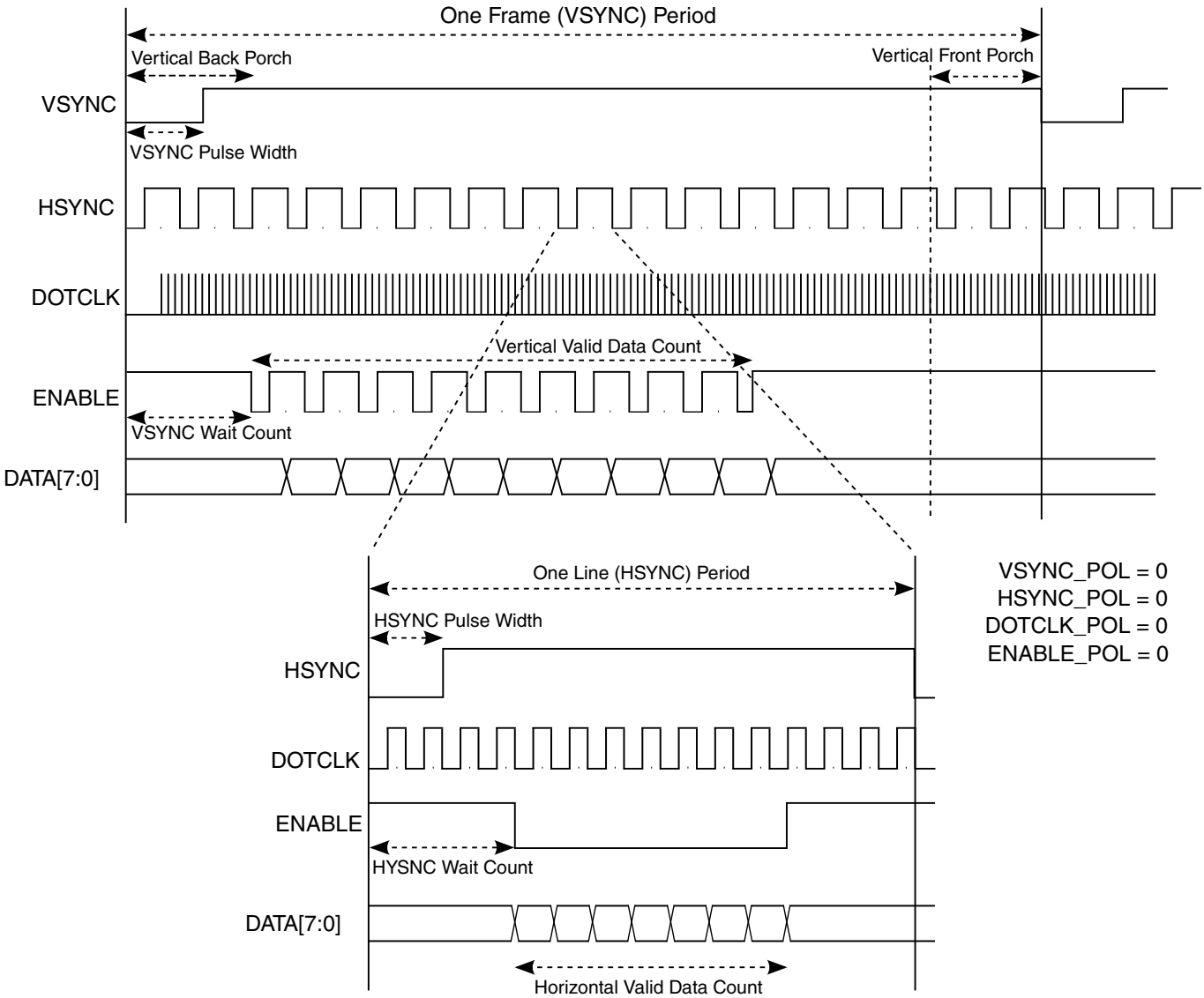


Figure 35-7. DOTCLK protocol with programmable parameters

The DOTCLK mode writes data at high speed to the LCD, and the display operation is synchronized with the VSYNC, HSYNC, ENABLE and DOTCLK signals. The polarities, periods and pulse-widths of the sync signals are programmable using the LCDIF_VDCTRL0-4 registers. The units for the VSYNC signal must be number of horizontal lines and can be selected using the VSYNC_PULSE_WIDTH_UNIT and VSYNC_PERIOD_UNIT bit fields. The VERTICAL_WAIT_CNT is by default given the same unit as the VSYNC_PERIOD. The DISPLAY CLOCK (pix_clk) frequency is managed by the CCM.

In DOTCLK mode, LCDIF_CTRL_BYPASS_COUNT bit must be set to 1. To end the current transfer, the software should make the DOTCLK_MODE bit 0, so that all data that is currently in the LCDIF LFIFO and TXFIFO is transmitted. Once that transfer is complete, the block will automatically clear the RUN bit and issue the cur_frame_done interrupt.

Refer to Write Data Path for more details.

35.3.3.1 Code Example

The following code shows an example for programming a 320x240 display.

NOTE

Setting up the display must be done via SPI.

```
// Note: Common initialization steps in Initialization must also be
// executed along with the following code
BF_CS1 (LCDIF_CTRL, DOTCLK_MODE, 1);
BF_CS1 (LCDIF_CTRL, BYPASS_COUNT, 1); //Always for DOTCLK mode
BF_CS1 (LCDIF_VDCTRL0, VSYNC_OEB, 0); //Vsync is always an output in the DOTCLK mode
BF_CS4 (LCDIF_VDCTRL0, VSYNC_POL, 0, HSYNC_POL, 0, DOTCLK_POL, 0, ENABLE_POL, 0);
BF_CS1 (LCDIF_VDCTRL0, ENABLE_PRESENT, 1);
BF_CS2 (LCDIF_VDCTRL0, VSYNC_PERIOD_UNIT, 1, VSYNC_PULSE_WIDTH_UNIT, 1);
BF_CS1 (LCDIF_VDCTRL0, VSYNC_PULSE_WIDTH, 2);
BF_CS1 (LCDIF_VDCTRL1, VSYNC_PERIOD, 280);
BF_CS2 (LCDIF_VDCTRL2, HSYNC_PULSE_WIDTH, 10, HSYNC_PERIOD, 360); //Assuming
                                                    //  LCD_DATABUS_WIDTH is 24bit
BF_CS2 (LCDIF_VDCTRL3, VSYNC_ONLY, 0);
BF_CS2 (LCDIF_VDCTRL3, HORIZONTAL_WAIT_CNT, 20, VERTICAL_WAIT_CNT, 20);
BF_CS1 (LCDIF_VDCTRL4, DOTCLK_H_VALID_DATA_CNT, 320);//Note that DOTCLK_V_VALID_DATA_CNT is
                                //implicitly assumed to be HW_LCDIF_TRANSFER_COUNT_V_COUNT
BF_CS1 (LCDIF_VDCTRL4, SYNC_SIGNALS_ON, 1);
BF_CS1 (LCDIF_CTRL, RUN, 1);
```

To stop the transfer completely, the ideal way is to make DOTCLK_MODE = 0. In that case, the block will transmit the contents in the FIFO and reset the RUN bit.

35.3.4 LUT

The Lookup Table (LUT) is used to expand the 8 bits pixel to 24 bits pixel before output to external displayer. This module is used when input data from memory is 8-bits in width under limited bandwidth system and external displayer has a 24 bits requirement.

There are two 256x24 bits LUT memory in eLCDIF module in case of changing LUT during active display. When using LUT function, it should be initialized through single PIO register writing first. Writing data to register LCDIF_LUT0_DATA will trigger storing data to lut0 memory and increase the lut0 memory address LCDIF_LUT0_ADDR

automatically. Writing data again to register LCDIF_LUT0_DATA will store the data to memory and increase the memory address accordingly. The same operation is performed while writing to LCDIF_LUT1_DATA and LCDIF_LUT1_ADDR.

After initialization, LCDIF_CUR_BUF_ADDR bit 0 determines which LUT is used during active display.

Lastly, when there is active display, it will use the lowest 8 bits eLCDIF data as LUT address and output the 24 bits from the lookup table.

35.3.5 Panel Interface Generator (Pigeon Mode)

There are several panel interface signal outputs (CLK signal not included), each of them with dedicated timing purpose as default. This is called "Legacy Mode". Pigeon Mode is a timing mode which can be independently enabled on any of the timing signals, with a unified flexible configuration. Signals within pigeon mode can be programmed into any supported signals, and are interchangeable. The following are the legacy timing signals which support pigeon mode:

- PIGEON[00] – LCD_DATA16
- PIGEON[01] – LCD_DATA17
- PIGEON[02] – LCD_DATA18
- PIGEON[03] – LCD_DATA19
- PIGEON[04] – LCD_DATA20
- PIGEON[05] – LCD_DATA21
- PIGEON[06] – LCD_DATA22
- PIGEON[07] – LCD_DATA23
- PIGEON[08] – LCD_ENABLE
- PIGEON[09] – LCD_HSYNC
- PIGEON[10] – LCD_VSYNC
- PIGEON[11] – LCD_CLK

Each pigeon signal has one local counter with a configurable start point and incremental condition. It will be compared to configuration register value for signal assertion/de-assertion control, plus delta offset for data alignment and other options like polarity/logic operation. A detailed running scenario is as follows:

1. Start local counter on the MASK rising edge (reference point/start point)
2. Increment on event selected through INC_SEL
3. Count and match SET_CNT: assert signal, reset counter (SET_CNT==0 means assert immediately on MASK's rising edge)
4. Count and match CLR_CNT: de-assert signal, stop counter (CLR_CNT==0 means de-assert on MASK's falling edge)

NOTE

When local counter is running, further changes to MASK are not cared unless CLR_CNT==0

MASK is the start point for local counter created using a combination any of the options (ANDed) from below:

1. STATE_MASK= (FS|FB|FD|FE) AND (LS|LB|LD|LE)

   In this case you have 8 bits to select in which vertical/horizontal state your counter starts ticking. This is the most common use-case because timing signals generally relate to scan states.

   For example, for a line timing signal start on Line Begin phase during Frame Begin/ Frame Data lines, use the configuration below:

Table 35-2. State Mask Combinations

STATE_MASK	LS	LB=1	LD	LE
FS				
FB=1		X		
FD=1		X		
FE				

2. MASK_CNT/MASK_CNT_SEL (global counter)

   Sometimes a more accurate reference point is required, such as "line 20 in a frame", or "line 12 in Frame Begin state" or "pixel 23 in LD phase". For such use-cases, several Global Counters shared by all pigeons are provided. Global counter type (line counter/frame counter/state counter, etc.) is selected using MASK_CNT_SEL, when global counter matches the MASK_CNT value. The pigeon local counter will start ticking.

3. Use another pigeon signal as mask (SIG_LOGIC=MASK)

   For some tightly coupled signals it is possible to use one as a reference to generate another.

   INC_SEL- select local counter tick event
   a. pclk - pixel clk
   b. line - line start pulse
   c. frame - frame start pulse
   d. another - another pigeon signal

OFFSET- offset on pclk basis: Some signals need to come out slightly earlier or later than programmed. For example, the CE type signal usually aligns with the Line Data phase, but some panels need it as one cycle pulse before Line Data. The user can set OFFSET to a negative value to achieve this.

Global counters (selectable through MASK_CNT_SEL):

- HCNT / VCNT : normal pclk counter / line counter
- HSTATE_CNT / VSTATE_CNT: similar to above, but reset when state changes
- HSTATE_CYCLE / VSTATE_CYCLE: (see figure below for definition of CYCLE/ PERIOD/CNT)

  Some panels have multiple Gate Drivers/Source Drivers, so Frame Data / Line Data state may be further split to match each driver, and signals such as CE[n] are only valid during part n of Line Data. For such signals, use LCDIF_PIGEON_CTRL[*_PERIOD] to specify PERIOD where CNT is reset and CYCLE is included. Then the user can select CYCLE as MASK_CNT to generate mask for CE[n].

- FRAME_CNT / FRAME_CYCLE (only for frame-crossing signals): frame cycle counter doesn't have a reset condition; use LCDIF_PIGEON_CTRL1[FRAME_CNT_CYCLES] to reset it.

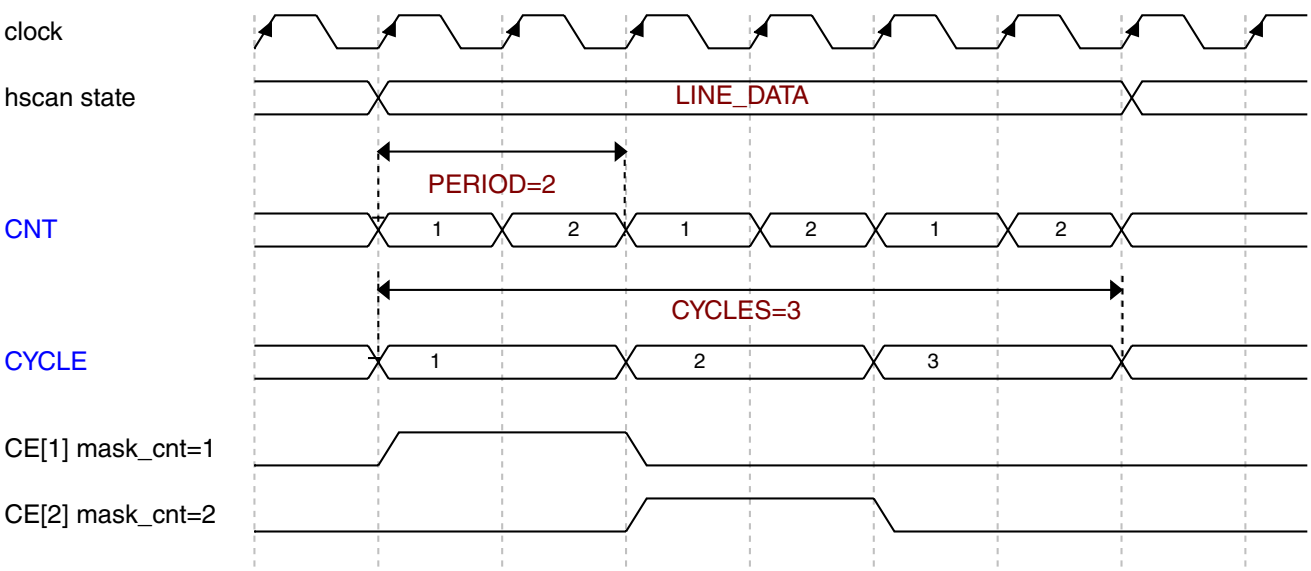


Figure 35-8. Definition of CNT, CYCLE, PERIOD

The following are register settings for the figure above:

- LCDIF_PIGEON_CTRL0[LD_PERIOD]=2
- MASK_CNT_SEL = 1 // HSTATE_CYCLE

- MASK_CNT = 1 // CE[1]
- MASK_CNT = 2 // CE[2]

35.3.6 eLCDIF Pin Usage by Interface Mode

The following tables detail how the eLCDIF level interface pins are used based on the desired mode of operation. The chip level I/Os should also be configured to be consistent with the desired eLCDIF operating mode.

NOTE

There is an option to internally mux the HSYNC, DOTCLK and ENABLE signals in the DOTCLK mode by setting the LCDIF_VDCTRL3[MUX_SYNC_SIGNALS] bit.

Table 35-3. Pin use in DOTCLK Mode

PIN NAME	8-bit DOTCLK LCD IF	16-bit DOTCLK LCD IF	18-bit DOTCLK LCD IF	24-bit DOTCLK LCD IF
LCD_VSYNC	LCD_VSYNC	LCD_VSYNC	LCD_VSYNC	LCD_VSYNC
LCD_HSYNC	LCD_HSYNC	LCD_HSYNC	LCD_HSYNC	LCD_HSYNC
LCD_DOTCLK	LCD_DOTCLK	LCD_DOTCLK	LCD_DOTCLK	LCD_DOTCLK
LCD_ENABLE	LCD_ENABLE	LCD_ENABLE	LCD_ENABLE	LCD_ENABLE
LCD_DATA23 (LCD_D23)	X	X	X	LCD_DATA23
LCD_DATA22 (LCD_D22)	X	X	X	LCD_DATA22
LCD_DATA21 (LCD_D21)	X	X	X	LCD_DATA21
LCD_DATA20 (LCD_D20)	X	X	X	LCD_DATA20
LCD_DATA19 (LCD_D19)	X	X	X	LCD_DATA19
LCD_DATA18 (LCD_D18)	X	X	X	LCD_DATA18
LCD_DATA17 (LCD_D17)	X	X	LCD_DATA17	LCD_DATA17
LCD_DATA16 (LCD_D16)	X	X	LCD_DATA16	LCD_DATA16
LCD_DATA15 (LCD_D15)	X	LCD_DATA15	LCD_DATA15	LCD_DATA15
LCD_DATA14 (LCD_D14)	X	LCD_DATA14	LCD_DATA14	LCD_DATA14
LCD_DATA13 (LCD_D13)	X	LCD_DATA13	LCD_DATA13	LCD_DATA13

Table continues on the next page...

Table 35-3. Pin use in DOTCLK Mode (continued)

PIN NAME	8-bit DOTCLK LCD IF	16-bit DOTCLK LCD IF	18-bit DOTCLK LCD IF	24-bit DOTCLK LCD IF
LCD_DATA12 (LCD_D12)	X	LCD_DATA12	LCD_DATA12	LCD_DATA12
LCD_DATA11 (LCD_D11)	X	LCD_DATA11	LCD_DATA11	LCD_DATA11
LCD_DATA10 (LCD_D10)	X	LCD_DATA10	LCD_DATA10	LCD_DATA10
LCD_DATA09 (LCD_D9)	X	LCD_DATA09	LCD_DATA09	LCD_DATA09
LCD_DATA08 (LCD_D8)	X	LCD_DATA08	LCD_DATA08	LCD_DATA08
LCD_DATA07 (LCD_D7)	LCD_DATA07	LCD_DATA07	LCD_DATA07	LCD_DATA07
LCD_DATA06 (LCD_D6)	LCD_DATA06	LCD_DATA06	LCD_DATA06	LCD_DATA06
LCD_DATA05 (LCD_D5)	LCD_DATA05	LCD_DATA05	LCD_DATA05	LCD_DATA05
LCD_DATA04 (LCD_D4)	LCD_DATA04	LCD_DATA04	LCD_DATA04	LCD_DATA04
LCD_DATA03 (LCD_D3)	LCD_DATA03	LCD_DATA03	LCD_DATA03	LCD_DATA03
LCD_DATA02 (LCD_D2)	LCD_DATA02	LCD_DATA02	LCD_DATA02	LCD_DATA02
LCD_DATA01 (LCD_D1)	LCD_DATA01	LCD_DATA01	LCD_DATA01	LCD_DATA01
LCD_DATA00 (LCD_D0)	LCD_DATA00	LCD_DATA00	LCD_DATA00	LCD_DATA00
LCD_RESET	LCD_RESET	LCD_RESET	LCD_RESET	LCD_RESET

35.3.7 CRC Polynomial

The eLCDIF uses the CRC32 polynomial to calculate the data signature. The initial value for the CRC calculation is 0xFFFFFFFF.

$$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x + 1$$

Figure 35-9. Polynomial function for CRC32

The CRC generator uses the XOR-ed combination of the input data bits and the previous result bits.

35.3.8 Clocks

The following table describes the clock sources for eLCDIF. Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 35-4. Clocks

Clock Name	Clock Root	Description
apb_clk	ipg_clk_root	AXI Clock
pix_clk	lcdif_clk_root	Pixel Clock

35.3.9 Reset

The Bus Clock (apb_clk) and Display Clock (pix_clk) must be running before making any changes to LCDIF_CTRL[SFTRST] or LCDIF_CTRL[CLKGATE] bits. A soft reset (SFTRST) can take multiple clock periods to complete, so do not set CLKGATE when setting SFTRST. The reset process gates the clocks automatically.

To reset the eLCDIF module, follow the steps shown below:

1. Ungate the clock by setting LCDIF_CTRL[CLKGATE] =0
2. Set LCDIF_CTRL[SFTRST] =1 to trigger reset
3. Wait until LCDIF_CTRL[CLKGATE] asserts
4. Repeat step 1
5. Set LCDIF_CTRL[SFTRST]= 0 to release reset

35.3.10 Interrupts

The eLCDIF module supports a number of interrupts to aid controlling and status reporting of the block.

All the interrupts have individual mask bits for enabling or disabling each of them. They all get funneled through a single interrupt line connected to the interrupt collector (ICOLL).

The following list describes the different interrupts supported by eLCDIF:

- Underflow interrupt is asserted when the clock domain crossing FIFO (TXFIFO) becomes empty but the block is in active display portion during that time. Software should take corrective action to make sure that this does not happen.

- In the bus master mode, the overflow interrupt will be asserted if the block has requested more data than it's FIFOs could hold.
- Cur_frame_done interrupt occurs at the end of every frame .

35.4 External Signals

The following table describes the external signals of eLCDIF:

Table 35-5. LCD External Signals

Signal	Description	Pad	Mode	Direction
LCD_CLK	Clock signal	GPIO_B0_00 pin 10	ALT0	O
LCD_ENABLE	Enable signal	GPIO_B0_01 pin 12	ALT0	IO
LCD_HSYNC	HSYNC signal	GPIO_B0_02 pin 11	ALT0	O
LCD_VSYNC	VSYNC Signal	GPIO_B0_03 pin 13	ALT0	O
LCD_DATA00	Data Signal	GPIO_B0_04 pin 40 (M1)	ALT0	IO
LCD_DATA01	Data Signal	GPIO_B0_05 pin 41 (M1)	ALT0	IO
LCD_DATA02	Data Signal	GPIO_B0_06 pin 42 (M1)	ALT0	IO
LCD_DATA03	Data Signal	GPIO_B0_07 pin 43 (M1)	ALT0	IO
LCD_DATA04	Data Signal	GPIO_B0_08 pin 44 (M1)	ALT0	IO
LCD_DATA05	Data Signal	GPIO_B0_09 pin 45 (M1)	ALT0	IO
LCD_DATA06	Data Signal	GPIO_B0_10 pin 6	ALT0	IO
LCD_DATA07	Data Signal	GPIO_B0_11 pin 9	ALT0	IO
LCD_DATA08	Data Signal	GPIO_B0_12 pin 32	ALT0	IO
LCD_DATA09	Data Signal	GPIO_B0_13	ALT0	IO
LCD_DATA10	Data Signal	GPIO_B0_14	ALT0	IO
LCD_DATA11	Data Signal	GPIO_B0_15	ALT0	IO
LCD_DATA12	Data Signal	GPIO_B1_00 pin 8	ALT0	IO
LCD_DATA13	Data Signal	GPIO_B1_01 pin 7	ALT0	IO
LCD_DATA14	Data Signal	GPIO_B1_02 pin 36 (T4.1)	ALT0	IO
LCD_DATA15	Data Signal	GPIO_B1_03 pin 37 (T4.1)	ALT0	IO
LCD_DATA16	Data Signal	GPIO_B1_04	ALT0	IO
LCD_DATA17	Data Signal	GPIO_B1_05	ALT0	IO
LCD_DATA18	Data Signal	GPIO_B1_06	ALT0	IO
LCD_DATA19	Data Signal	GPIO_B1_07	ALT0	IO
LCD_DATA20	Data Signal	GPIO_B1_08	ALT0	IO
LCD_DATA21	Data Signal	GPIO_B1_09	ALT0	IO
LCD_DATA22	Data Signal	GPIO_B1_10	ALT0	IO
LCD_DATA23	Data Signal	GPIO_B1_11	ALT0	IO

35.5 Initialization

This section describes write modes.

35.5.1 Write Modes

The following initialization steps are common to all eLCDIF write modes of operation before entering any particular mode.

Initialization steps:

1. Configure the external I/Os to correctly interface the external display, when required.
2. Start the Display Clock (pix_clk) clock and set the appropriate frequency by programming the registers in CCM.
3. Start the Bus Clock (apb_clk) and set the appropriate frequency by programming the registers in CCM.
4. Bring the eLCDIF out of soft reset and disable the clock gate bit.
5. Set the transfer mode of operation to bus master. The LCDIF_CTRL[MASTER] bit determines the transfer mode selected. To select bus master mode, set LCDIF_CTRL[MASTER] =1 (to select APBDMA, set LCDIF_CTRL[MASTER] =0).
6. Set the LCDIF_CTRL[INPUT_DATA_SWIZZLE] according to the endianness of the eLCDIF controller. Also, set the LCDIF_CTRL[DATA_SHIFT_DIR] and LCDIF_CTRL[SHIFT_NUM_BITS] if it is required to shift the data left or right before it is output.
7. Set the LCDIF_CTRL[WORD_LENGTH] field appropriately: 0 = 16-bit input, 1 = 8-bit input, 2 = 18-bit input, 3 = 24/32-bit input. Also, select the correct 16/18/24 bit data format with the corresponding fields in LCDIF_CTRL register.
8. Set the LCDIF_CTRL1[BYTE_PACKING_FORMAT] field according to the input frame.
9. Set the LCDIF_CTRL[LCD_DATABUS_WIDTH] appropriately: 0 = 16-bit output, 1 = 8-bit output, 2 = 18-bit output, 3 = 24/32-bit output.
10. Enable the necessary IRQs.

35.6 eLCDIF Memory Map/Register Definition

Some of the eLCDIF registers (XXX_SET, XXX_CLR, and XXX_TOG) allow direct bit field masking and access.

- When writing 1 to XXX_SET bit fields, these registers allow setting the masked 1 bit fields, while keeping unchanged all bit fields which remain on 0 logic state.
- When writing 1 to XXX_CLR bit fields, these registers allow clearing the masked 1 bit fields, while keeping unchanged all other bit fields which remained on 0 logic state.
- When writing 1 to XXX_TOG bit fields, these registers allow inverting the logic state of all masked 1 bit fields, while they keep unchanged the remaining bit fields which were kept on 0 logic state.

LCDIF Hardware Register Format Summary

LCDIF memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
402B_8000	LCDIF General Control Register (LCDIF_CTRL)	32	R/W	C000_0000h	35.6.1/1882
402B_8004	LCDIF General Control Register (LCDIF_CTRL_SET)	32	R/W	C000_0000h	35.6.1/1882
402B_8008	LCDIF General Control Register (LCDIF_CTRL_CLR)	32	R/W	C000_0000h	35.6.1/1882
402B_800C	LCDIF General Control Register (LCDIF_CTRL_TOG)	32	R/W	C000_0000h	35.6.1/1882
402B_8010	LCDIF General Control1 Register (LCDIF_CTRL1)	32	R/W	000F_0000h	35.6.2/1884
402B_8014	LCDIF General Control1 Register (LCDIF_CTRL1_SET)	32	R/W	000F_0000h	35.6.2/1884
402B_8018	LCDIF General Control1 Register (LCDIF_CTRL1_CLR)	32	R/W	000F_0000h	35.6.2/1884
402B_801C	LCDIF General Control1 Register (LCDIF_CTRL1_TOG)	32	R/W	000F_0000h	35.6.2/1884
402B_8020	LCDIF General Control2 Register (LCDIF_CTRL2)	32	R/W	0020_0000h	35.6.3/1887
402B_8024	LCDIF General Control2 Register (LCDIF_CTRL2_SET)	32	R/W	0020_0000h	35.6.3/1887
402B_8028	LCDIF General Control2 Register (LCDIF_CTRL2_CLR)	32	R/W	0020_0000h	35.6.3/1887
402B_802C	LCDIF General Control2 Register (LCDIF_CTRL2_TOG)	32	R/W	0020_0000h	35.6.3/1887
402B_8030	LCDIF Horizontal and Vertical Valid Data Count Register (LCDIF_TRANSFER_COUNT)	32	R/W	0001_0000h	35.6.4/1889
402B_8040	LCD Interface Current Buffer Address Register (LCDIF_CUR_BUF)	32	R/W	0000_0000h	35.6.5/1889
402B_8050	LCD Interface Next Buffer Address Register (LCDIF_NEXT_BUF)	32	R/W	0000_0000h	35.6.6/1890
402B_8070	LCDIF VSYNC Mode and Dotclk Mode Control Register0 (LCDIF_VDCTRL0)	32	R/W	0000_0000h	35.6.7/1890
402B_8074	LCDIF VSYNC Mode and Dotclk Mode Control Register0 (LCDIF_VDCTRL0_SET)	32	R/W	0000_0000h	35.6.7/1890
402B_8078	LCDIF VSYNC Mode and Dotclk Mode Control Register0 (LCDIF_VDCTRL0_CLR)	32	R/W	0000_0000h	35.6.7/1890
402B_807C	LCDIF VSYNC Mode and Dotclk Mode Control Register0 (LCDIF_VDCTRL0_TOG)	32	R/W	0000_0000h	35.6.7/1890
402B_8080	LCDIF VSYNC Mode and Dotclk Mode Control Register1 (LCDIF_VDCTRL1)	32	R/W	0000_0000h	35.6.8/1892

Table continues on the next page...

LCDIF memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
402B_8090	LCDIF VSYNC Mode and Dotclk Mode Control Register2 (LCDIF_VDCTRL2)	32	R/W	0000_0000h	35.6.9/1892
402B_80A0	LCDIF VSYNC Mode and Dotclk Mode Control Register3 (LCDIF_VDCTRL3)	32	R/W	0000_0000h	35.6.10/ 1893
402B_80B0	LCDIF VSYNC Mode and Dotclk Mode Control Register4 (LCDIF_VDCTRL4)	32	R/W	0000_0000h	35.6.11/ 1894
402B_8190	Bus Master Error Status Register (LCDIF_BM_ERROR_STAT)	32	R/W	0000_0000h	35.6.12/ 1895
402B_81A0	CRC Status Register (LCDIF_CRC_STAT)	32	R/W	0000_0000h	35.6.13/ 1895
402B_81B0	LCD Interface Status Register (LCDIF_STAT)	32	R	9500_0000h	35.6.14/ 1896
402B_8380	LCDIF Pigeon Mode Control0 Register (LCDIF_PIGEONCTRL0)	32	R/W	0000_0000h	35.6.15/ 1898
402B_8384	LCDIF Pigeon Mode Control0 Register (LCDIF_PIGEONCTRL0_SET)	32	R/W	0000_0000h	35.6.15/ 1898
402B_8388	LCDIF Pigeon Mode Control0 Register (LCDIF_PIGEONCTRL0_CLR)	32	R/W	0000_0000h	35.6.15/ 1898
402B_838C	LCDIF Pigeon Mode Control0 Register (LCDIF_PIGEONCTRL0_TOG)	32	R/W	0000_0000h	35.6.15/ 1898
402B_8390	LCDIF Pigeon Mode Control1 Register (LCDIF_PIGEONCTRL1)	32	R/W	0000_0000h	35.6.16/ 1898
402B_8394	LCDIF Pigeon Mode Control1 Register (LCDIF_PIGEONCTRL1_SET)	32	R/W	0000_0000h	35.6.16/ 1898
402B_8398	LCDIF Pigeon Mode Control1 Register (LCDIF_PIGEONCTRL1_CLR)	32	R/W	0000_0000h	35.6.16/ 1898
402B_839C	LCDIF Pigeon Mode Control1 Register (LCDIF_PIGEONCTRL1_TOG)	32	R/W	0000_0000h	35.6.16/ 1898
402B_83A0	LCDIF Pigeon Mode Control2 Register (LCDIF_PIGEONCTRL2)	32	R/W	0000_0000h	35.6.17/ 1899
402B_83A4	LCDIF Pigeon Mode Control2 Register (LCDIF_PIGEONCTRL2_SET)	32	R/W	0000_0000h	35.6.17/ 1899
402B_83A8	LCDIF Pigeon Mode Control2 Register (LCDIF_PIGEONCTRL2_CLR)	32	R/W	0000_0000h	35.6.17/ 1899
402B_83AC	LCDIF Pigeon Mode Control2 Register (LCDIF_PIGEONCTRL2_TOG)	32	R/W	0000_0000h	35.6.17/ 1899
402B_8800	Panel Interface Signal Generator Register (LCDIF_PIGEON_0_0)	32	R/W	0000_0000h	35.6.18/ 1900
402B_8810	Panel Interface Signal Generator Register (LCDIF_PIGEON_0_1)	32	R/W	0000_0000h	35.6.19/ 1901
402B_8820	Panel Interface Signal Generator Register (LCDIF_PIGEON_0_2)	32	R/W	0000_0000h	35.6.20/ 1901
402B_8840	Panel Interface Signal Generator Register (LCDIF_PIGEON_1_0)	32	R/W	0000_0000h	35.6.18/ 1900

Table continues on the next page...

LCDIF memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
402B_8850	Panel Interface Signal Generator Register (LCDIF_PIGEON_1_1)	32	R/W	0000_0000h	35.6.19/ 1901
402B_8860	Panel Interface Signal Generator Register (LCDIF_PIGEON_1_2)	32	R/W	0000_0000h	35.6.20/ 1901
402B_8880	Panel Interface Signal Generator Register (LCDIF_PIGEON_2_0)	32	R/W	0000_0000h	35.6.18/ 1900
402B_8890	Panel Interface Signal Generator Register (LCDIF_PIGEON_2_1)	32	R/W	0000_0000h	35.6.19/ 1901
402B_88A0	Panel Interface Signal Generator Register (LCDIF_PIGEON_2_2)	32	R/W	0000_0000h	35.6.20/ 1901
402B_88C0	Panel Interface Signal Generator Register (LCDIF_PIGEON_3_0)	32	R/W	0000_0000h	35.6.18/ 1900
402B_88D0	Panel Interface Signal Generator Register (LCDIF_PIGEON_3_1)	32	R/W	0000_0000h	35.6.19/ 1901
402B_88E0	Panel Interface Signal Generator Register (LCDIF_PIGEON_3_2)	32	R/W	0000_0000h	35.6.20/ 1901
402B_8900	Panel Interface Signal Generator Register (LCDIF_PIGEON_4_0)	32	R/W	0000_0000h	35.6.18/ 1900
402B_8910	Panel Interface Signal Generator Register (LCDIF_PIGEON_4_1)	32	R/W	0000_0000h	35.6.19/ 1901
402B_8920	Panel Interface Signal Generator Register (LCDIF_PIGEON_4_2)	32	R/W	0000_0000h	35.6.20/ 1901
402B_8940	Panel Interface Signal Generator Register (LCDIF_PIGEON_5_0)	32	R/W	0000_0000h	35.6.18/ 1900
402B_8950	Panel Interface Signal Generator Register (LCDIF_PIGEON_5_1)	32	R/W	0000_0000h	35.6.19/ 1901
402B_8960	Panel Interface Signal Generator Register (LCDIF_PIGEON_5_2)	32	R/W	0000_0000h	35.6.20/ 1901
402B_8980	Panel Interface Signal Generator Register (LCDIF_PIGEON_6_0)	32	R/W	0000_0000h	35.6.18/ 1900
402B_8990	Panel Interface Signal Generator Register (LCDIF_PIGEON_6_1)	32	R/W	0000_0000h	35.6.19/ 1901
402B_89A0	Panel Interface Signal Generator Register (LCDIF_PIGEON_6_2)	32	R/W	0000_0000h	35.6.20/ 1901
402B_89C0	Panel Interface Signal Generator Register (LCDIF_PIGEON_7_0)	32	R/W	0000_0000h	35.6.18/ 1900
402B_89D0	Panel Interface Signal Generator Register (LCDIF_PIGEON_7_1)	32	R/W	0000_0000h	35.6.19/ 1901
402B_89E0	Panel Interface Signal Generator Register (LCDIF_PIGEON_7_2)	32	R/W	0000_0000h	35.6.20/ 1901
402B_8A00	Panel Interface Signal Generator Register (LCDIF_PIGEON_8_0)	32	R/W	0000_0000h	35.6.18/ 1900
402B_8A10	Panel Interface Signal Generator Register (LCDIF_PIGEON_8_1)	32	R/W	0000_0000h	35.6.19/ 1901

Table continues on the next page...

LCDIF memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
402B_8A20	Panel Interface Signal Generator Register (LCDIF_PIGEON_8_2)	32	R/W	0000_0000h	35.6.20/ 1901
402B_8A40	Panel Interface Signal Generator Register (LCDIF_PIGEON_9_0)	32	R/W	0000_0000h	35.6.18/ 1900
402B_8A50	Panel Interface Signal Generator Register (LCDIF_PIGEON_9_1)	32	R/W	0000_0000h	35.6.19/ 1901
402B_8A60	Panel Interface Signal Generator Register (LCDIF_PIGEON_9_2)	32	R/W	0000_0000h	35.6.20/ 1901
402B_8A80	Panel Interface Signal Generator Register (LCDIF_PIGEON_10_0)	32	R/W	0000_0000h	35.6.18/ 1900
402B_8A90	Panel Interface Signal Generator Register (LCDIF_PIGEON_10_1)	32	R/W	0000_0000h	35.6.19/ 1901
402B_8AA0	Panel Interface Signal Generator Register (LCDIF_PIGEON_10_2)	32	R/W	0000_0000h	35.6.20/ 1901
402B_8AC0	Panel Interface Signal Generator Register (LCDIF_PIGEON_11_0)	32	R/W	0000_0000h	35.6.18/ 1900
402B_8AD0	Panel Interface Signal Generator Register (LCDIF_PIGEON_11_1)	32	R/W	0000_0000h	35.6.19/ 1901
402B_8AE0	Panel Interface Signal Generator Register (LCDIF_PIGEON_11_2)	32	R/W	0000_0000h	35.6.20/ 1901
402B_8B00	Look Up Table Control Register (LCDIF_LUT_CTRL)	32	R/W	0000_0001h	35.6.21/ 1902
402B_8B10	Lookup Table 0 Index Register (LCDIF_LUT0_ADDR)	32	R/W	0000_0000h	35.6.22/ 1903
402B_8B20	Lookup Table 0 Data Register (LCDIF_LUT0_DATA)	32	R/W	0000_0000h	35.6.23/ 1903
402B_8B30	Lookup Table 1 Index Register (LCDIF_LUT1_ADDR)	32	R/W	0000_0000h	35.6.24/ 1904
402B_8B40	Lookup Table 1 Data Register (LCDIF_LUT1_DATA)	32	R/W	0000_0000h	35.6.25/ 1904

35.6.1 LCDIF General Control Register (LCDIF_CTRL*n*)

The LCD Interface Control Register provides overall control of the LCDIF block. The LCDIF Control Register provides a variety of control functions to the programmer. These functions allow the interface to be very flexible to work with a variety of LCD controllers, and to minimize overhead and increase performance of LCD programming. The register has been organized such that switching between the different LCD modes can be done with minimum PIO writes.

Address: 402B_8000h base + 0h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SFTRST	CLKGATE	-	Reserved	Reserved	DATA_SHIFT_DIR	SHIFT_NUM_BITS					Reserved	BYPASS_COUNT	Reserved	DOTCLK_MODE	Reserved
Reset	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	INPUT_ DATA_ SWIZZLE		CSC_DATA_SWIZZLE		LCD_ DATABUS_ WIDTH		WORD_ LENGTH		Reserved	ENABLE_PXP_ HANDSHAKE	MASTER	Reserved	DATA_FORMAT_16_BIT	DATA_FORMAT_18_BIT	DATA_FORMAT_24_BIT	RUN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIF_CTRL*n* field descriptions

Field	Description
31 SFTRST	This bit must be set to zero to enable normal operation of the LCDIF. When set to one, it forces a block level reset.
30 CLKGATE	This bit must be set to zero for normal operation. When set to one it gates off the clocks to the block.
29 -	Reserved
28 -	This field is reserved. Reserved

Table continues on the next page...

LCDIF_CTRL*n* field descriptions (continued)

Field	Description
27 - Reserved	This field is reserved.
26 DATA_SHIFT_DIR	Use this bit to determine the direction of shift of transmit data. 0x0 **TXDATA_SHIFT_LEFT** — Data to be transmitted is shifted LEFT by SHIFT_NUM_BITS bits. 0x1 **TXDATA_SHIFT_RIGHT** — Data to be transmitted is shifted RIGHT by SHIFT_NUM_BITS bits.
25–21 SHIFT_NUM_BITS	The data to be transmitted is shifted left or right by this number of bits.
20 - Reserved	This field is reserved.
19 BYPASS_COUNT	When this bit is 0, it means that LCDIF will stop the block operation and turn off the RUN bit after the amount of data indicated by the LCDIF_TRANSFER_COUNT register has been transferred out. When this bit is set to 1, the block will continue normal operation indefinitely until it is told to stop. This bit must be 0 in MPU and VSYNC modes, and must be 1 in DOTCLK and DVI modes of operation.
18 - Reserved	This field is reserved.
17 DOTCLK_MODE	Set this bit to 1 to make the hardware go into the DOTCLK mode, i.e. VSYNC/HSYNC/DOTCLK/ENABLE interface mode. ENABLE is optional, selected by the ENABLE_PRESENT bit. Toggle this bit from 1 to 0 to make the hardware go out of DOTCLK mode after completing all data transfer and deasserting the RUN bit.
16 - Reserved	This field is reserved.
15–14 INPUT_DATA_SWIZZLE	This field specifies how to swap the bytes fetched by the bus master interface. The swizzle function is independent of the WORD_LENGTH bit. The supported swizzle configurations are: 0x0 **NO_SWAP** — No byte swapping.(Little endian) 0x0 **LITTLE_ENDIAN** — Little Endian byte ordering (same as NO_SWAP). 0x1 **BIG_ENDIAN_SWAP** — Big Endian swap (swap bytes 0,3 and 1,2). 0x1 **SWAP_ALL_BYTES** — Swizzle all bytes, swap bytes 0,3 and 1,2 (aka Big Endian). 0x2 **HWD_SWAP** — Swap half-words. 0x3 **HWD_BYTE_SWAP** — Swap bytes within each half-word.
13–12 CSC_DATA_SWIZZLE	This field specifies how to swap the bytes after the data has been converted into an internal representation of 24 bits per pixel and before it is transmitted over the LCD interface bus. The data is always transmitted with the least significant byte/hword (half word) first after the swizzle takes place. So, INPUT_DATA_SWIZZLE takes place first on the incoming data, and then CSC_DATA_SWIZZLE is applied. The swizzle function is independent of the WORD_LENGTH or the LCD_DATABUS_WIDTH fields. If RGB_TO_YCRCB422_CSC bit is set, the swizzle occurs on the Y, Cb, Cr values. The supported swizzle configurations are: 0x0 **NO_SWAP** — No byte swapping.(Little endian) 0x0 **LITTLE_ENDIAN** — Little Endian byte ordering (same as NO_SWAP). 0x1 **BIG_ENDIAN_SWAP** — Big Endian swap (swap bytes 0,3 and 1,2). 0x1 **SWAP_ALL_BYTES** — Swizzle all bytes, swap bytes 0,3 and 1,2 (aka Big Endian). 0x2 **HWD_SWAP** — Swap half-words. 0x3 **HWD_BYTE_SWAP** — Swap bytes within each half-word.
11–10 LCD_DATABUS_WIDTH	LCD Data bus transfer width. **NOTE:** When LUT enabled, this field should be set to 0x01.

Table continues on the next page...

LCDIF_CTRL*n* field descriptions (continued)

Field	Description
	0x0 **16_BIT** — 16-bit data bus mode. 0x1 **8_BIT** — 8-bit data bus mode. 0x2 **18_BIT** — 18-bit data bus mode. 0x3 **24_BIT** — 24-bit data bus mode.
9–8 WORD_LENGTH	Input data format. 0x0 **16_BIT** — Input data is 16 bits per pixel. 0x1 **8_BIT** — Input data is 8 bits wide. 0x2 **18_BIT** — Input data is 18 bits per pixel. 0x3 **24_BIT** — Input data is 24 bits per pixel.
7 -	This field is reserved. Reserved
6 ENABLE_PXP_HANDSHAKE	If this bit is set and LCDIF_MASTER bit is set, the LCDIF will act as bus master and the handshake mechanism between LCDIF and PXP will be turned on. If LCDIF_MASTER bit is not set, this bit becomes a don't care.
5 MASTER	Set this bit to make the LCDIF act as a bus master. If this bit is reset, the LCDIF will support APBDMA or PIO mode.
4 RSRVD0	This field is reserved. Reserved bits. Write as 0.
3 DATA_FORMAT_16_BIT	When this bit is 1 and WORD_LENGTH = 0, it implies that the 16-bit data is in ARGB555 format. When this bit is 0 and WORD_LENGTH = 0, it implies that the 16-bit data is in RGB565 format. When WORD_LENGTH is not 0, this bit does not care.
2 DATA_FORMAT_18_BIT	Used only when WORD_LENGTH = 2, i.e. 18-bit. 0x0 **LOWER_18_BITS_VALID** — Data input to the block is in 18 bpp format, such that lower 18 bits contain RGB 666 and upper 14 bits do not contain any useful data. 0x1 **UPPER_18_BITS_VALID** — Data input to the block is in 18 bpp format, such that upper 18 bits contain RGB 666 and lower 14 bits do not contain any useful data.
1 DATA_FORMAT_24_BIT	Used only when WORD_LENGTH = 3, i.e. 24-bit. Note that this applies to both packed and unpacked 24-bit data. 0x0 **ALL_24_BITS_VALID** — Data input to the block is in 24 bpp format, such that all RGB 888 data is contained in 24 bits. 0x1 **DROP_UPPER_2_BITS_PER_BYTE** — Data input to the block is actually RGB 18 bpp, but there is 1 color per byte, hence the upper 2 bits in each byte do not contain any useful data, and should be dropped.
0 RUN	When this bit is set by software, the LCDIF will begin transferring data between the SoC and the display. This bit must remain set until the operation is complete.

35.6.2 LCDIF General Control1 Register (LCDIF_CTRL1*n*)

The LCDIF Control Register provides overall control of the LCDIF block.

The LCDIF Control1 Register provides additional programming to the LCDIF. It implements some bits which are unlikely to change often in a particular application. It also carries interrupt-related bits which are common across more than one mode of operation.

Address: 402B_8000h base + 10h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	IMAGE_DATA_SELECT	CS_OUT_SELECT	Reserved		Reserved	BM_ERROR_IRQ_EN	BM_ERROR_IRQ	RECOVER_ON_UNDERFLOW	INTERLACE_FIELDS	START_INTERLACE_FROM_SECOND_FIELD	FIFO_CLEAR	IRQ_ON_ALTERNATE_FIELDS	BYTE_PACKING_FORMAT			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	OVERFLOW_IRQ_EN	UNDERFLOW_IRQ_EN	CUR_FRAME_DONE_IRQ_EN	VSYNC_EDGE_IRQ_EN	OVERFLOW_IRQ	UNDERFLOW_IRQ	CUR_FRAME_DONE_IRQ	VSYNC_EDGE_IRQ	Reserved					Reserved	Reserved	Reserved
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIF_CTRL1*n* field descriptions

Field	Description
31 IMAGE_DATA_SELECT	Command Mode MIPI image data select bit. This bit will control MIPI DSI SYS_ADDR[1]. This bit should only be changed when RUN is 0.
30 CS_OUT_SELECT	This bit is CS0/CS1 valid select signals. When set 0x0, LCDIF_CSn0 output valid, and LCSIF_CSn1 will alway output 1. WHne set to 0x1, LCDIF_CSn1 output valid, and LCSIF_CSn0 will alway output 1.
29–28 -	This field is reserved. Reserved bits. Write as 0.
27 -	This field is reserved. Reserved.
26 BM_ERROR_IRQ_EN	This bit is set to enable bus master error interrupt in the LCDIF master mode.
25 BM_ERROR_IRQ	This bit is set to indicate that an interrupt is requested by the LCDIF block. This bit is cleared by software by writing a one to its SCT clear address. This bit will be set when the LCDIF is in master mode and an error response was returned by the slave.

Table continues on the next page...

LCDIF_CTRL1*n* field descriptions (continued)

Field	Description
	0x0 **NO_REQUEST** — No Interrupt Request Pending. 0x1 **REQUEST** — Interrupt Request Pending.
24 RECOVER_ON_ UNDERFLOW	Set this bit to enable the LCDIF block to recover in the next field/frame if there was an underflow in the current field/frame.
23 INTERLACE_ FIELDS	Set this bit if it is required that the LCDIF block fetches odd lines in one field and even lines in the other field. It will work only in LCDIF_MASTER is set to 1.
22 START_ INTERLACE_ FROM_ SECOND_FIELD	The default is to grab the odd lines first and then the even lines. Set this bit if it is required to grab the even lines first and then the odd lines. (Line numbers start from 1, so odd lines are 1,3,5,etc. and even lines are 2,4,6, etc.)
21 FIFO_CLEAR	Set this bit to clear all the data in the latency FIFO (LFIFO), TXFIFO and the RXFIFO.
20 IRQ_ON_ ALTERNATE_ FIELDS	If this bit is set, the LCDIF block will assert the cur_frame_done interrupt only on alternate fields, otherwise it will issue the interrupt on both odd and even field. This bit is mostly relevant if INTERLACE_FIELDS is set.
19–16 BYTE_ PACKING_ FORMAT	This bitfield is used to show which data bytes in a 32-bit word are valid. Default value 0xf indicates that all bytes are valid. For 8-bit transfers, any combination in this bitfield will mean valid data is present in the corresponding bytes. In the 16-bit mode, a 16-bit half-word is valid only if adjacent bits [1:0] or [3:2] or both are 1. A value of 0x0 will mean that none of the bytes are valid and should not be used. For example, set the bit field value to 0x7 if the display data is arranged in the 24-bit unpacked format (A-R-G-B where A value does not have be transmitted).
15 OVERFLOW_ IRQ_EN	This bit is set to enable an overflow interrupt in the TXFIFO in the write mode.
14 UNDERFLOW_ IRQ_EN	This bit is set to enable an underflow interrupt in the TXFIFO in the write mode.
13 CUR_FRAME_ DONE_IRQ_EN	This bit is set to 1 enable an interrupt every time the hardware enters in the vertical blanking state.
12 VSYNC_EDGE_ IRQ_EN	This bit is set to enable an interrupt every time the hardware encounters the leading VSYNC edge in the VSYNC and DOTCLK modes, or the beginning of every field in DVI mode.
11 OVERFLOW_ IRQ	This bit is set to indicate that an interrupt is requested by the LCDIF block. This bit is cleared by software by writing a one to its SCT clear address. A latency FIFO (LFIFO) overflow in the write mode (MPU/VSYNC/DOTCLK/DVI mode) was detected, data samples have been lost. 0x0 **NO_REQUEST** — No Interrupt Request Pending. 0x1 **REQUEST** — Interrupt Request Pending.
10 UNDERFLOW_ IRQ	This bit is set to indicate that an interrupt is requested by the LCDIF block. This bit is cleared by software by writing a one to its SCT clear address. A TXFIFO underflow in the write mode (MPU/VSYNC/DOTCLK/DVI mode) was detected. Could produce an error in the DOTCLK / DVI modes. 0x0 **NO_REQUEST** — No Interrupt Request Pending. 0x1 **REQUEST** — Interrupt Request Pending.

Table continues on the next page...

LCDIF_CTRL1*n* field descriptions (continued)

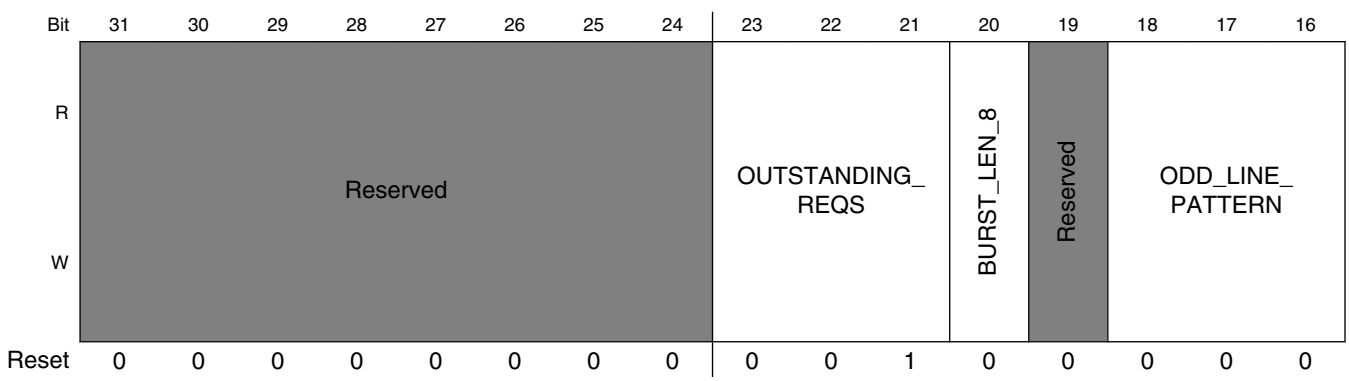
Field	Description
9 CUR_FRAME_DONE_IRQ	This bit is set to indicate that an interrupt is requested by the LCDIF block. This bit is cleared by software by writing a one to its SCT clear address. It indicates that the hardware has completed transmitting the current frame and is in the vertical blanking period in the DOTCLK/DVI modes. In the MPU and VSYNC modes, this IRQ is asserted at the end of the data transfer indicated by LCDIF_TRANSFER_COUNT register. 0x0 **NO_REQUEST** — No Interrupt Request Pending. 0x1 **REQUEST** — Interrupt Request Pending.
8 VSYNC_EDGE_IRQ	This bit is set to indicate that an interrupt is requested by the LCDIF block. This bit is cleared by software by writing a one to its SCT clear address. It is set whenever the leading VSYNC edge is detected in the VSYNC and DOTCLK modes. In the DVI mode, it is asserted every time the block enters a new field. 0x0 **NO_REQUEST** — No Interrupt Request Pending. 0x1 **REQUEST** — Interrupt Request Pending.
7–3 RSRVD0	This field is reserved. Reserved bits. Write as 0.
2 -	This field is reserved. Reserved
1 -	This field is reserved. Reserved.
0 -	This field is reserved. Reserved.

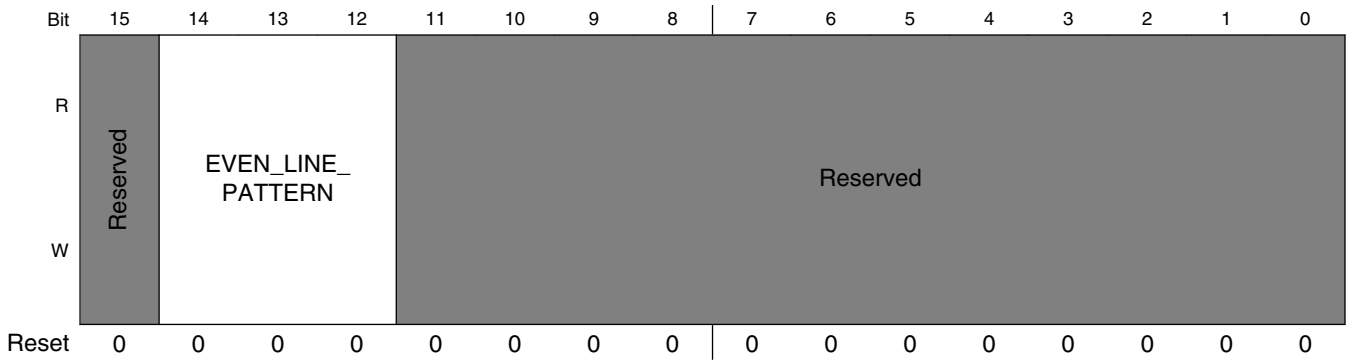
35.6.3 LCDIF General Control2 Register (LCDIF_CTRL2*n*)

The LCDIF Control Register provides overall control of the LCDIF block.

The LCDIF Control2 Register provides additional programming to the LCDIF. It implements some bits which are unlikely to change often in a particular application.

Address: 402B_8000h base + 20h offset + (4d × i), where i=0d to 3d





LCDIF_CTRL2*n* field descriptions

Field	Description
31–24 RSRVD5	This field is reserved. Reserved bits. Write as 0.
23–21 OUTSTANDING_ REQS	This bitfield indicates the maximum number of outstanding transactions that LCDIF should request when it is acting as a bus master. Default is 2 outstanding transactions. 0x0 **REQ_1** — 0x1 **REQ_2** — 0x2 **REQ_4** — 0x3 **REQ_8** — 0x4 **REQ_16** —
20 BURST_LEN_8	By default, when the LCDIF is in the bus master mode, it will issue AXI bursts of length 16 (except when in packed 24 bpp mode, it will issue bursts of length 15). When this bit is set to 1, the block will issue bursts of length 8 (except when in packed 24 bpp mode, it will issue bursts of length 9). Note that this bitfield is only applicable when LCDIF_MASTER is set to 1.
19 RSRVD4	This field is reserved. Reserved bits. Write as 0.
18–16 ODD_LINE_ PATTERN	This field determines the order of the RGB components of each pixel in ODD lines (line numbers 1,3,5,..). 0x0 **RGB** — 0x1 **RBG** — 0x2 **GBR** — 0x3 **GRB** — 0x4 **BRG** — 0x5 **BGR** —
15 RSRVD3	This field is reserved. Reserved bits. Write as 0.
14–12 EVEN_LINE_ PATTERN	This field determines the order of the RGB components of each pixel in EVEN lines (line numbers 2,4,6,..). 0x0 **RGB** — 0x1 **RBG** — 0x2 **GBR** — 0x3 **GRB** — 0x4 **BRG** — 0x5 **BGR** —

Table continues on the next page...

LCDIF_CTRL2*n* field descriptions (continued)

Field	Description
RSRVD0	This field is reserved. Reserved bits. Write as 0.

35.6.4 LCDIF Horizontal and Vertical Valid Data Count Register (LCDIF_TRANSFER_COUNT)

This register tells the LCDIF how much data will be sent for this frame, or transaction. The total number of words is a product of the V_COUNT and H_COUNT fields. The word size is specified by the WORD_LENGTH field.

This register gives the dimensions of the input frame. For normal operation, but V_COUNT and H_COUNT should be non-zero.

Address: 402B_8000h base + 30h offset = 402B_8030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	V_COUNT																H_COUNT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIF_TRANSFER_COUNT field descriptions

Field	Description
31–16 V_COUNT	Number of horizontal lines per frame which contain valid data. In DOTCLK mode, V_COUNT should be the same as the number of active horizontal lines in a progressive frame.
H_COUNT	Total valid data (pixels) in each horizontal line. The data size is given by the WORD_LENGTH. In 24-bit packed format (WORD_LENGTH=0x3, BYTE_PACKING_FORMAT=0xF), the H_COUNT must be a multiple of 4 pixels. In 16-bit packed format (WORD_LENGTH=0x0, BYTE_PACKING_FORMAT=0xF), the H_COUNT must be a multiple of 2 pixels.

35.6.5 LCD Interface Current Buffer Address Register (LCDIF_CUR_BUF)

This register indicates the address of the current frame being transmitted by LCDIF.

When the LCDIF is behaving as a master, this address points to the address of the current frame of data being sent out via the LCDIF. When the current frame is done, the LCDIF block will assert the cur_frame_done interrupt for software to take action. The block will also copy the LCDIF_NEXT_BUF_ADDR into this bitfield so that the software can program the next frame address into the LCDIF_NEXT_BUF_ADDR bitfield. This address must always be double-word aligned.

Address: 402B_8000h base + 40h offset = 402B_8040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIF_CUR_BUF field descriptions

Field	Description
ADDR	Address of the current frame being transmitted by LCDIF.

35.6.6 LCD Interface Next Buffer Address Register (LCDIF_NEXT_BUF)

This register indicates the address of next frame that will be transmitted by LCDIF.

When the LCDIF is behaving as a master, this address points to the address of the next frame of data that will be sent out via the LCDIF. It is up to the software to make sure that this register is programmed before the end of the current frame, otherwise it might result in old data going out the LCDIF. This address must always be double-word aligned.

Address: 402B_8000h base + 50h offset = 402B_8050h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIF_NEXT_BUF field descriptions

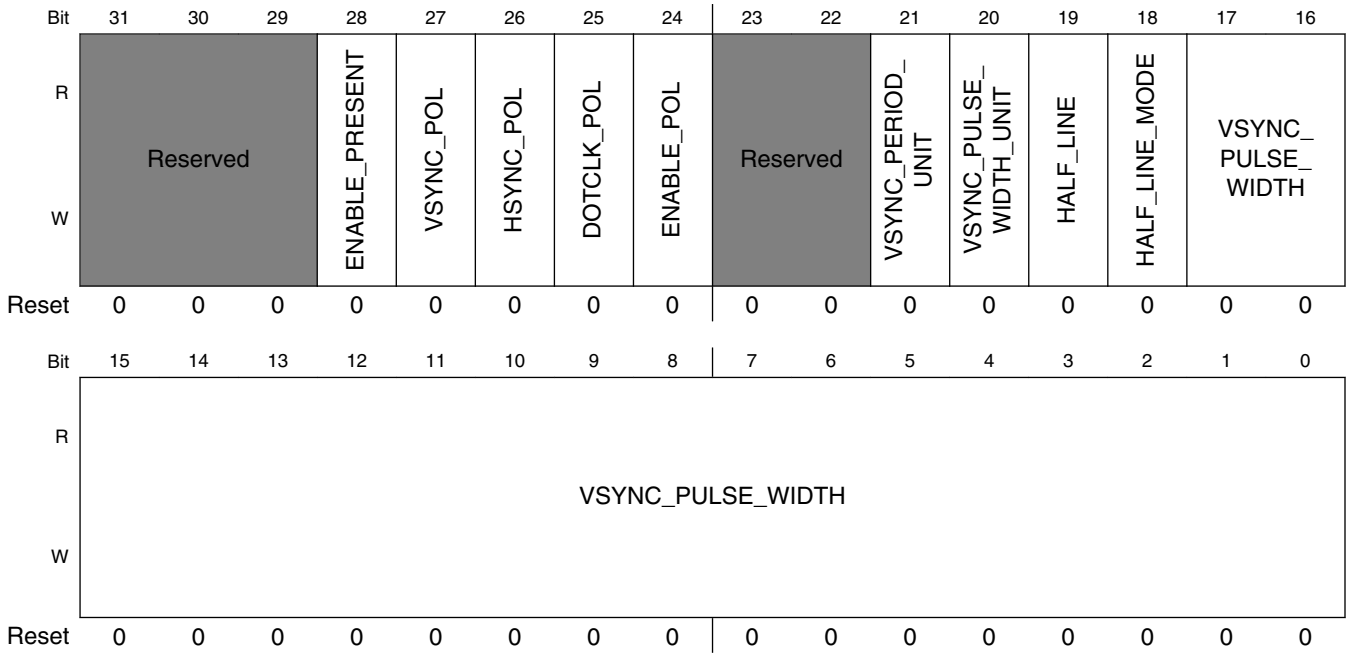
Field	Description
ADDR	Address of the next frame that will be transmitted by LCDIF.

35.6.7 LCDIF VSYNC Mode and Dotclk Mode Control Register0 (LCDIF_VDCTRL0*n*)

This register is used to control the VSYNC and DOTCLK modes of the LCDIF so as to work with different types of LCDs like moving picture displays and delta pixel displays.

This register gives general programmability to the VSYNC signal including polarity, direction, pulse width, etc.

Address: 402B_8000h base + 70h offset + (4d × i), where i=0d to 3d



LCDIF_VDCTRL0*n* field descriptions

Field	Description
31–29 RSRVD2	This field is reserved. Reserved bits. Write as 0.
28 ENABLE_ PRESENT	Setting this bit to 1 will make the hardware generate the ENABLE signal in the DOTCLK mode, thereby making it the true RGB interface along with the remaining three signals VSYNC, HSYNC and DOTCLK.
27 VSYNC_POL	Default 0 active low during VSYNC_PULSE_WIDTH time and will be high during the rest of the VSYNC period. Set it to 1 to invert the polarity.
26 HSYNC_POL	Default 0 active low during HSYNC_PULSE_WIDTH time and will be high during the rest of the HSYNC period. Set it to 1 to invert the polarity.
25 DOTCLK_POL	Default is data launched at negative edge of DOTCLK and captured at positive edge. Set it to 1 to invert the polarity. Set it to 0 in DVI mode.
24 ENABLE_POL	Default 0 active low during valid data transfer on each horizontal line.
23–22 RSRVD1	This field is reserved. Reserved bits. Write as 0.
21 VSYNC_ PERIOD_UNIT	Default 0 for counting VSYNC_PERIOD in terms of DISPLAY CLOCK (pix_clk) cycles. Set it to 1 to count in terms of complete horizontal lines. DISPLAY CLOCK (pix_clk) cycles should be used in the VSYNC mode, while horizontal line should be used in the DOTCLK mode.
20 VSYNC_PULSE_ WIDTH_UNIT	Default 0 for counting VSYNC_PULSE_WIDTH in terms of DISPLAY CLOCK (pix_clk) cycles. Set it to 1 to count in terms of complete horizontal lines.
19 HALF_LINE	Setting this bit to 1 will make the total VSYNC period equal to the VSYNC_PERIOD field plus half the HORIZONTAL_PERIOD field (i.e. VSYNC_PERIOD field plus half horizontal line), otherwise it is just VSYNC_PERIOD. Should be only used in the DOTCLK mode, not in the VSYNC interface mode.
18 HALF_LINE_ MODE	When this bit is 0, the first field (VSYNC period) will end in half a horizontal line and the second field will begin with half a horizontal line. When this bit is 1, all fields will end with half a horizontal line, and none will begin with half a horizontal line.

Table continues on the next page...

LCDIF_VDCTRL0*n* field descriptions (continued)

Field	Description
VSYNC_PULSE_WIDTH	Number of units for which VSYNC signal is active. For the DOTCLK mode, the unit is determined by the VSYNC_PULSE_WIDTH_UNIT. If the VSYNC_PULSE_WIDTH_UNIT is 0 for DOTCLK mode, VSYNC_PULSE_WIDTH must be less than HSYNC_PERIOD. For the VSYNC interface mode, it should be in terms of number of DISPLAY CLOCK (pix_clk) cycles only.

35.6.8 LCDIF VSYNC Mode and Dotclk Mode Control Register1 (LCDIF_VDCTRL1)

This register is used to control the VSYNC signal in the VSYNC and DOTCLK modes of the block.

This register determines the period and duty cycle of the VSYNC signal when it is generated in the block.

Address: 402B_8000h base + 80h offset = 402B_8080h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	VSYNC_PERIOD																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIF_VDCTRL1 field descriptions

Field	Description
VSYNC_PERIOD	Total number of units between two positive or two negative edges of the VSYNC signal. If HALF_LINE is set, it is implicitly calculated to be VSYNC_PERIOD plus half HSYNC_PERIOD.

35.6.9 LCDIF VSYNC Mode and Dotclk Mode Control Register2 (LCDIF_VDCTRL2)

This register is used to control the HSYNC signal in the DOTCLK mode of the block.

This register determines the period and duty cycle of the HSYNC signal when it is generated in the block.

Address: 402B_8000h base + 90h offset = 402B_8090h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	HSYNC_PULSE_WIDTH																HSYNC_PERIOD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIF_VDCTRL2 field descriptions

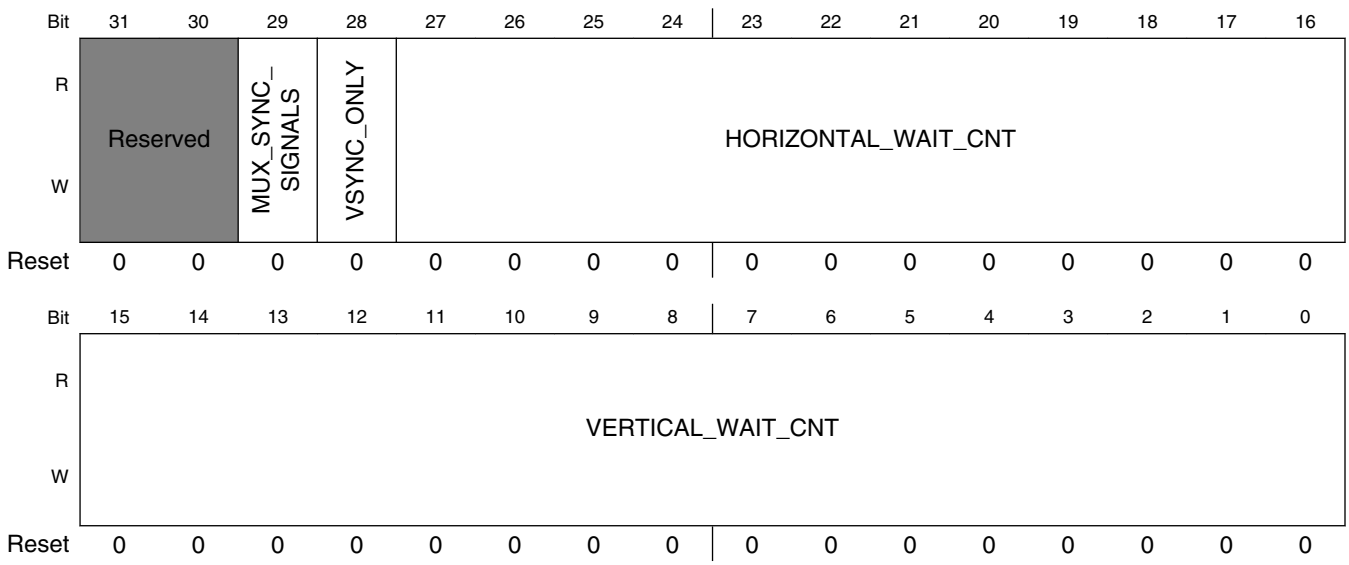
Field	Description
31–18 HSYNC_PULSE_ WIDTH	Number of DISPLAY CLOCK (pix_clk) cycles for which HSYNC signal is active.
HSYNC_PERIOD	Total number of DISPLAY CLOCK (pix_clk) cycles between two positive or two negative edges of the HSYNC signal.

35.6.10 LCDIF VSYNC Mode and Dotclk Mode Control Register3 (LCDIF_VDCTRL3)

This register is used to determine the vertical and horizontal wait counts.

This register determines the back porches of HSYNC and VSYNC signals when they are generated by the block.

Address: 402B_8000h base + A0h offset = 402B_80A0h



LCDIF_VDCTRL3 field descriptions

Field	Description
31–30 RSRVD0	This field is reserved. Reserved bits, write as 0.
29 MUX_SYNC_ SIGNALS	When this bit is set, the LCDIF block will internally mux HSYNC with LCD_D14, DOTCLK with LCD_D13 and ENABLE with LCD_D12, otherwise these signals will go out on separate pins. This feature can be used to maintain backward compatible with 37xx.
28 VSYNC_ONLY	This bit must be set to 1 in the VSYNC mode of operation, and 0 in the DOTCLK mode of operation.

Table continues on the next page...

LCDIF_VDCTRL3 field descriptions (continued)

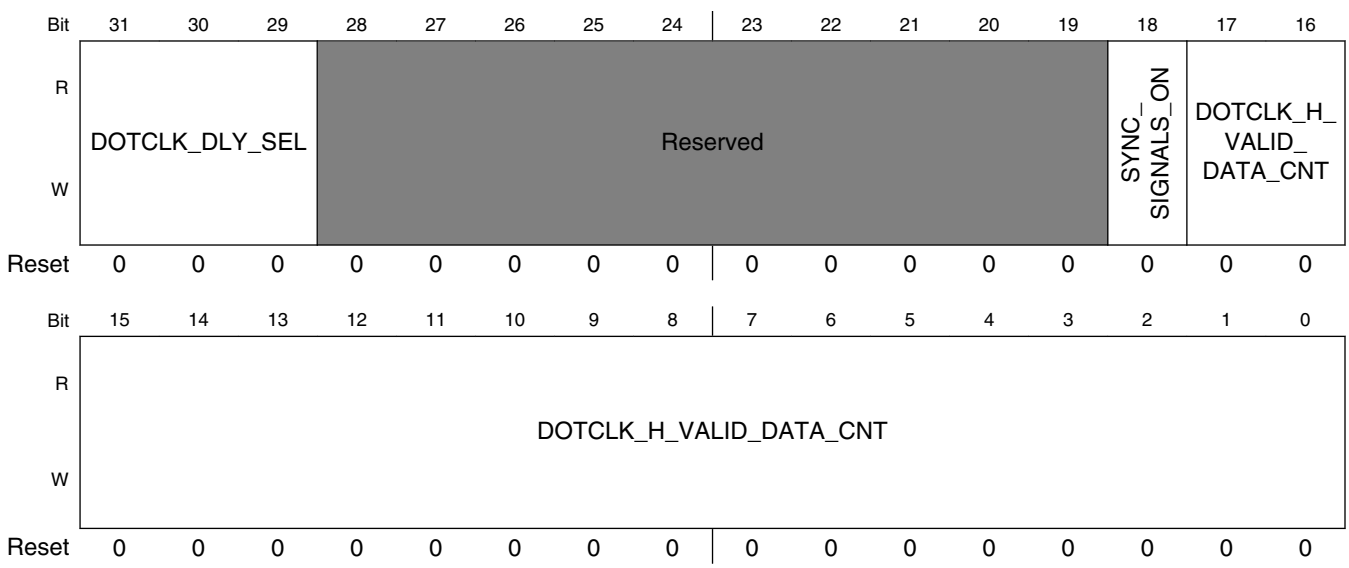
Field	Description
27–16 HORIZONTAL_ WAIT_CNT	In the DOTCLK mode, wait for this number of clocks from falling edge (or rising if HSYNC_POL is 1) of HSYNC signal to account for horizontal back porch plus the number of DOTCLKs before the moving picture information begins.
VERTICAL_ WAIT_CNT	In the VSYNC interface mode, wait for this number of DISPLAY CLOCK (pix_clk) cycles from the falling VSYNC edge (or rising if VSYNC_POL is 1) before starting LCD transactions and is applicable only if WAIT_FOR_VSYNC_EDGE is set. Minimum is CMD_SETUP+5. In the DOTCLK mode, it accounts for the vertical back porch lines plus the number of horizontal lines before the moving picture begins. The unit for this parameter is inherently the same as the VSYNC_PERIOD_UNIT.

35.6.11 LCDIF VSYNC Mode and Dotclk Mode Control Register4 (LCDIF_VDCTRL4)

This register is used to control the DOTCLK mode of the block.

This register determines the active data in each horizontal line in the DOTCLK mode. Note that the total number of active horizontal lines in the DOTCLK mode is the same as the V_COUNT bitfield in the LCDIF_TRANSFER_COUNT register.

Address: 402B_8000h base + B0h offset = 402B_80B0h



LCDIF_VDCTRL4 field descriptions

Field	Description
31–29 DOTCLK_DLY_ SEL	This bitfield selects the amount of time by which the DOTCLK signal should be delayed before coming out of the LCD_DOTCK pin. 0 = 2ns; 1=4ns;2=6ns;3=8ns. Remaining values are reserved.
28–19 RSRVD0	This field is reserved. Reserved bits, write as 0.

Table continues on the next page...

LCDIF_VDCTRL4 field descriptions (continued)

Field	Description
18 SYNC_ SIGNALS_ON	Set this field to 1 if the LCD controller requires that the VSYNC or VSYNC/HSYNC/DOTCLK control signals should be active at least one frame before the data transfers actually start and remain active at least one frame after the data transfers end. The hardware does not count the number of frames automatically. Rather, the VSYNC edge interrupt can be monitored by software to count the number of frames that have occurred after this bit is set and then the RUN bit can be set to start the data transactions. This bit must always be set in the DOTCLK mode of operation, and it must be set in the VSYNC mode of operation when VSYNC signal is an output.
DOTCLK_H_ VALID_DATA_ CNT	Total number of DISPLAY CLOCK (pix_clk) cycles on each horizontal line that carry valid data in DOTCLK mode.

35.6.12 Bus Master Error Status Register (LCDIF_BM_ERROR_STAT)

This register reflects the virtual address at which the AXI master received an error response from the slave.

When the BM_ERROR_IRQ is asserted, the address of the bus error is updated in the register.

Address: 402B_8000h base + 190h offset = 402B_8190h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIF_BM_ERROR_STAT field descriptions

Field	Description
ADDR	Virtual address at which bus master error occurred.

35.6.13 CRC Status Register (LCDIF_CRC_STAT)

This register reflects the CRC value of each frame sent out by LCDIF. The CRC is done on the final output bus, so the value will be dependent on the LCD_DATABUS_WIDTH bitfield even if the input data is the same.

This register will be updated when the CUR_FRAME_DONE_IRQ is asserted.

Address: 402B_8000h base + 1A0h offset = 402B_81A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CRC_VALUE																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIF_CRC_STAT field descriptions

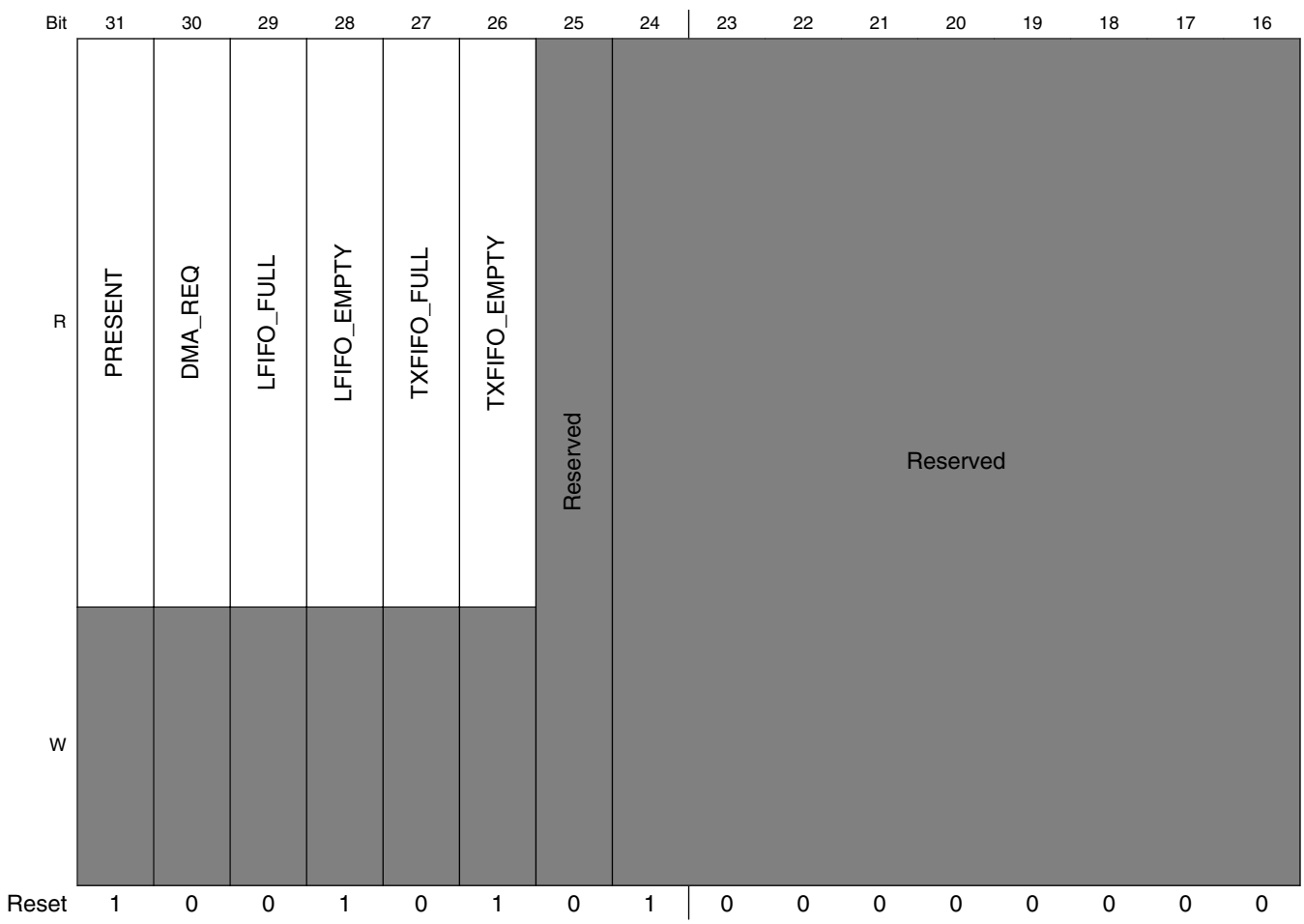
Field	Description
CRC_VALUE	Calculated CRC value. Display output valid data will be calculated by CRC32 and updated during each frame. Refer to CRC Polynomial for details.

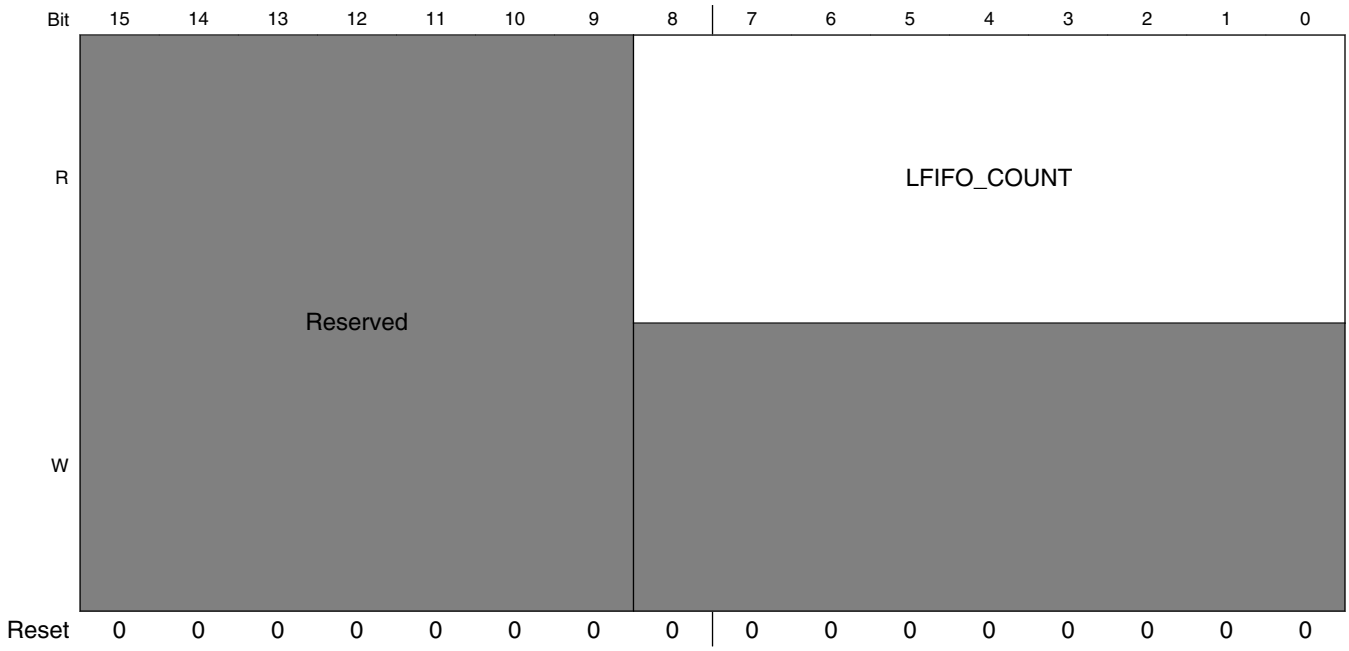
35.6.14 LCD Interface Status Register (LCDIF_STAT)

The LCD interface status register can be used to check the current status of the LCDIF block.

The LCD interface status register that contains read only views of some parameters or current state of the block.

Address: 402B_8000h base + 1B0h offset = 402B_81B0h





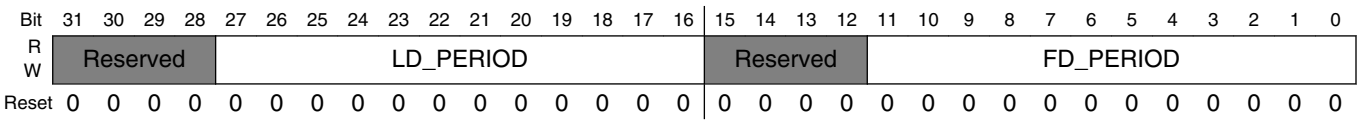
LCDIF_STAT field descriptions

Field	Description
31 PRESENT	0: LCDIF not present on this product 1: LCDIF is present.
30 DMA_REQ	Reflects the current state of the DMA Request line for the LCDIF. The DMA Request line toggles for each new request.
29 LFIFO_FULL	Read only view of the signals that indicates LCD LFIFO is full.
28 LFIFO_EMPTY	Read only view of the signals that indicates LCD LFIFO is empty.
27 TXFIFO_FULL	Read only view of the signals that indicates LCD TXFIFO is full.
26 TXFIFO_EMPTY	Read only view of the signals that indicates LCD TXFIFO is empty.
25 -	This field is reserved. Reserved
24–9 RSRVD0	This field is reserved. Reserved bits. Write as 0.
LFIFO_COUNT	Read only view of the current count in Latency buffer (LFIFO).

35.6.15 LCDIF Pigeon Mode Control0 Register (LCDIF_PIGEONCTRL0*n*)

This register contains global counter settings for Pigeon Mode also houses general purpose timing adjustment registers

Address: 402B_8000h base + 380h offset + (4d × i), where i=0d to 3d



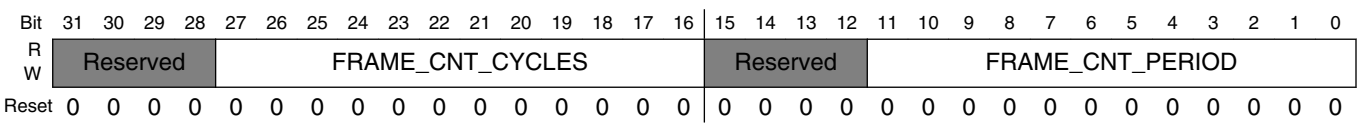
LCDIF_PIGEONCTRL0*n* field descriptions

Field	Description
31–28 -	This field is reserved. Reserved
27–16 LD_PERIOD	Period of pclk counter during LD phase
15–12 -	This field is reserved. Reserved
FD_PERIOD	Period of line counter during FD phase

35.6.16 LCDIF Pigeon Mode Control1 Register (LCDIF_PIGEONCTRL1*n*)

This register contains global counter settings for Pigeon Mode also houses general purpose timing adjustment registers

Address: 402B_8000h base + 390h offset + (4d × i), where i=0d to 3d



LCDIF_PIGEONCTRL1*n* field descriptions

Field	Description
31–28 -	This field is reserved. Reserved

Table continues on the next page...

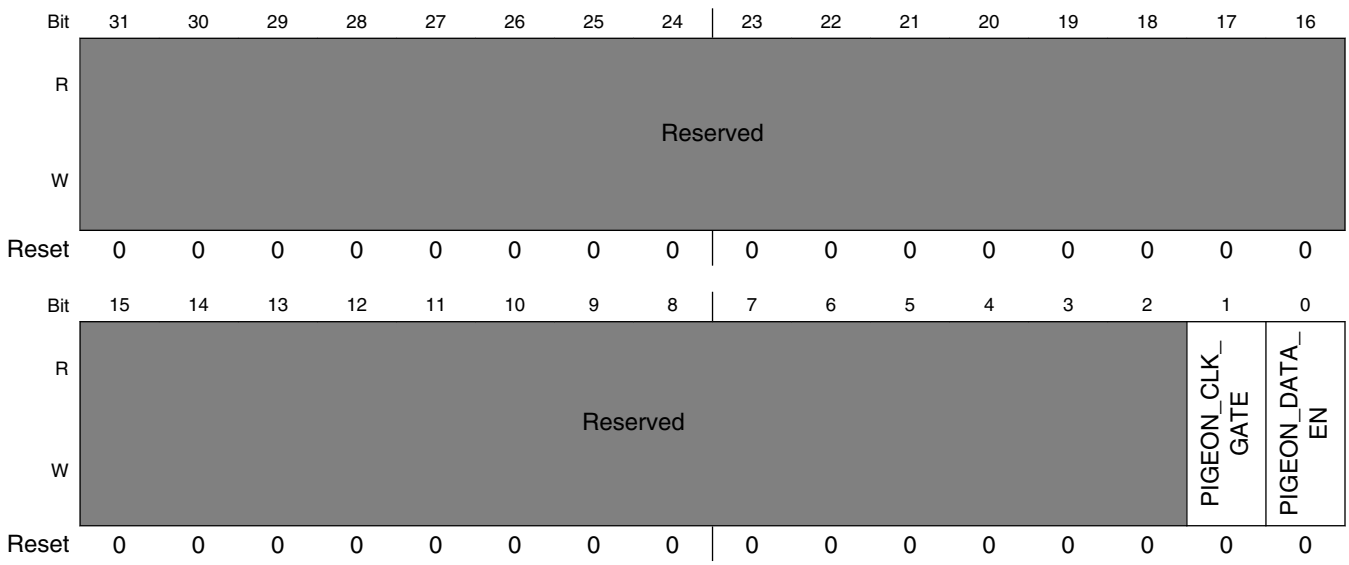
LCDIF_PIGEONCTRL1*n* field descriptions (continued)

Field	Description
27–16 FRAME_CNT_CYCLES	Max cycles of frame counter
15–12 -	This field is reserved. Reserved
FRAME_CNT_PERIOD	Period of frame counter

35.6.17 LCDIF Pigeon Mode Control2 Register (LCDIF_PIGEONCTRL2*n*)

This register contains global counter settings for Pigeon Mode also houses clock gating and data enable registers

Address: 402B_8000h base + 3A0h offset + (4d × i), where i=0d to 3d



LCDIF_PIGEONCTRL2*n* field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
1 PIGEON_CLK_GATE	Pigeon mode dot clock gate enable
0 PIGEON_DATA_EN	Pigeon mode data enable

35.6.18 Panel Interface Signal Generator Register (LCDIF_PIGEON*n*)

This register contains parameters for timing signal generation

Address: 402B_8000h base + 800h offset + (64d × i), where i=0d to 11d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	STATE_MASK								MASK_CNT							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MASK_CNT				MASK_CNT_SEL				OFFSET				INC_SEL		POL	EN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIF_PIGEON*n* field descriptions

Field	Description
31–24 STATE_MASK	state_mask = (FS|FB|FD|FE) and (LS|LB|LD|LE) , select any combination of scan states as reference point for local counter to start ticking 0x1 **FS** — FRAME SYNC 0x2 **FB** — FRAME BEGIN 0x4 **FD** — FRAME DATA 0x8 **FE** — FRAME END 0x10 **LS** — LINE SYNC 0x20 **LB** — LINE BEGIN 0x40 **LD** — LINE DATA 0x80 **LE** — LINE END
23–12 MASK_CNT	When the global counter selected through MASK_CNT_SEL matches value in this reg, pigeon local counter start ticking. 0=disable
11–8 MASK_CNT_SEL	select global counters as mask condition, use together with MASK_CNT 0x0 **HSTATE_CNT** — pclk counter within one hscan state 0x1 **HSTATE_CYCLE** — pclk cycle within one hscan state 0x2 **VSTATE_CNT** — line counter within one vscan state 0x3 **VSTATE_CYCLE** — line cycle within one vscan state 0x4 **FRAME_CNT** — frame counter 0x5 **FRAME_CYCLE** — frame cycle 0x6 **HCNT** — horizontal counter (pclk counter within one line) 0x7 **VCNT** — vertical counter (line counter within one frame)
7–4 OFFSET	offset on pclk unit. 0=align with data, positive value means delay, minus value mean ahead. Supported range depends on panel mode
3–2 INC_SEL	Event to incrment local counter 0x0 **PCLK** — pclk 0x1 **LINE** — Line start pulse

Table continues on the next page...

LCDIF_PIGEON*n* field descriptions (continued)

Field	Description
	0x2 **FRAME** — Frame start pulse 0x3 **SIG_ANOTHER** — Use another signal as tick event
1 POL	Polarity of signal output 0x0 **ACTIVE_HIGH** — Normal Signal (Active high) 0x1 **ACTIVE_LOW** — Inverted signal (Active low)
0 EN	Enable pigeon Mode on this signal

35.6.19 Panel Interface Signal Generator Register (LCDIF_PIGEON*n*)

This register contains parameters for timing signal generation

Address: 402B_8000h base + 810h offset + (64d × i), where i=0d to 11d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CLR_CNT																SET_CNT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIF_PIGEON*n* field descriptions

Field	Description
31–16 CLR_CNT	Deassert signal output when counter match this value 0x0 **CLEAR_USING_MASK** — Keep active until mask off
SET_CNT	Assert signal output when counter match this value 0x0 **START_ACTIVE** — Start as active

35.6.20 Panel Interface Signal Generator Register (LCDIF_PIGEON*n*)

This register contains parameters for timing signal generation

Address: 402B_8000h base + 820h offset + (64d × i), where i=0d to 11d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																							SIG_ANOTHER					SIG_LOGIC			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

LCDIF_PIGEON*n* field descriptions

Field	Description
31–9 RSVD	This field is reserved. Reserved
8–4 SIG_ANOTHER	Select another signal for logic operation or as mask or counter tick event 0 **CLEAR_USING_MASK** — Keep active until mask off
SIG_LOGIC	Logic operation with another signal: DIS/AND/OR/COND 0 **DIS** — No logic operation 1 **AND** — sigout = sig_another AND this_sig 2 **OR** — sigout = sig_another OR this_sig 3 **MASK** — mask = sig_another AND other_masks

35.6.21 Look Up Table Control Register (LCDIF_LUT_CTRL)

Address: 402B_8000h base + B00h offset = 402B_8B00h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

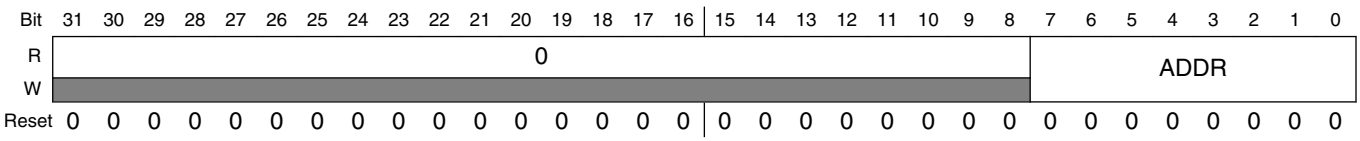
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0															LUT_BYPASS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

LCDIF_LUT_CTRL field descriptions

Field	Description
31–1 Reserved	This read-only field is reserved and always has the value 0.
0 LUT_BYPASS	Setting this bit will bypass the LUT memory resource completely. No pixel transformations will occur at last stage before output to external displayer.

35.6.22 Lookup Table 0 Index Register (LCDIF_LUT0_ADDR)

Address: 402B_8000h base + B10h offset = 402B_8B10h

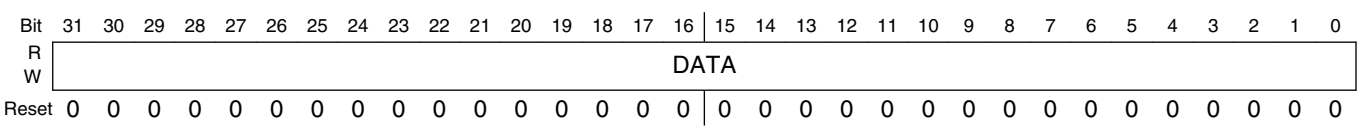


LCDIF_LUT0_ADDR field descriptions

Field	Description
31–8 Reserved	This read-only field is reserved and always has the value 0.
ADDR	LUT indexed address pointer. This address into the LUT memory is always four byte aligned for PIO access. The least two significant bits are not used to drive the LUT memory array. For PIO LUT access, when the LUT data register is written, the contents of the LUT at the address specified by this address field will be loaded with a 32-bit DWORD. This address pointer will be incremented after the LUT data is written. This will provide recursive writes to the LUT data register to initialize the entire LUT array with recursive writes to the LUT data register.

35.6.23 Lookup Table 0 Data Register (LCDIF_LUT0_DATA)

Address: 402B_8000h base + B20h offset = 402B_8B20h

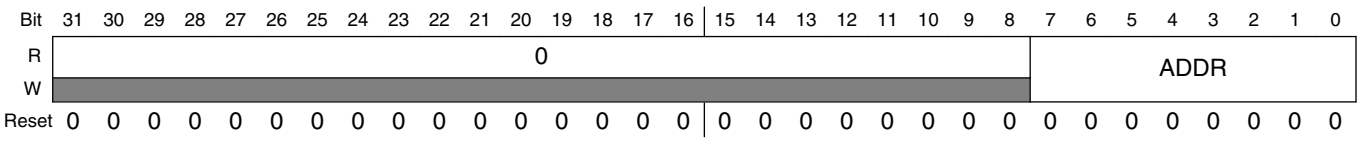


LCDIF_LUT0_DATA field descriptions

Field	Description
DATA	Writing this field will load 4 bytes, aligned to four byte boundaries, of data indexed by the ADDR field of the REG_LUT_CTRL register.

35.6.24 Lookup Table 1 Index Register (LCDIF_LUT1_ADDR)

Address: 402B_8000h base + B30h offset = 402B_8B30h

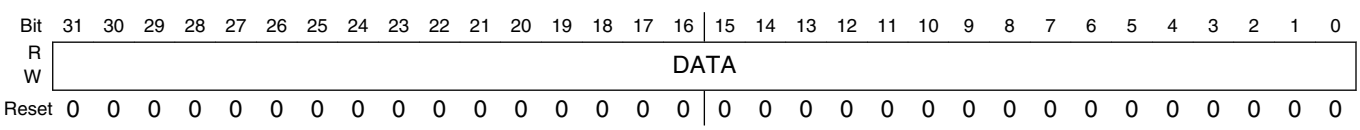


LCDIF_LUT1_ADDR field descriptions

Field	Description
31–8 Reserved	This read-only field is reserved and always has the value 0.
ADDR	LUT indexed address pointer. This address into the LUT memory is always four byte aligned for PIO access. The least two significant bits are not used to drive the LUT memory array. For PIO LUT access, when the LUT data register is written, the contents of the LUT at the address specified by this address field will be loaded with a 32-bit DWORD. This address pointer will be incremented after the LUT data is written. This will provide recursive writes to the LUT data register to initialize the entire LUT array with recursive writes to the LUT data register.

35.6.25 Lookup Table 1 Data Register (LCDIF_LUT1_DATA)

Address: 402B_8000h base + B40h offset = 402B_8B40h



LCDIF_LUT1_DATA field descriptions

Field	Description
DATA	Writing this field will load 4 bytes, aligned to four byte boundaries, of data indexed by the ADDR field of the REG_LUT_CTRL register.

Chapter 36
Pixel Pipeline (PXP)

36.1 Chip-specific PXP information

Table 36-1. Reference links to related information

Topic	Related module(s)	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
Interrupts, DMA Events and XBAR Assignments	-	Interrupts, DMA Events and XBAR Assignments

36.2 Overview

The Pixel Processing Pipeline (PXP) is used to process graphics buffers or composite video and graphics data before sending to an LCD display or TV encoder. It is used to minimize the memory footprint required for the display pipeline and provide an area and performance optimized to both SDRAM-less and SRAM-based systems.

The PXP integrates of several independent processing stages into a cohesive strategy to create flexible pixel pipeline.

36.2.1 Block Diagram

The PXP block diagram is shown below:

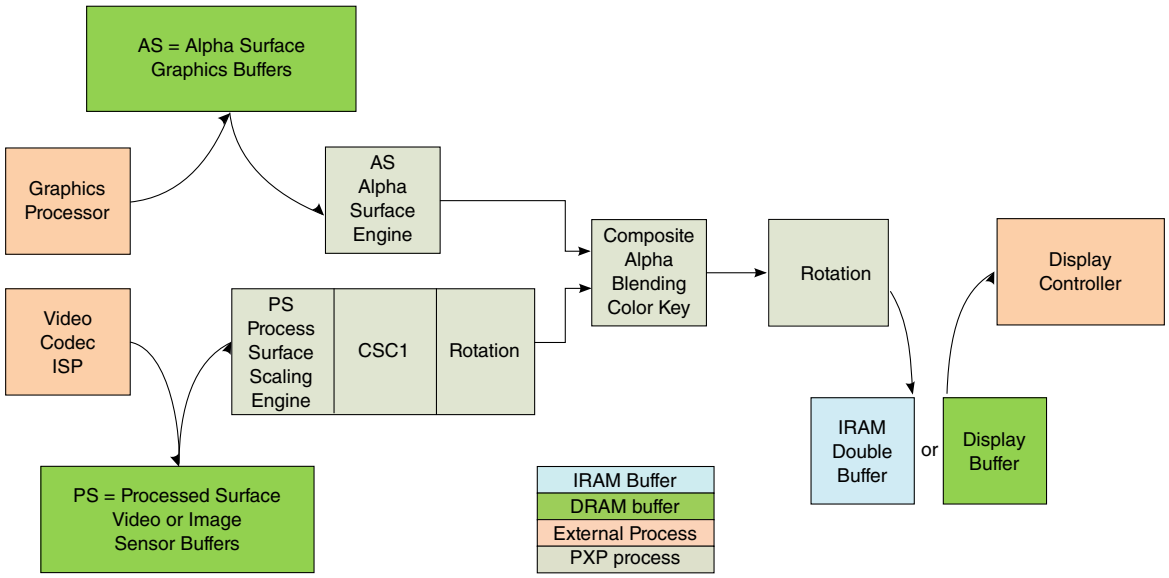


Figure 36-1. PXP Architecture

By integrating multiple blocks, intermediate buffer operations to external memory are removed, reducing external memory bandwidth, power, and software control complexity.

The PXP combines the following into a single processing engine:

- Scaling
- Color Space Conversion (CSC)
- Rotation

36.2.2 Features

The following features are supported in PXP:

- BitBlit
- Flexible image compositions
- Porter-Duff operation
- Image rotation (90°, 180°, 270°)
- Image resize
- Color space conversion
- Multiple pixel format support (RGB, YUV444, YUV422, YUV420, YUV400)
- Standard 2D-DMA operation

36.3 Functional Description

36.3.1 Top-level architecture

The PXP consist of several pipelined blocks that perform the video source frame scaling, color space conversion, alpha-blending/color key algorithm, secondary CSC, pixel correction.

The entire pipeline operate within the requirements of the PXP architecture, and perform operations on either 8x8 or 16x16 pixel blocks in the representative source buffers. The entire pipeline operate within the context of two iteration counters that iterate through the appropriate grid of input blocks to produce the rotated output grid blocks in scan-line order.

Figure 36-1 shows the high-level architecture of the scaling, color space conversion, blending, pixel correction,rotation engines.The Alpha Surface Engine fetches one RGB graphics plane alpha surface (AS). The scaling engine fetches a single processed surface (PS), which can be blended with the AS surface.Although the PXP processes NxN pixel macro blocks, each of the AS or PS surfaces can have any pixel alignment within the output buffer. There are no restrictions and any pixel coordinates within the output buffer are valid. The upper left origin of the output buffer is defined as pixel (0,0). The upper left and lower right coordinates for each of the AS and PS are inclusive within the output buffer.

Figure 36-2 represents a sample output buffer configuration with both an AS and PS included. The alignment of each AS and PS within the output buffer can be at any arbitrary pixel locations. For example, the PS has an upper left coordinate (ULC) of (2,2) and a lower right coordinate (LRC) at pixel (13,13). The maximum value for the ULC and LRC for each of the AS and PS is bounded by the LRC of the output buffer, (15,15) for this example.

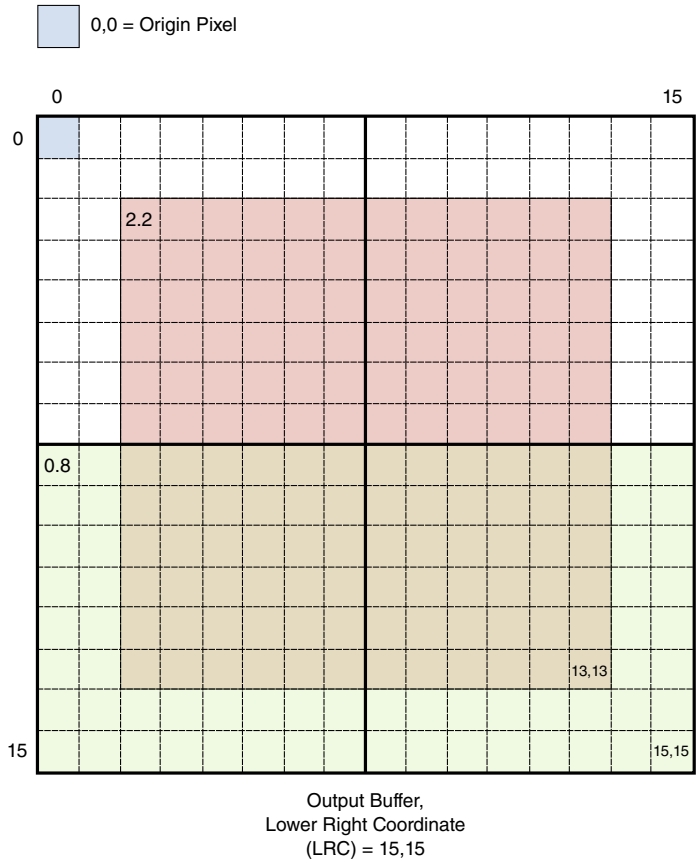


Figure 36-2. Sample output buffer configuration

The AS engine supports RGB pixel formats, and the PS engine supports RGB, YUV, and YCbCr pixel formats. The CSC1 can be used to convert to RGB pixel formats so that the PS surface can be blended with the AS surfaces in the compositing engine in the RGB color space. There is a single rotation engine in the PXP with a programmable location with two possible rotations, one after PS processing, other just before the transfer to display buffer. Rotation can occur after image composition or at the output of the PS engine. In the first scenario, all the data produced by the AS and PS engines is rotated. When the rotation module is programmed to rotate only PS images, the AS is not rotated, and AS pixels are combined with rotated PS surfaces.

36.3.1.1 Processing Details

The PXP architecture has been driven primarily by the requirement that the output buffer must be processed and rotated without intermediate frame buffer stored in external memory.

This reduces the use of external memory bandwidth requirements thus reducing overall system power consumed.

Since the output of the rotation block must be NxN pixel blocks in scan order, the entire pipeline will operate on NxN pixel blocks. In essence, the pipeline will be able to operate on blocks in a random access fashion, but the entire pipeline will operate within the context of two iteration counters that will iterate through the horizontal and vertical input blocks to generate the required output block.

Processing Pipeline

The control block will coordinate the processing of the pixel blocks within the source and destination image buffers. It begins by issuing a command to each stage of the pipeline requesting that operations be done for the block at offset x, y. When the block accepts the command, it asserts its acknowledge signal for a single cycle to indicate the acceptance and allow the control unit to move to the next block.

When the PS and AS fetch engines have received a command, they will fetch the required data and place it into their fetch buffers. If compositing the RGB AS surface with the PS surface, then the output of the PS engine needs to be converted to the RGB color space using CSC1, since all compositing occurs in the RGB color space. For YUV output pixel formats, the CSC1 unit can be enabled to convert pixels into the RGB space for subsequent compositing with AS pixels. If the final output color space is YUV and there is no compositing required (AS not present, for example), then the CSC unit can be bypassed and the pixel data path will pass the YUV pixels to the rotation engine. For YUV output formats, scaling operations, and rotation operations are still valid, but blending RGB AS surfaces with YUV PS surfaces is NOT supported.

NOTE

If input format and output format are both YUV, the CSC1_COFF0[YCBCR_MODE] need be set according to the input format even when CSC1 is bypassed.

The alpha blender/color key module will process a pixel any time that both inputs present valid data.

A handshake will be created between each stage and a pipeline controller to handle the advance of the pipeline and generation of the iteration counters. The pipeline controller will also maintain the interlocks with the LCD interface for the case where the LCD display and pixel processing pipeline use the SRAM to maintain the double buffer block intermediate buffer.

36.3.1.2 Scaling Operation

The scaling engine operates on YUV (or YCbCr) 422 or 420 and any RGB formatted pixels. Each color plane is sourced from color planes indicated by different base address registers.

The scaling source data can be stored as 3 individual planes for each Y, U, and V data, stored as two planes as a single Y and interleaved UV plane, or stored as a single plane with YUV/RGB interleaved on a per byte basis.

The scaled output image is presented to the CSC1 module as YUV444 or RGB888 pixels with a single byte for each color channel. The scaler can reduce an input image by a maximum factor of 16. In this case, the output image will be 1/16 the dimension of the input image in each of the X and Y axis. There are no limits, essentially, on increasing the source image size. The theoretical maximum increase is 4096 since a 12 bit fractional step function is used when scaling an input image. Scaling in either axis, X or Y is independent, so a source image can appear stretched in either direction.

All source images pass through the scale engine. The PXP alpha blend module and AS pixel streams are in the RGB888 format, so PS pixel buffers must be converted to the RGB888 format for alpha blending. The scaling engine works with the CSC1 module to translate YUV/YCbCr pixel formats to RGB888 for output frame buffer compositing using the alpha blender. The CSC1 unit can be bypassed so compositing can occur in the alpha engine.

The scaling operation is divided into two scaling steps. The first step is a decimation scaler, and the second step is a bilinear filter. The decimation filter provide a maximum down scaling factor of 8, and the subsequent bilinear filter provides a maximum scaling factor of 2. Combined, the maximum scaling factor can be up to 16. The decimation and bilinear scaling engines are independently programmable. There is also an initial offset that is programmable to allow more source data to be considered in the bilinear scaling engine.

36.3.1.3 Decimation Image Scaling

The first of two scaling engines is the decimation filter.

The intent of the decimation filter is to use as much source data as is possible to create the output image frame buffer. The decimation filter simply discards certain pixels from the source PS image depending on the reduction selected.

For RGB pixel formats, each color channel is treated equally since there is the same amount of pixel data within each color plane. For YUV422/420 formats, the chroma samples are already subsampled by 2. In these decimation scenarios, the chroma

decimation factor is adjusted to account for the pre-decimation of the chroma samples. For example, since YUV422 is already sub-sampled by 2 horizontally, an X decimation factor of 2 does not apply to the YUV422 pixels in the X direction. All the chroma samples are passed on to the bilinear filter in this case. As another example, an X decimation factor of 4 will decimate the chroma samples by 2, since this factor combined with the pre-decimation factor of 2 in the pixel source buffers totals an overall decimation factor of 4.

The following example will show which pixels (in green) in a source RGB buffer that are passed to the bilinear filter for an X decimation factor of 2 and a Y decimation factor of 4. All pixels coincident with dashed lines are discarded.

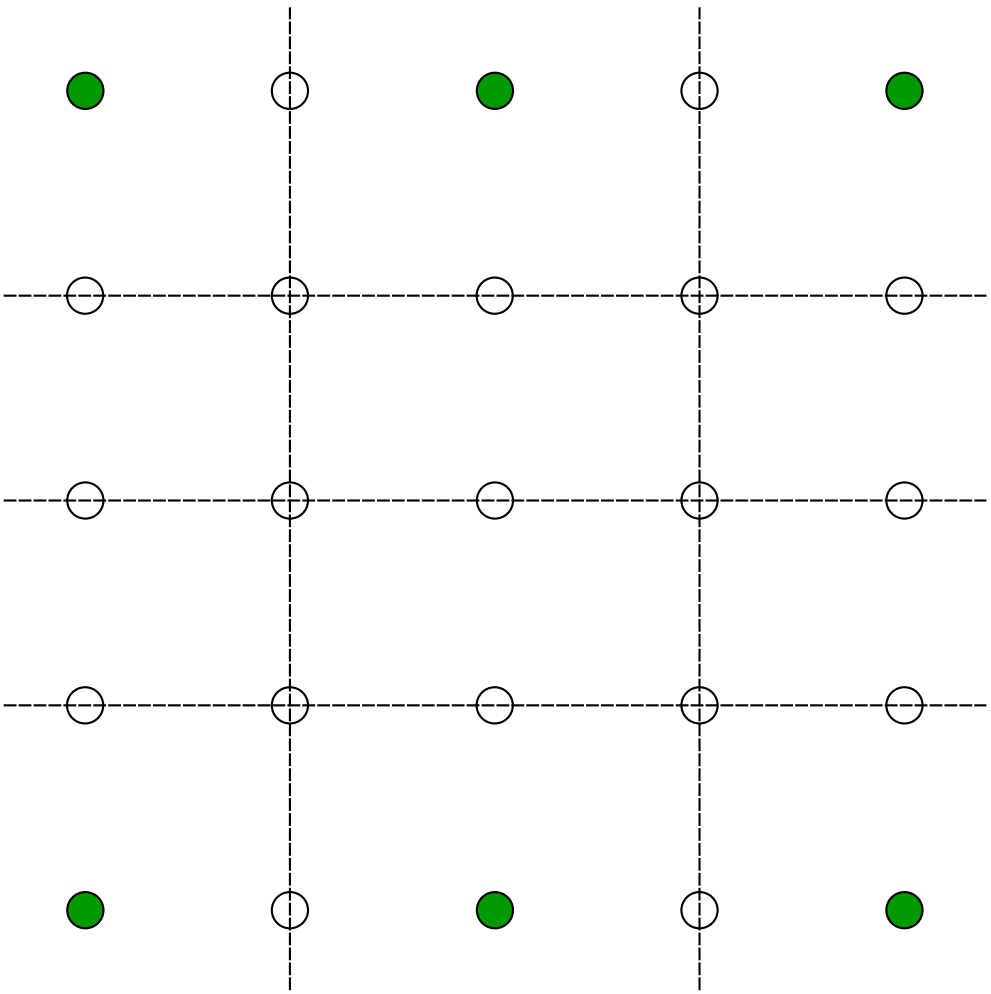


Figure 36-3. RGB decimation X /2, Y /4

Using the same decimation factor as the above scenario for RGB pixels, but using YUV420 source buffers, it can be shown that the decimation factor for the Y and UV components of data are decimated differently. This is due to the pre-decimation of the chroma samples in the source frame buffers. Figure 4: YUV420 decimation X /2, Y /4 indicates that the U/V samples in the X direction are not decimated, but the Y samples in the X direction are decimated by the factor of 2.

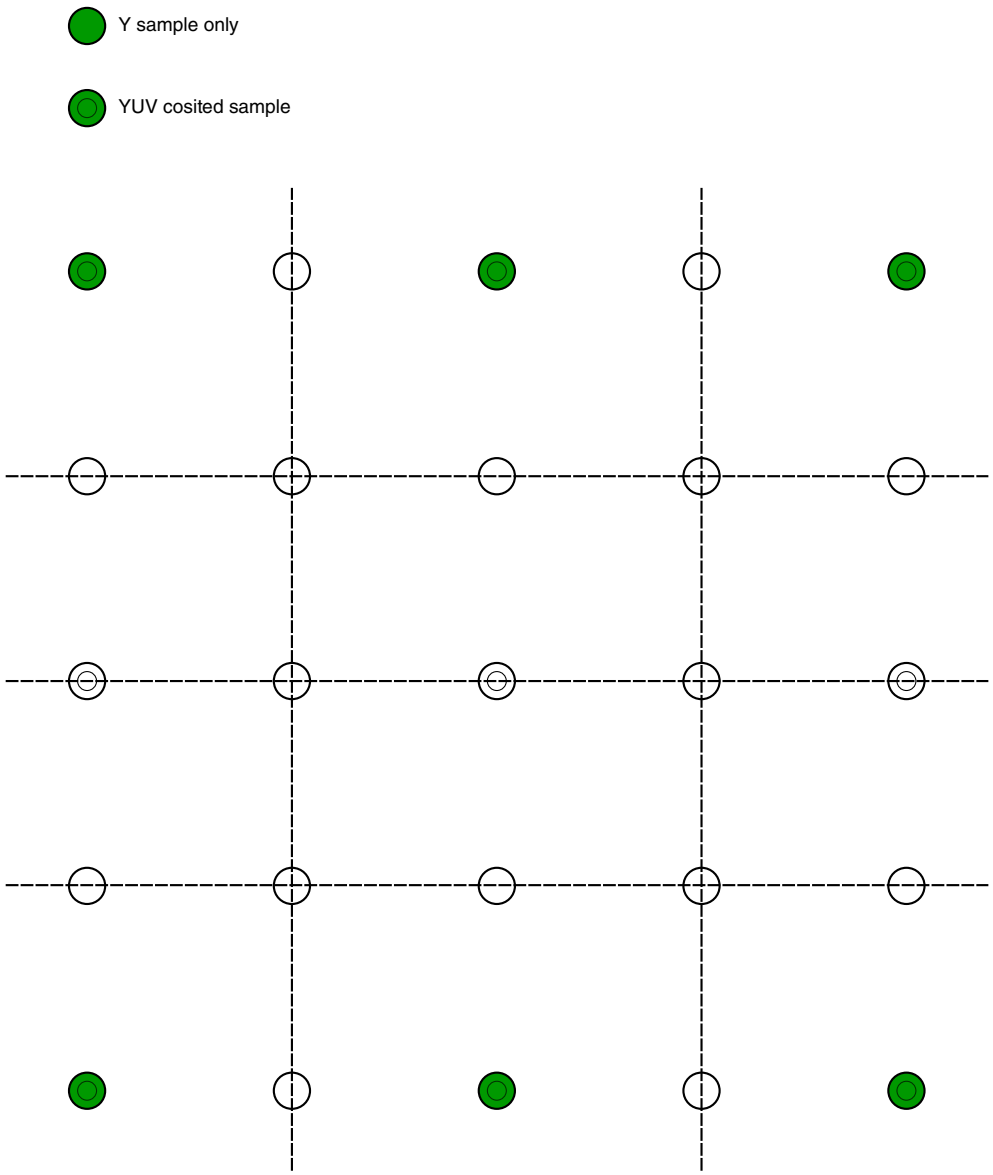


Figure 36-4. YUV420 decimation X /2, Y /4

36.3.1.4 Bilinear Image Scaling Filter

The PXP implements a bilinear scaling filter to resize an input image to a different resolution for display output.

The bilinear filter is a weighted average of the four nearest pixels that can be sourced to approximate the pixel in the output frame buffer.

When scaling YUV data, the UV values are offset by 0x80 (top bit inverted) to shift the signed UV bits into an unsigned equivalent with a range of 0 to 255. YCbCr data does not have to be shifted since it is defined as an unsigned byte. The REG_CSC1_COEF0[YCBCR_MODE] bit controls whether this operation is applied to the input UV bytes.

After scaling, the offset is removed so that the range for UV data is signed from -128 to 127.

The reason for this adjustment is based on the implementation of an unsigned scaling engine, and therefore, is to ensure that the scaled values are handled properly. Consider the following table:

Format	pixel0	pixel1	average	Result
decimal	-2	+2	0	Correct
CbCr	0x7E	0x82	0x80	Correct (0x80 is 0 in CbCr)
UV	0xFE	0x02	0x80	Incorrect (0x80 is -128 in UV)
decimal	-32	+16	-8	Correct
CbCr	0x60	0x90	0x78	Correct (0x78 is -8 in CbCr)
UV	0xE0	0x10	0x78	Incorrect (0x78 is +120 in UV)

To compute the output pixel value at position as indicated by P, consider the diagram below.

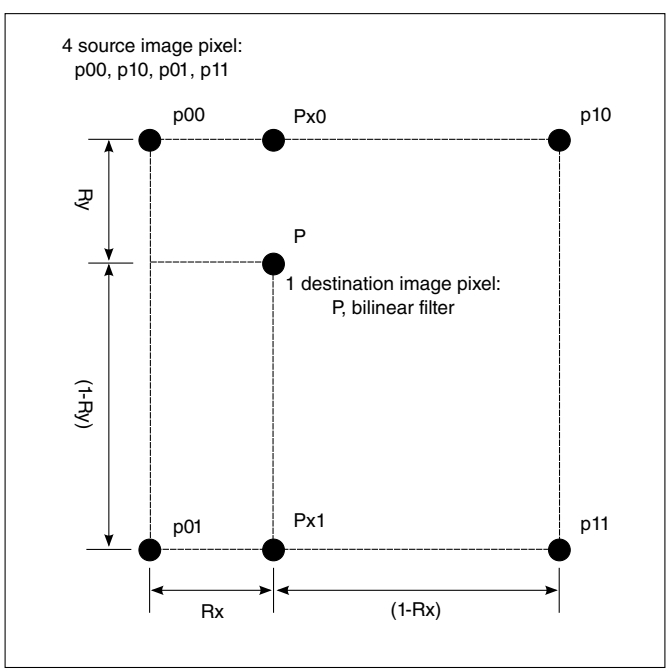


Figure 36-5. Output Pixel Value

A step function is used to indicate the position of the pixel "P" in the output frame. This position may not coincide with a single pixel position in the input frame buffer. In this case, the four closest pixels in the input frame are used to approximate the value of the pixel in the output frame.

The PXP scaler first computes a linear filter in the X axis to create the two intermediate pixel values Px0 and Px1. The step function's X fractional component is used to provide the weighting factor for blending p00 with p10 to provide Px0. Likewise, Px1 is also derived from a linear filter using p01 and p11.

The equations for Px0 and Px1 are as follows:

Px0 = p00*(1-Rx) + p10*Rx

Px1 = p01*(1-Rx) + p11*Rx

The PXP scaler uses the intermediate X pixels Px0 and Px1 and implements a bilinear filter on these two pixel values to produce the final pixel value at position P. The remainder of the step function for the Y axis is used to compute the weighted average pixel result. The equation for final filtered pixel is:

P = Px0*(1-Ry) + Px1*Ry

36.3.1.5 YUV 4:2:2 Image Scaling

The following figure illustrates the positioning of YUV samples for the 4:2:2 formats. There are twice as many Y luma samples then U and V chroma samples horizontally.

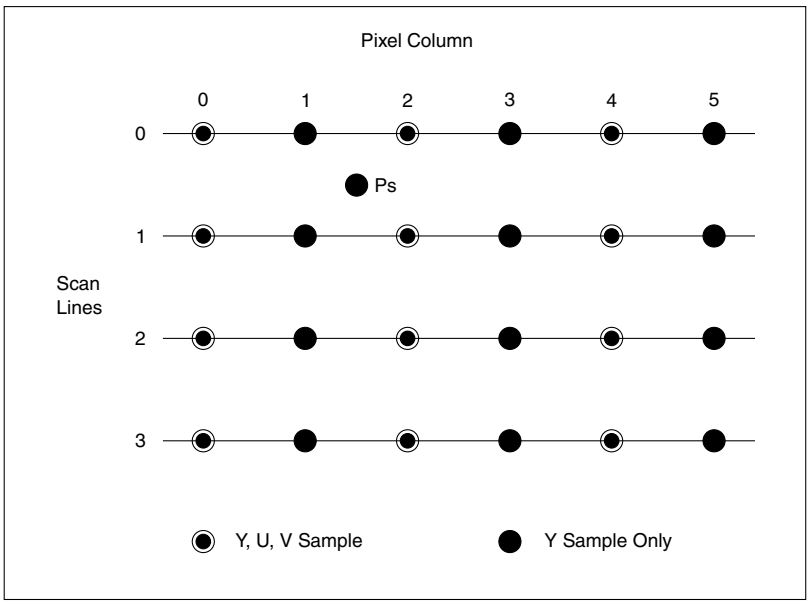


Figure 36-6. YUV Sample Positioning, 4:2:2

Consider the scaled output pixel Ps (pixel scaled) which has an accumulated step function of X=1.5 and Y=0.5. The remainder for the step function is Rx = 0.5 and Ry = 0.5. Or, the sub pixel position of output pixel Ps is half way between line 0 and 1 and half way between column 1 and 2.

The Y output component of Ps is simply the bilinear function of the four nearest Y samples from the input image. Specifically, the Y values at [1,0], [2,0], [1,1], and [2,1] are used to compute the Y for Ps.

For the U and V components of Ps, there are no samples present in the column position 1. The bilinear filter uses chroma components located at [0,0], [2,0], [0,1] and [2,1]. Since the chroma components are not sub sampled vertically, the remainder used to combine pixels vertically is Ry=0.5 (the same as for Y). However, horizontally, the scaling engine shifts the remainder by a factor of 2. So an X axis step function value of X=1.5 has a remainder Rx=0.75. Source chroma values are not replicated, they are completely interpolated using the four nearest chroma samples to approximate U and V at Ps.

36.3.1.6 YUV 4:2:0 Image Scaling

The following figure illustrates the positioning of YUV samples for the 4:2:0 formats. Chroma is sub sampled both horizontally and vertically. In this format, the chroma frame buffers contain ¼ the data that the luma frame buffers store.

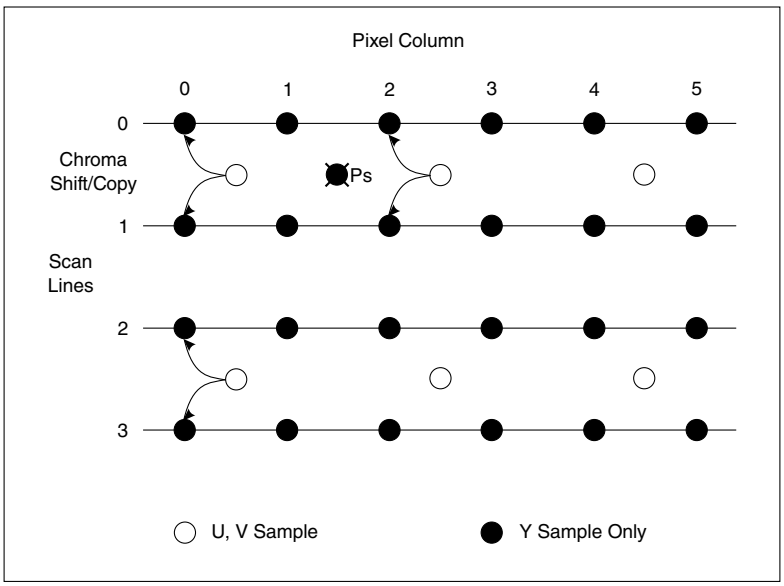


Figure 36-7. YUV Sample Positioning, 4:2:0

The Y output component for all scaled pixels in 4:2:0 formats are the same as for the 4:2:2 pixel formats.

The U and V output components have two considerations when computing the output pixel Ps.

1. All chroma samples from the input source image are shifted left and up by ½ a sample position of the input pixel matrix.
2. Odd scan lines are replicated using the previous even chroma scan line values. So, output image chroma values that map between even to odd scan lines are replicated in the vertical axis. In contrast, output image chroma values between odd to even scan lines are interpolated vertically.

The chroma values are interpolated horizontally as in the 4:2:2 pixel format.

As an example, consider the interpolated pixel Ps in the 4:2:0 diagram above. For the Y component, the interpolated output luma is a function of the Y values in the source frame buffer at position [1,0], [2,0], [1,1], [2,1].

For the U and V interpolated samples, the chroma values on scan line position 0.5 are shifted so that they coincide with the even luma sample points. They are also replicated so that a single chroma scan line is used twice. The chroma scan line at 0.5 is replicated to represent the 4:2:2 sample points for scan line 0 and 1. The chroma scan line at 2.5 is replicated to represent the 4:2:2 sample points for scan line 2 and 3. This pattern of chroma replication occurs for the entire source frame buffer during the scaling operation.

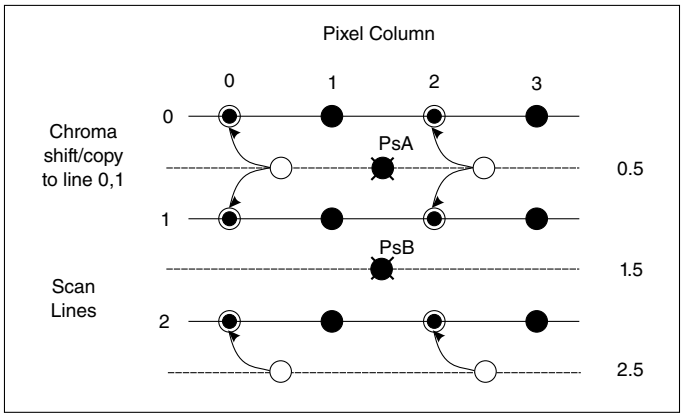


Figure 36-8. Scaled Chroma Computation Examples

The preceding diagram has two examples for the computation of the scaled chroma output pixel. For chroma at output position PsA (vertical position 0.5), interpolation occurs in the X axis using chroma values at column 0 and column 2. However, since line 0 and line 1 have equal chroma values due to chroma line replication, scaling in the Y axis results in replication of chroma values.

For chroma at output position PsB (vertical position 1.5), interpolation occurs in both the X and Y axis. The Y axis is an interpolation since the chroma values copied to scan line 1 and 2 and not the same.

In summary, any output image pixels that map to an odd scan line above and an even scan line below are interpolated vertically. Output image pixels that map to an even scan line above and an odd scan line below are replicated vertically.

36.3.1.7 RGB/YUV444 Image Scaling

For all RGB formats, the RGB pixels are converted up to RGB888 with 8 bits per each color component.

Then each color component is passed to the scaling engine and each component is treated in the same manor. The RGB scaling operation is the same as for the Y scaling operation described in the preceding sections. Also, YUV444 contains a byte for each color plane at each pixel location, so all three color components are scaled in the same manner.

36.3.1.8 Color Space Conversion (CSC)

The CSC unit converts pixels between color spaces. The CSC1 unit is right after the scaling unit and is dedicated to converting from YUV to RGB. All coefficients are programmed as two's compliment numbers and CSC unit can be bypassed if conversion is not desired in the pixel data path.

36.3.1.9 CSC1 Operation

The CSC1 module receives scaled YUV/YCbCr444 pixels from the scale engine and converts the pixels to the RGB888 color space only if CSC1 is enabled.

The CSC1 module will convert only to the RGB color space and it can be bypassed to allow YUV pixels through the data path. These pixels are loaded into the pixel FIFO for processing by subsequent modules in the pixel data path.

The following equations are used to perform YUV/YCbCr -> RGB conversion. The constants will be stored in the PXP control registers as two's compliment values to allow flexibility in the implementation and to allow for differences in the video encode and decode operations. In addition, this provides a software mechanism to manipulate brightness or contrast.

R = C0(Y+Yoffset) + C1(V+UVoffset)

G = C0(Y+Yoffset) + C3(U+UVoffset) + C2(V+UVoffset)

B = C0(Y+Yoffset) + C4(U+UVoffset)

Note: In the equations above, U and V are synonymous with Cb and Cr in regards to the color space format of the source frame buffer.

Saturation of each color channel is checked and corrected for excursions outside the nominal YUV/YCbCr color spaces. Overflow for the three channels are saturated at 0x255 and underflow is saturated at 0x00.

The table below indicates the expected coefficients for YUV and YCbCr modes of operation:

Coefficient	YUV	YCbCr
Yoffset	0x000	0x1F0 (-16)
UVoffset	0x000	0x180 (-128)
C0	0x100 (1.00)	0x12A (1.164)
C1	0x123 (1.140)	0x198 (1.596)
C2	0x76B (-0.581)	0x730 (-0.813)
C3	0x79B (-0.394)	0x79C (-0.392)

Table continues on the next page...

C4	0x208 (2.032)	0x204 (2.017)

36.3.1.10 YUV versus YCbCr Support

By default, the PXP color space coefficients are set to support the conversion of YUV data to RGB data.

If YCbCr input is present, software must change the coefficient registers appropriately (see the register definitions for values). Software must also set the YCBCR_MODE bit in the COEFF0 register to ensure proper conversion of YUV versus YCBCR data.

36.3.1.11 Alpha Blending/Color Key

Regardless of pixel input format, the PS and AS pixels are normalized to 32-bits, organized as one alpha and three data bytes. Alpha blending occurs in the RGB space, if blending is required, PS pixels should be converted to RGB space. If no alpha blending is required, then YUV pixels can bypass the alpha blending ALU without color space conversion. All pixels are processed by the pixel ALU, but the ALU operations can be disabled to achieve pixel pass through for either PS or AS source pixels.

36.3.1.12 Alpha Blend

The alpha value for an individual pixel represents a mathematical weighting factor applied to the AS pixel. An alpha value of 0x00 corresponds to a transparent pixel and a value of 0xFF corresponds to an opaque pixel.

The effective alpha value for an AS pixel is determined by the AS_CTRL[ALPHA] and AS_CTRL[ALPHA_CTRL] register fields. If AS_CTRL[ALPHA_CTRL] = ALPHA_OVERRIDE , the alpha value for the pixel is taken from the AS_CTRL[ALPHA] . This can be useful for applying a constant alpha to an entire image or for image formats that don't include an alpha value. If AS_CTRL[ALPHA_CTRL] = ALPHA_MULTIPLY, the pixel's alpha value will be multiplied by the pixel's ALPHA value in order to allow scaling of the pixel's alpha or to provide better control for pixel formats such as ARGB1555, which only contains a single bit of alpha.

For each color channel, the equation used to blend two source pixels is defined below:

Gá = PIO programmed global alpha (8-bit value).
Eá = Embedded alpha associated with AS pixel.
á = (Gá * Eá + 0x80)/128, and á doesn't exceed a value of 128

The result for the red channel as an example:

R[7:0] = (á * PS.r) + ((1 - á) * AS.r)

When á is 0xff, the PS pixel will not be blended with the AS pixel, but PS will be passed as the output pixel and will not be blended with AS. In this case, AS will be discarded. Likewise, if á is 0x00 for a given pixel, AS will be loaded as the output pixel.

AS_CTRL[ALPHA_INVERT] provides the option to invert the final alpha value. This essentially inverts the effect the alpha value has on the AS and PS blending operation.

36.3.1.12.1 Porter-Duff Alpha Blend

Porter-Duff blend includes 12 blending modes to describe digital image composite. These processes include:

- Clear
- Source Only
- Destination Only
- Source Over
- Source In
- Source Out
- Source Atop
- Destination Over
- Destination In
- Destination Out
- Destination Atop
- XOR

Through these processes it can achieve any 2D image composite.

To control the blending modes, please see the following picture. Please see PORTER_DUFF_CTRL register for more details.

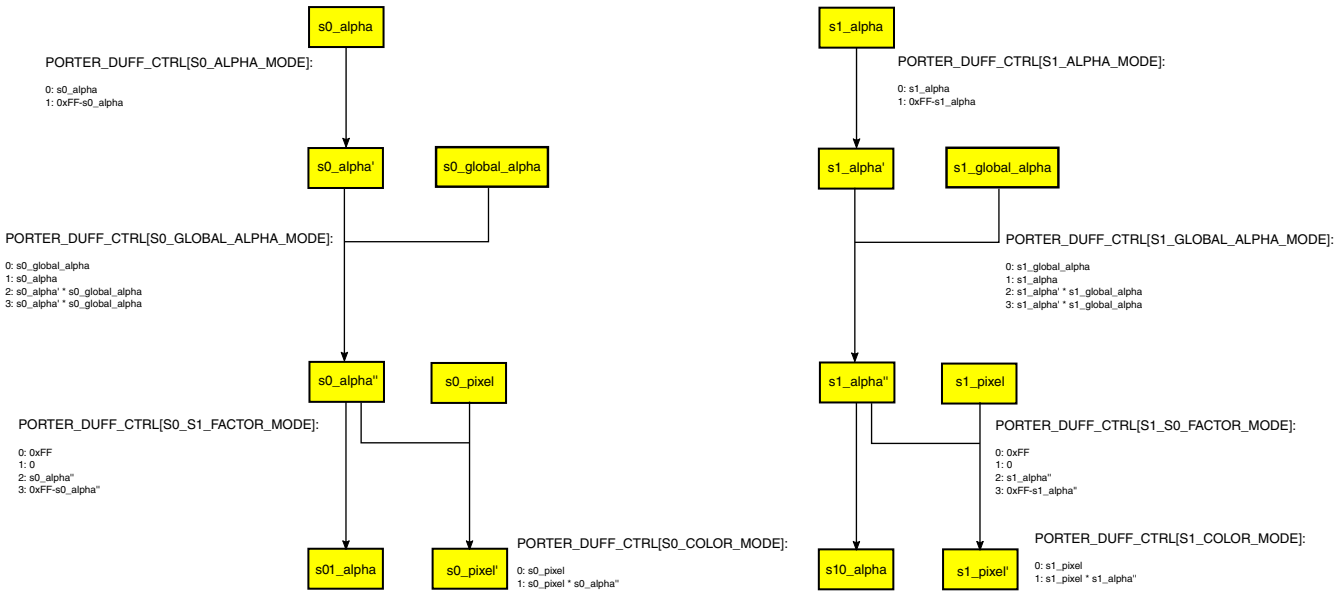


Figure 36-9. Porter-Duff Alpha Blend

image output = s01_alpha × s1_pixel' + s10_alpha × s0_pixel'

alpha output = s10_alpha × s0_alpha'' + s01_alpha × s1_alpha''

NOTE

s0 refer to PS and s1 refer to AS.

36.3.1.13 Color Key

The color key function is provided to create transparent effects on the output pixel.

Color keying is applied on the input pixels after they are converted to 8-bits for each red, green, and blue color channels (color keys are not applied directly to 16-bit pixel formats but to their corresponding 24-bit representation). A color key range is programmable for both PS and AS pixels. If the PS 24-bit pixel is within the PS color key range, then AS is passed through the pixel pipeline. In this case, alpha blending does NOT occur. Conversely, if AS is within AS color key range, then AS is passed via the PXP data pipeline. If both PS and AS color key tests pass, then the back ground color register is passed onto following PXP processing components in the pipeline.

The condition for color keying to be satisfied is:

CK0.r.low <= PS.r <= CK0.r.high

CK0.g.low <= PS.g <= CK0.g.high

CK0.b.low <= PS.b <= CK0.b.high

For example, if the "red" 8-bit value for the PS pixel (or PS.r) is between the color key low and high values (CK0.r.l and CK0.r.h), the condition is true for the red color plane. When All three color planes meet this condition, PS pixel is invisible in output register.

To disable color keying, program the low color key register value to 0xff and the high value to 0x00. This will guarantee that the color key range test will never be true.

36.3.1.14 Rotation

There is a single rotation resource integrated into the PXP. The location of this resource within the PXP data path is programmable. Rotation can occur after compositing the AS and PS buffers in the output stage.

As an alternative configuration, the PS buffer can be rotated and later composited with the AS surface that is not rotated. There is a single configuration bit that provides the configuration of where rotation is implemented within the PXP.

To rotate graphics, the hardware must read pixels in one direction across a frame buffer and write them in a alternate orientation. For the 90 and 270 degree cases, this means that lines of pixels must either be read or written vertically in a frame buffer.

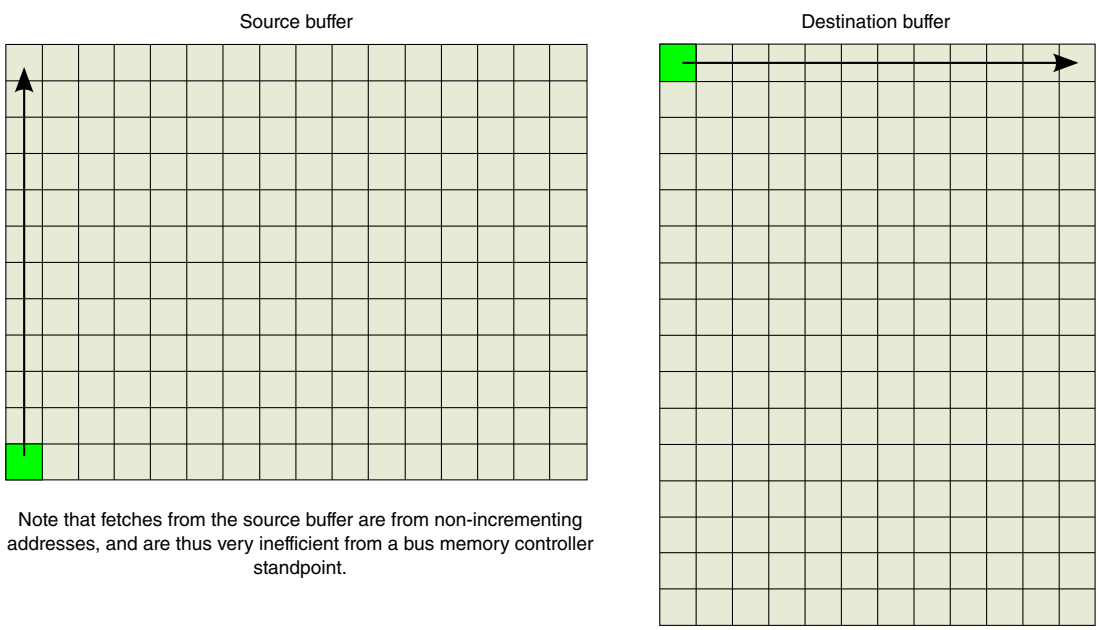


Figure 36-10. Rotation Read and Write

In order to rotate efficiently, multiple columns must be rotated to enable the engine to both fetch and store bursts of pixels, thus improving memory performance. The simplest method of doing this is to operate on square blocks of pixels. To rotate the image, each sub-block of pixels must be rotated by the required rotation angle.

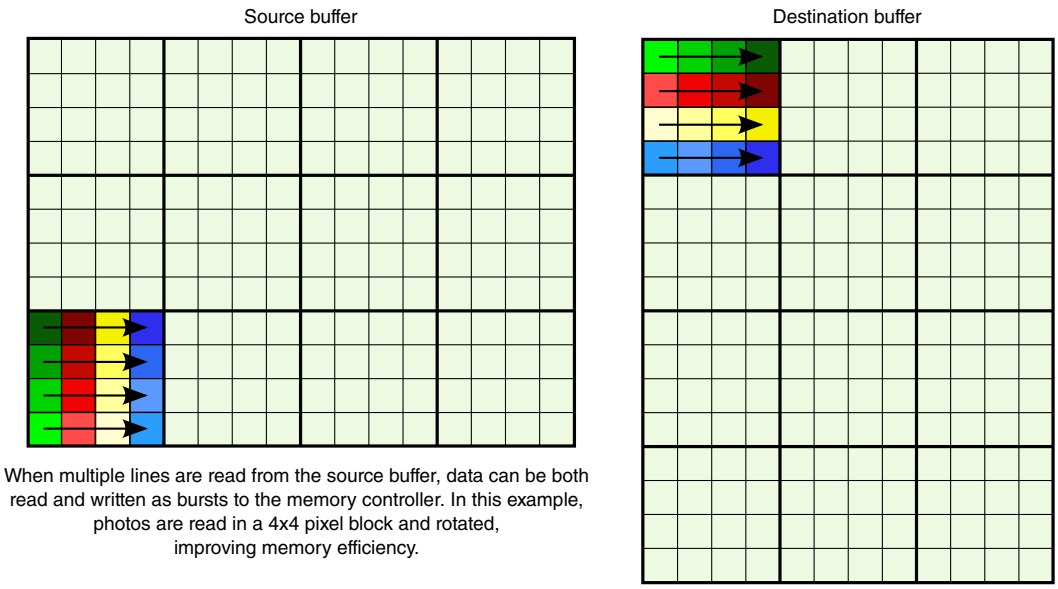


Figure 36-11. Rotated Sub-blocks

To manage the rotation process, the source image can be broken into a grid of sub-blocks that have coordinates as shown in the diagram below. In addition to rotating the sub-block, each block must be translated to a new coordinate location. For each of the rotation angles (0, 90, 180, 270), it is possible to define a simple algorithm for computing the new translated grid address. The hardware must then simply compute the memory address from the base grid address for both load and store operations.

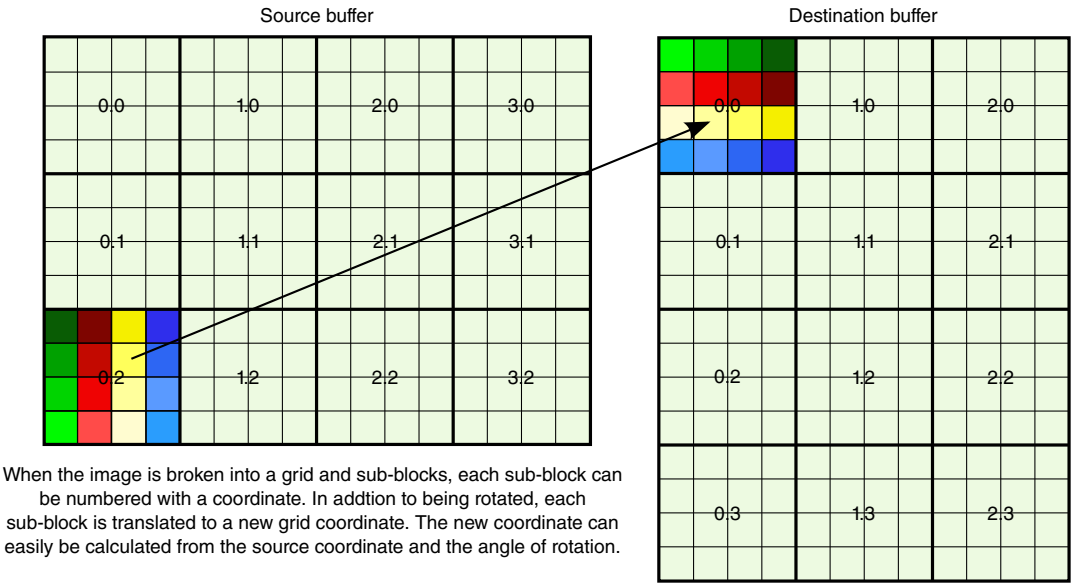


Figure 36-12. Grid of Sub-Blocks with Coordinates

In order to balance the requirements of reasonable burst sizes to the memory controller as well as keep the hardware storage requirements to a minimum, the blending/rotation engine will operate on either 8x8 or 16x16 pixel blocks. When using the Rotate engine with the input fetch engine, you need to program the input fetch engine to work in 8x8 block mode.

NOTE

An important artifact of the PXP is when rotating a source image and the output is NOT divisible by the block size selected. The output engine essentially truncates any output pixels after the desired number of pixels has been written. Since the output buffer is written as a horizontal row of blocks, the incorrect pixels could be truncated and the final output image can look shifted. In the case where the block size is programmed to 8x8, and the output size that is programmed is 12x12, then there is a remainder of 4 pixels that will be truncated in either the X and/or Y axis when the PXP operation is complete. The output will be shifted by 4 pixels in this example. To compensate for this, the source base address needs to be adjusted so the correct pixels get truncated and the image does not look shifted. In this example, with 90 degrees of rotation, the PS base address should be adjusted by 4 times the actual PS base address -(4*pitch).

36.3.1.15 Output Buffer

The output buffer engine accepts data from the PXP pixel pipeline and issues requests to transfer the output pixels to external DRAM or the internal SRAM double buffer row of blocks.

36.3.1.16 Address calculator

Each of the blocks will manage its own fetch address using a common address calculator block that computes real addresses from a base address and relative block offset from the base.

Each block will then perform the multiple line fetches (or stores) required to perform the operation. This hides all the address buffer computations from the processing blocks and allows each block to simply track the coordinate of the block it is working on.

36.3.1.17 Block size selection

The PXP can be configured to process blocks that are either 8x8 pixels or 16x16 pixels with the REG_CTRL[BLOCK_SIZE] control bit.

When selecting a 16x16 pixel block size, the accesses to fetch AS and PS images and write the final frame buffer are more efficient since twice as much data is requested and processed per memory request.

When optimizing the system for memory bandwidth and image processing time, configure the PXP to process 16x16 pixel blocks.

36.3.1.18 Interlaced Video Support

The PXP has some minimal ability to generate interlaced video content from a progressive source. There two available options, based on the bandwidth requirements and how software is managing video frames.

The PXP can either interlace on the input side (by reading every other line of input data) or on the output side (by writing the individual lines of video into two separate fields). Generally, output interleaving should be used since it is the most flexible mode (it allows scaling and full overlay support) and it only requires a single pass of the PXP to generate two separate output fields.

Input interleaving can be beneficial in cases where the PXP is running at 60fps, since it requires fewer fetches to produce the output data. There is no direct hardware support for input interleaving, in that, there is no configuration bit that can be set to alter how the PXP processes a frame for input interleaving. Input interleaving is achieved by simply setting the source frame buffer pitch value to twice the value it would normally be set to for the equivalent progressive frame. The output parameters also need to be consistent with the desired processing effect. For example, the vertical resolution would be set to account for the reduced resolution to process the interlaced input buffers.

36.3.1.19 LCDIF Handshake

The PXP and LCDIF support a mode where the internal SRAM can be used for the frame buffer to minimize external memory bandwidth required.

This is accomplished by creating two buffers in SRAM, a double buffer row of blocks, where each correspond to 8/16-lines of the frame buffer. The buffers must be consecutive and allocated as a single block of data.

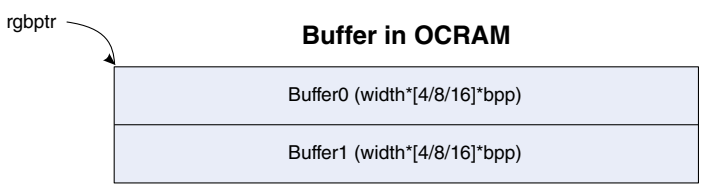


Figure 36-13. Buffer in OCRAM

The storage required can be calculated for an 8x8 block size as

- storage = 16 (lines) * rotated_row_length * pixel_size

and for 16x16 block size as

- storage = 32 (lines) * rotated_row_length * pixel_size

where pixel_size = 4 for 32bpp or 2 for 16bpp modes. The following table lists the storage requirements for common image sizes using 8x8 block size:

Image Size	Storage (16bpp)	Storage (24bpp)	Storage (32bpp)
320x240 (QVGA) - 0/180 rotation	10KB	15KB	20KB
320x240 (QVGA) - 90/270 rotation	7.5KB	11.5KB	15KB
640x480 (VGA) - 0/180 rotation	20KB	30KB	40KB

Table continues on the next page...

640x480 (VGA) - 90/270 rotation	15KB	22.5KB	30KB

The following diagram shows how the minimal rotation buffer would be organized. As the engine and LCD progress down the image, they continually swap roles of filling and emptying each eight-line buffer.

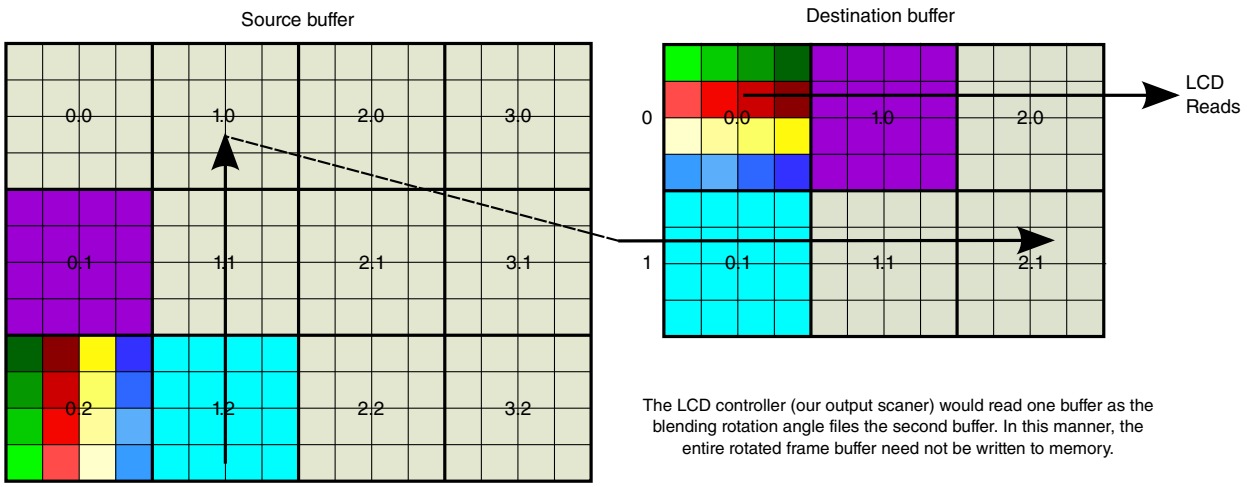


Figure 36-14. Minimal Rotation Buffer Organization

When this mode is enabled, the PXP will process one row of pixel blocks and write the results to the first SRAM buffer (buffer 0). The PXP will then alternate between writing subsequent rows to buffer 0 and buffer 1. After the PXP generates the data for one buffer, the LCDIF will begin reading that buffer and send the contents to the display device. After the LCDIF finishes reading a buffer, it will start displaying from the other buffer while the PXP continues filling the previously processed buffer.

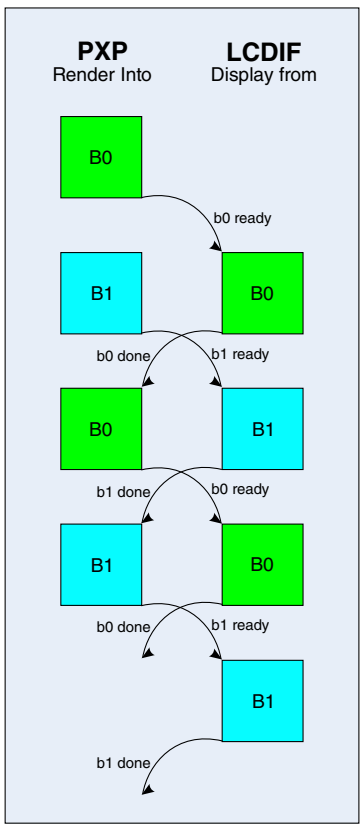


Figure 36-15. PXP and LCDIF Buffer Sharing

To accomplish the buffer sharing, the PXP and LCDIF will maintain buffer status using a pair of handshake signals. When a buffer is filled by the PXP, it will assert the pxp_lcdif_bx_ready (where x is 0 or 1) signal to indicate to the LCDIF that the buffer has valid data. The LCDIF will then release the buffer by asserting the lcdif_pxp_bx_done signal.

The basic protocol is shown in the diagram below:

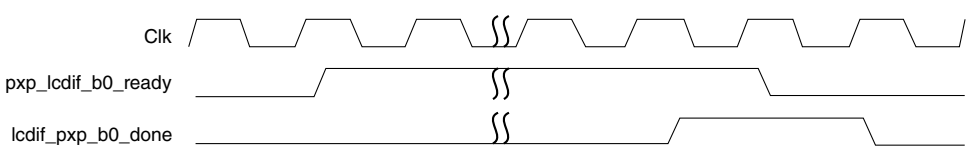


Figure 36-16. Buffer Sharing Protocol

The PXP will continue to assert the bx_ready signal until the corresponding bx_done signal is sampled high for one clock cycle. It will then deassert the bn_ready until the next time the buffer has been filled. After the PXP samples the bx_done signal asserted, it is free to begin filling the buffer with the next block size lines of display data. If a buffer has not been released when the PXP is ready to process data for that buffer, it will suspend rendering operations until the buffer has been released by the LCDIF.

36.3.1.20 LCDIF Abort

When the memory subsystem is not loaded, the PXP should be able to render the buffers faster than the LCDIF can drain the buffers.

It is possible under some scenarios (high LCDIF output rates with high memory latency) that the PXP may not be able to keep up with the LCDIF, even in the SRAM mode of operation. When this happens, the LCDIF will signal that it has completed one of the buffers before the other has been rendered by the PXP. This condition will be detected by the PXP's control logic as an "LCDIF Abort", which will cause the PXP to abort processing in the current row and proceed to the following row. It will acknowledge the abort to the LCDIF by raising the buffer_ready signal for the current buffer to enable the LCDIF to begin displaying the partially-filled buffer. While an abort will create artifacts in the video display, it does minimize the artifacts by limiting them to the remaining pixels blocks in the current row versus ruining the entire frame buffer.

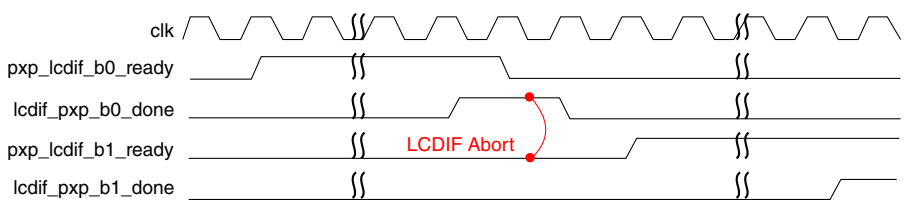


Figure 36-17. LCDIF Abort

36.3.1.21 Theory of Operation

The PXP can be used to accelerate graphics operations by offloading graphics processing from the processor. The block can perform alpha blending and color key substitution on two RGB graphics buffers.

The PXP is organized as having a processed surface (PS) and an alpha surface (AS) that can be blended with the processed surface. There are no restrictions on the location of the AS or PS within the output surface (OS). As the PXP processes NxN blocks, operations

are performed on a pixel by pixel basis. The AS and PS pixels are alpha blended, color keyed, process by CSC resources as individual pixel components. This allows efficient block processing with supporting arbitrary alignment for both the AS and PS surfaces. The resulting pixel block is then written to the corresponding block in the output buffer.

36.3.1.22 Pixel Handling

All pixels are internally represented as 32-bit values regardless of input or output pixel formats. The pixels get converted in the AS and PS buffer engines to 24-bit pixel values. There is also an 8-bit alpha value at stages up to the alpha blender within the PXP for blending within the RGB color space. Compositing of AS and PS images can only occur in the RGB color space. If compositing is not required, then YUV pixels can be transferred and processed at all PXP pixel resource components.. The color orientation of pixels within the PXP can be controlled by the CSC1 resource.

For RGB, input pixels are converted into 32-bit pixel values using the following rules for both AS and PS:

1. 32-bit ARGB8888 pixels are read directly with no conversion.
2. 32-bit RGB888 pixels are assumed to have an alpha value of 0xFF (full opaque).
3. 6-bit RGB565 and RGB555 values are expanded into the corresponding 24-bit color space and assigned an alpha value of 0xFF (opaque). The expansion process replicates the upper pixel bits into the lower pixel bits (for instance a 16-bit RGB555 triplet of 0x1F/0x10/0x07 would be expanded to 0xFF/0x84/0x39).
4. 16-bit RGB1555 values are expanded into the corresponding 24-bit color space and assigned an alpha value of either 0x00 or 0xFF, based on the 1-bit alpha value in the pixel. The ALPHA_MULTIPLY function is useful in this scenario to allow scaling of the opaque pixels to a semi-transparent value.

Alpha values can be passed through the entire PXP data path and output in ARGB8888 or ARGB1555 pixel modes. Also, output pixels can be assigned an alpha value using the OUT_CTRL[ALPHA] register. 16-bit pixels values are formed from the most significant bits of the 24-bit pixel values.

When YUV/YCbCr output formats are selected, all pixels are internally represented as either RGB or YUV pixels values.

36.3.1.23 Output Buffer Composition

The output buffer will be rendered by composing each pixel block from the associated PS and AS buffers.

The AS pixel buffer can be blended or color-keyed with the associated data from the PS buffer (either the PS image pixels or PS_BACKGROUND register based on PS programmed coordinates).

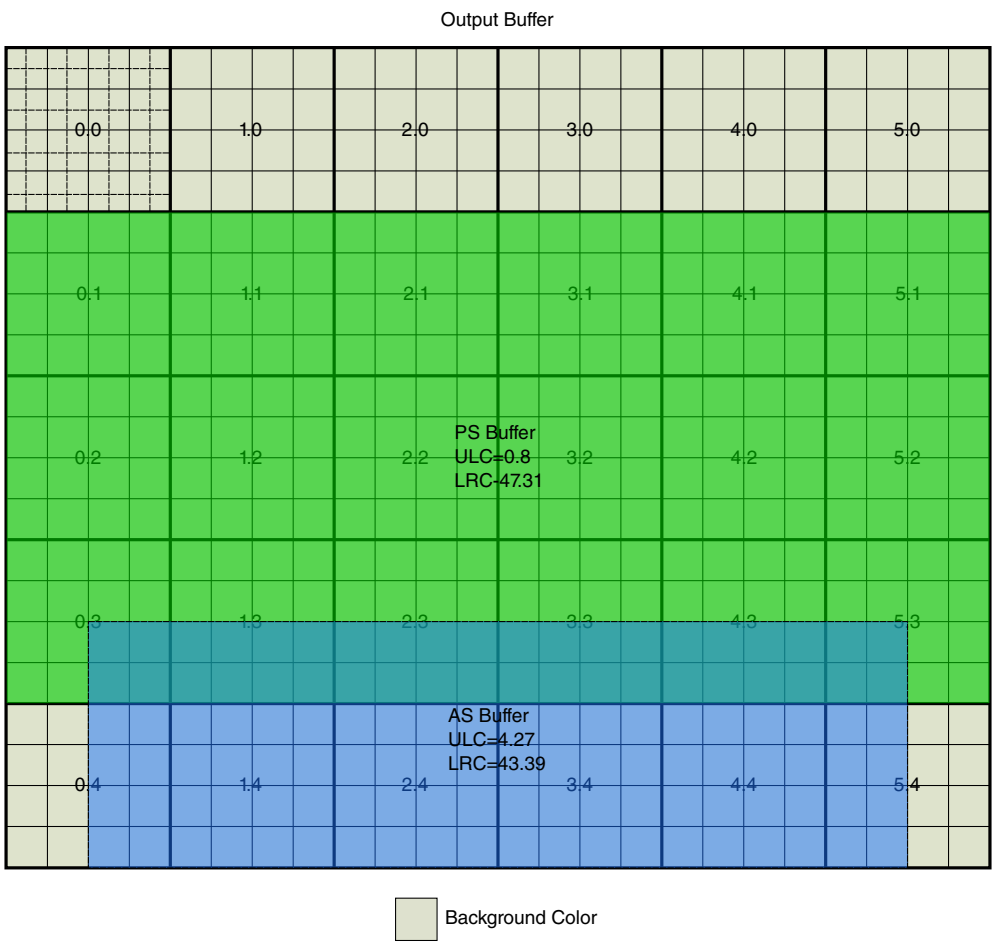


Figure 36-18. Output Buffer Composition

36.3.1.24 PS Image Processing

As the PXP processes image buffers, it iterates over the output buffer by fetching the corresponding input buffer blocks and processing the pixels embedded in these.

36.3.1.25 Letterboxing

At each pixel coordinate, the control logic determines if the PS pixel (argument also applies to AS pixels) will be used in rendering the output pixel.

This is determined by checking the output pixel's coordinates against the REG_OUT_PS_ULC and REG_OUT_PS_LRC (ULC and LRC in short) register contents. For pixels outside this region, the PS pixel will be loaded with the pixel value from REG_PS_BACKGROUND, which can be used to effectively control the letterboxing color. There are no block size or block boundary restrictions when setting the ULC or LRC for either the AS or PS. The only restriction is that the ULC and LRC are within the OUT LRC extents.

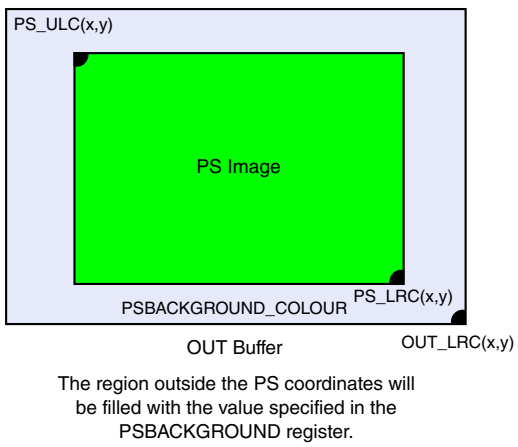


Figure 36-19. OUT Buffer

36.3.1.26 Clipping source images

A subset of the PS buffer can be used in rendering the output buffer. The PXP_PS_BUF register can indicate an offset into the PS buffer that will be used for display within the OUTPUT buffer.

The pixel at the address defined in the PXP_PS_BUF register will be the pixel that is displayed at the pixel coordinate indicated by PXP_OUT_PS_ULC within the output buffer. Essentially, the PXP_PS_BUF register can be used to establish an offset into the PS buffer thus clipping all PS buffer pixels that are at a lower address. The PXP_PS_PITCH will always indicate the number of bytes that are vertically adjacent in the PS buffer. The settings in the PXP_PS_BUF, PXP_OUT_PS_ULC, and PXP_OUT_PS_LRC will determine the subset of the PS buffer, or clipped PS source buffer, that will be used in the output buffer.

It is important to note that when scaling the PS buffer, the coordinates of the PS buffer within the output buffer need to be consistent with the scaling factors and original PS buffer size.

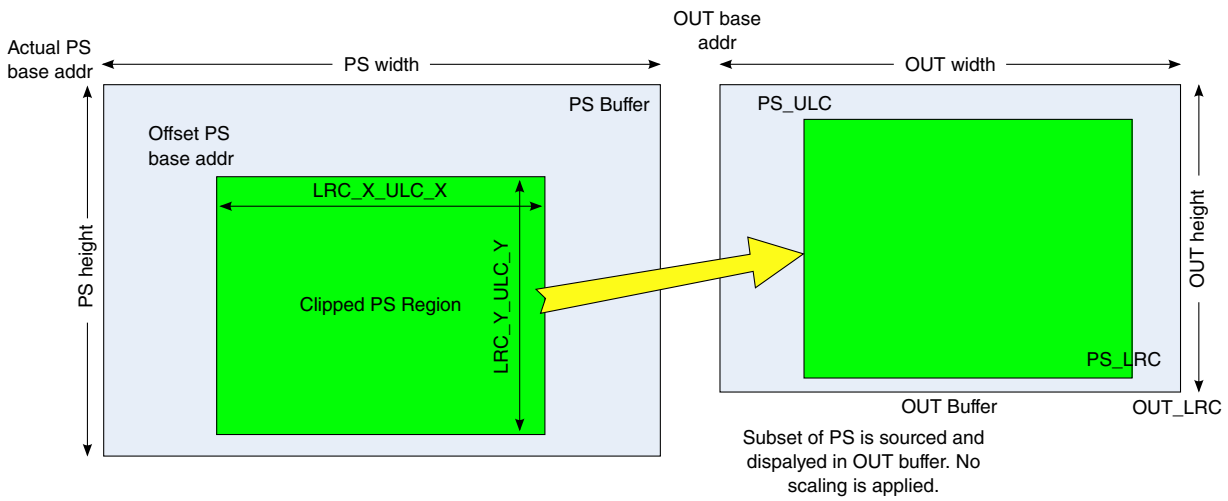


Figure 36-20. PS Buffer Scaling

When sourcing a subset of the PS image, it should fall completely within the PS buffer to avoid displaying incorrect data. The following conditions should be met:

x_base_addr_offset + x_scale * (LRC_X-ULC_X) <= PS_pitch

y_base_addr_offset + y_scale * (LRC_Y-ULC_Y) <= PS_size

The PXP hardware does not check for these conditions and will render the image as programmed. The following case could indicate invalid programming parameters for the PXP:

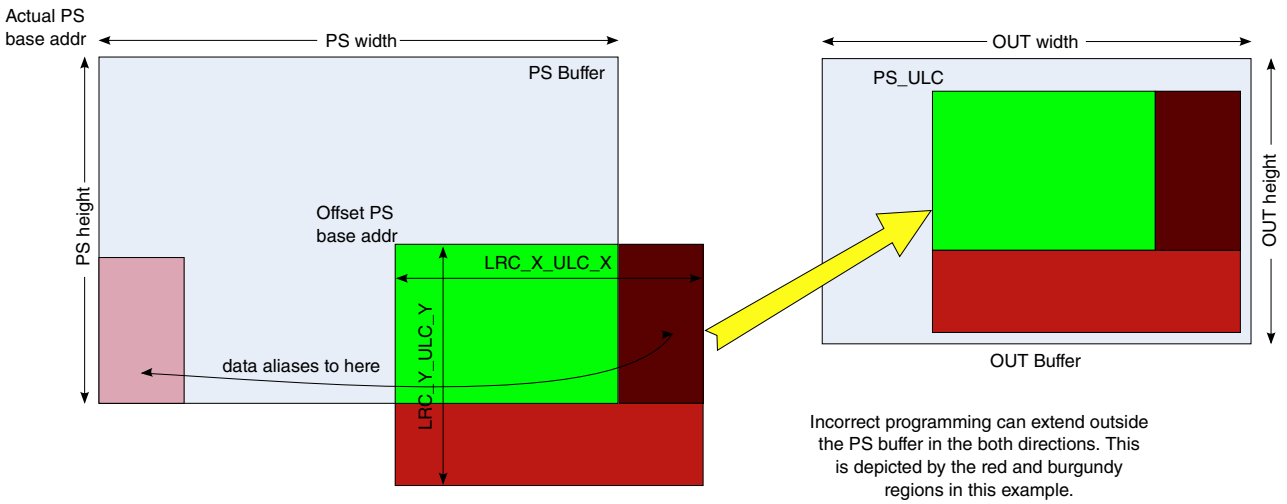


Figure 36-21. Example with Invalid Parameters

36.3.1.27 Color Key Processing

Pixels may be made transparent to the corresponding AS by using the PS color key registers.

If a PS pixel matches the range specified by the REG_PS_COLORKEYLOW and REG_PS_COLORKEYHIGH registers, the pixel from the associated AS will be displayed. If no AS is present for the pixel, a black pixel will be generated since the default AS pixel is 0x00000000 (transparent black pixel).

The most common use for this is when a bitmap does not support an alpha-field or for applications such as "green screen" where an image is substituted for a solid background color .

Figure 36-22. The PS image (player) and AS image (stadium)

The green portion of the background image can be color keyed to display the contents of the AS buffer for locations that match the color range. For this example, the color range is:

PS Colorkey: 00<R<80 70<G<ff 00<B<80

The resulting image becomes:

Figure 36-23. Resulting Image

36.3.1.28 In Place Processing (PS buffer is destination buffer)

The PXP also has the ability to process an image and write the resulting buffer back to the original PS buffer. This is referred to as "in place" rendering.

This could be useful for basic blit operations into the PS buffer. IN_PLACE operations are achieved by programming the OUT base address to the pixel location in the PS buffer that marks the upper left pixel of the update region. The actual region that is updated should be indicated by programming the ULC = (0,0) and the LRC = (X,Y). The region bounded by the coordinates will be updated, and the rest of the PS buffer will not be modified.

36.3.1.29 Alpha Surface (AS) Processing

The AS surface has a complete set of registers that determines how the AS effects the final OUT surface.

Most of the registers that exist for the PS surface also are defined for the AS surface where applicable. This is provided to replicate the SW interface for each PS and AS processes.

36.3.1.30 Alpha Handling

Alpha values in the AS are embedded in the source image pixels. For AS pixel formats that do not support an alpha value, the pixel is assigned an alpha value of 0xFF (opaque).

This can be modified by the AS control by setting either the ALPHA_MULTIPLY or ALPHA_OVERRIDE bit in the associated AS_CTRL register. If ALPHA_MULTIPLY is enabled, the 8-bit ALPHA value from the AS_CTRL register is multiplied by the source alpha before blending with the PS image. If the ALPHA_OVERRIDE bit is set, the 8-bit ALPHA value is simply substituted for the pixel.

36.3.1.31 Color Key Processing (AS_CTRL)

The AS_CTRL[ENABLE_COLORKEY] bit that can be used to enable or disable color key substitution for the AS.

When enabled, the pixel values are compared to the AS_COLORKEYLOW and AS_COLORKEYHIGH registers to determine if a match has occurred. When an AS pixel matches the color key range, the pixel from the AS image is considered transparent and the corresponding PS pixel is rendered.

AS color keys are handled in a manner similar to PS color keys. The same images used in the PS color key example could be used with the images swapped. In this case, matches on the AS image to the AS_COLORKEY register would display the PS pixels.

36.3.2 Output Image Processing

Several PXP options affect the resulting output image.

36.3.2.1 Output Image Size

The PXP generates an output image in the resolution programmed by the OUT_LRC register. As the PXP processes pixels, it iterates over the NxN blocks (in output scan-block order) based on the final image resolution.

36.3.2.2 Output Format

The result of PXP operations are written to the buffer pointed to by the OUT_BUF/ OUT_BUF2 registers. The pixel format is controlled by the OUT_CTRL[FORMAT] bit-field.

32-bit pixels are formed directly from the internal 24-bit representations and 16-bit pixel formats are generated by truncating the internal 24-bit values to the appropriate number of bits. For formats supporting an alpha value, the PXP assigns the alpha using the 8-bit value in the OUT_CTRL[ALPHA] field. For ARGB1555, the most significant alpha bit is appended to the output pixel. Also, for ARGB4444, the most significant nibble is appended to the output pixel. Single and dual buffer YUV output formats are also available. Since each pixel in the data path is represented by a full YUV444 24bpp value, decimation reduces the output in cases of YUV422/420 output formats.

36.3.2.3 Rotation/Flip operations

The PXP supports four rotation angles in conjunction with vertical and horizontal flip options. The flip operations effectively take place before the rotation.

Rotations of 0, 90, 180, and 270 degrees are supported and any combination of rotation and flip are supported. There is no performance difference between any of these modes of operation.

36.3.3 Queuing PXP transactions

The PXP supports a primitive ability to queue up one operation while the current operation is running. This is enabled through the use of the NEXT register.

When this register is written, it enables the PXP to reload its current register contents with the data found at the location pointed to by this address when it completes processing of the current frame. This feature may be useful in helping to reduce the interrupt latency in servicing the PXP, especially in cases where the PXP and eLCDIF are using the on-chip SRAM buffer handshake (since the PXP must begin generating next frame data immediately).

If the PXP is idle when the NEXT register is written, the PXP treats this as an indication that it should immediately load the values at the pointer and begin processing the frame. This ability should allow software to use the same routines when programming the PXP (so that the first frame doesn't differ from subsequent frames).

When loading values from the NEXT register, all registers in the PXP are reloaded. Some register loads have no effect

After writing the NEXT register, the PXP will set the NEXT[ENABLED] bit of the NEXT register to indicate that the next command has been queued. Software should first check the status of this bit to ensure that a previous command has not been enabled. Likewise, after programming the first frame in a sequence of frames, software should poll this bit until it is sampled logic 1'b0 before queuing the next operation.

The PXP will issue interrupts from frames as they complete, regardless of whether they were started by writing the control registers directly or using the NEXT register. When software receives an interrupt, it should check/clear the PXP's status register as normal, poll the NEXT[ENABLED] bit, and then issue the next operation. A queued operation may be cancelled by issuing a CLEAR operation to the NEXT[ENABLED] register bit. The SET and TOGGLE operations should never be used with this register.

36.3.4 Error Handling

The PXP does minimal checking on the control registers, so it is important that these are correctly specified. The PXP does monitor the bus transactions for errors and will report errors in the status register.

Upon receipt of a bus error, the PXP will set the ERROR interrupt and abort any further operations. Bus errors can be generated from any system access that results in an error response returned from the internal SIM Bus errors in the PXP are signaled as either a read or a write error, but do not indicate the failing address. Software may deduce the failing address from the current block status indicators.

36.3.4.1 Known PXP Limitations/Issues

The PXP has the following known limitations:

1. When using the NEXT register, the interrupt enable setting should remain the same for all frames. If not, the PXP will change the interrupt enable register value and possible cause the loss of an interrupt.
2. Rotations of 180/270 are not supported when performing LCD handshakes
3. PS buffer rotation and decimation does not function at the same time
4. No combination of rotate with flip, scaling or decimation is possible if buffer is unaligned.

36.3.5 Clocks

The following table describes the clock sources for PXP. Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 36-2. PXP Clocks

Clock name	Clock Root	Description
clk	ipg_clk_root	PXP clock

36.3.6 Interrupts

The PXP consists of the following interrupts. Refer to the STAT register for details.

- Current PXP status interrupt (STAT[IRQ])
- LUT DMA transfer interrupt (STAT[LUT_DMA_LOAD_DONE_IRQ]
- Next Command issued interrupt (STAT[NEXT_IRQ])
- AXI ID error interrupt (STAT[AXI_ERROR_ID])
- AXI read error interrupt (STAT[AXI_READ_ERROR])
- AXI write error interrupt (STAT[AXI_WRITE_ERROR])

36.4 External Signals

PXP does not contain external signals.

36.5 Initialization

To initialize PXP, follow the steps:

1. Set CTRL[SFTRST] initially. Subsequently clear CTRL[SFTRST] and CTRL[CLKGATE] to reset PXP.
2. Configure the PXP registers
3. Set CTRL[ENABLE] = 1 to start PXP processing
4. Monitor STAT register to check if the processing is complete

Repeat steps 2-4 for the next fetch.

36.6 PXP Memory Map/Register Definition

PXP Hardware Register Format Summary

PXP memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
402B_4000	Control Register 0 (PXP_CTRL)	32	R/W	C000_0000h	36.6.1/1943
402B_4004	Control Register 0 (PXP_CTRL_SET)	32	R/W	C000_0000h	36.6.1/1943
402B_4008	Control Register 0 (PXP_CTRL_CLR)	32	R/W	C000_0000h	36.6.1/1943
402B_400C	Control Register 0 (PXP_CTRL_TOG)	32	R/W	C000_0000h	36.6.1/1943
402B_4010	Status Register (PXP_STAT)	32	R/W	0000_0000h	36.6.2/1946
402B_4014	Status Register (PXP_STAT_SET)	32	R/W	0000_0000h	36.6.2/1946
402B_4018	Status Register (PXP_STAT_CLR)	32	R/W	0000_0000h	36.6.2/1946
402B_401C	Status Register (PXP_STAT_TOG)	32	R/W	0000_0000h	36.6.2/1946
402B_4020	Output Buffer Control Register (PXP_OUT_CTRL)	32	R/W	0000_0000h	36.6.3/1948
402B_4024	Output Buffer Control Register (PXP_OUT_CTRL_SET)	32	R/W	0000_0000h	36.6.3/1948
402B_4028	Output Buffer Control Register (PXP_OUT_CTRL_CLR)	32	R/W	0000_0000h	36.6.3/1948
402B_402C	Output Buffer Control Register (PXP_OUT_CTRL_TOG)	32	R/W	0000_0000h	36.6.3/1948
402B_4030	Output Frame Buffer Pointer (PXP_OUT_BUF)	32	R/W	0000_0000h	36.6.4/1950

Table continues on the next page...

PXP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
402B_4040	Output Frame Buffer Pointer #2 (PXP_OUT_BUF2)	32	R/W	0000_0000h	36.6.5/1951
402B_4050	Output Buffer Pitch (PXP_OUT_PITCH)	32	R/W	0000_0000h	36.6.6/1951
402B_4060	Output Surface Lower Right Coordinate (PXP_OUT_LRC)	32	R/W	0000_0000h	36.6.7/1952
402B_4070	Processed Surface Upper Left Coordinate (PXP_OUT_PS_ULC)	32	R/W	0000_0000h	36.6.8/1953
402B_4080	Processed Surface Lower Right Coordinate (PXP_OUT_PS_LRC)	32	R/W	0000_0000h	36.6.9/1954
402B_4090	Alpha Surface Upper Left Coordinate (PXP_OUT_AS_ULC)	32	R/W	0000_0000h	36.6.10/ 1955
402B_40A0	Alpha Surface Lower Right Coordinate (PXP_OUT_AS_LRC)	32	R/W	0000_0000h	36.6.11/ 1956
402B_40B0	Processed Surface (PS) Control Register (PXP_PS_CTRL)	32	R/W	0000_0000h	36.6.12/ 1957
402B_40B4	Processed Surface (PS) Control Register (PXP_PS_CTRL_SET)	32	R/W	0000_0000h	36.6.12/ 1957
402B_40B8	Processed Surface (PS) Control Register (PXP_PS_CTRL_CLR)	32	R/W	0000_0000h	36.6.12/ 1957
402B_40BC	Processed Surface (PS) Control Register (PXP_PS_CTRL_TOG)	32	R/W	0000_0000h	36.6.12/ 1957
402B_40C0	PS Input Buffer Address (PXP_PS_BUF)	32	R/W	0000_0000h	36.6.13/ 1958
402B_40D0	PS U/Cb or 2 Plane UV Input Buffer Address (PXP_PS_UBUF)	32	R/W	0000_0000h	36.6.14/ 1959
402B_40E0	PS V/Cr Input Buffer Address (PXP_PS_VBUF)	32	R/W	0000_0000h	36.6.15/ 1960
402B_40F0	Processed Surface Pitch (PXP_PS_PITCH)	32	R/W	0000_0000h	36.6.16/ 1960
402B_4100	PS Background Color (PXP_PS_BACKGROUND)	32	R/W	0000_0000h	36.6.17/ 1961
402B_4110	PS Scale Factor Register (PXP_PS_SCALE)	32	R/W	1000_1000h	36.6.18/ 1962
402B_4120	PS Scale Offset Register (PXP_PS_OFFSET)	32	R/W	0000_0000h	36.6.19/ 1964
402B_4130	PS Color Key Low (PXP_PS_CLRKEYLOW)	32	R/W	00FF_FFFFh	36.6.20/ 1964
402B_4140	PS Color Key High (PXP_PS_CLRKEYHIGH)	32	R/W	0000_0000h	36.6.21/ 1965
402B_4150	Alpha Surface Control (PXP_AS_CTRL)	32	R/W	0000_0000h	36.6.22/ 1966
402B_4160	Alpha Surface Buffer Pointer (PXP_AS_BUF)	32	R/W	0000_0000h	36.6.23/ 1968
402B_4170	Alpha Surface Pitch (PXP_AS_PITCH)	32	R/W	0000_0000h	36.6.24/ 1969

Table continues on the next page...

PXP memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
402B_4180	Overlay Color Key Low (PXP_AS_CLRKEYLOW)	32	R/W	00FF_FFFFh	36.6.25/ 1969
402B_4190	Overlay Color Key High (PXP_AS_CLRKEYHIGH)	32	R/W	0000_0000h	36.6.26/ 1970
402B_41A0	Color Space Conversion Coefficient Register 0 (PXP_CSC1_COEF0)	32	R/W	0400_0000h	36.6.27/ 1971
402B_41B0	Color Space Conversion Coefficient Register 1 (PXP_CSC1_COEF1)	32	R/W	0123_0208h	36.6.28/ 1972
402B_41C0	Color Space Conversion Coefficient Register 2 (PXP_CSC1_COEF2)	32	R/W	079B_076Ch	36.6.29/ 1973
402B_4320	PXP Power Control Register (PXP_POWER)	32	R/W	0000_0000h	36.6.30/ 1974
402B_4400	Next Frame Pointer (PXP_NEXT)	32	R/W	0000_0000h	36.6.31/ 1974
402B_4440	PXP Alpha Engine A Control Register. (PXP_PORTER_DUFF_CTRL)	32	R/W	0000_0000h	36.6.32/ 1977

36.6.1 Control Register 0 (PXP_CTRL*n*)

The CTRL register contains controls for the PXP module.

PXP_CTRL: 0x000

PXP_CTRL_SET: 0x004

PXP_CTRL_CLR: 0x008

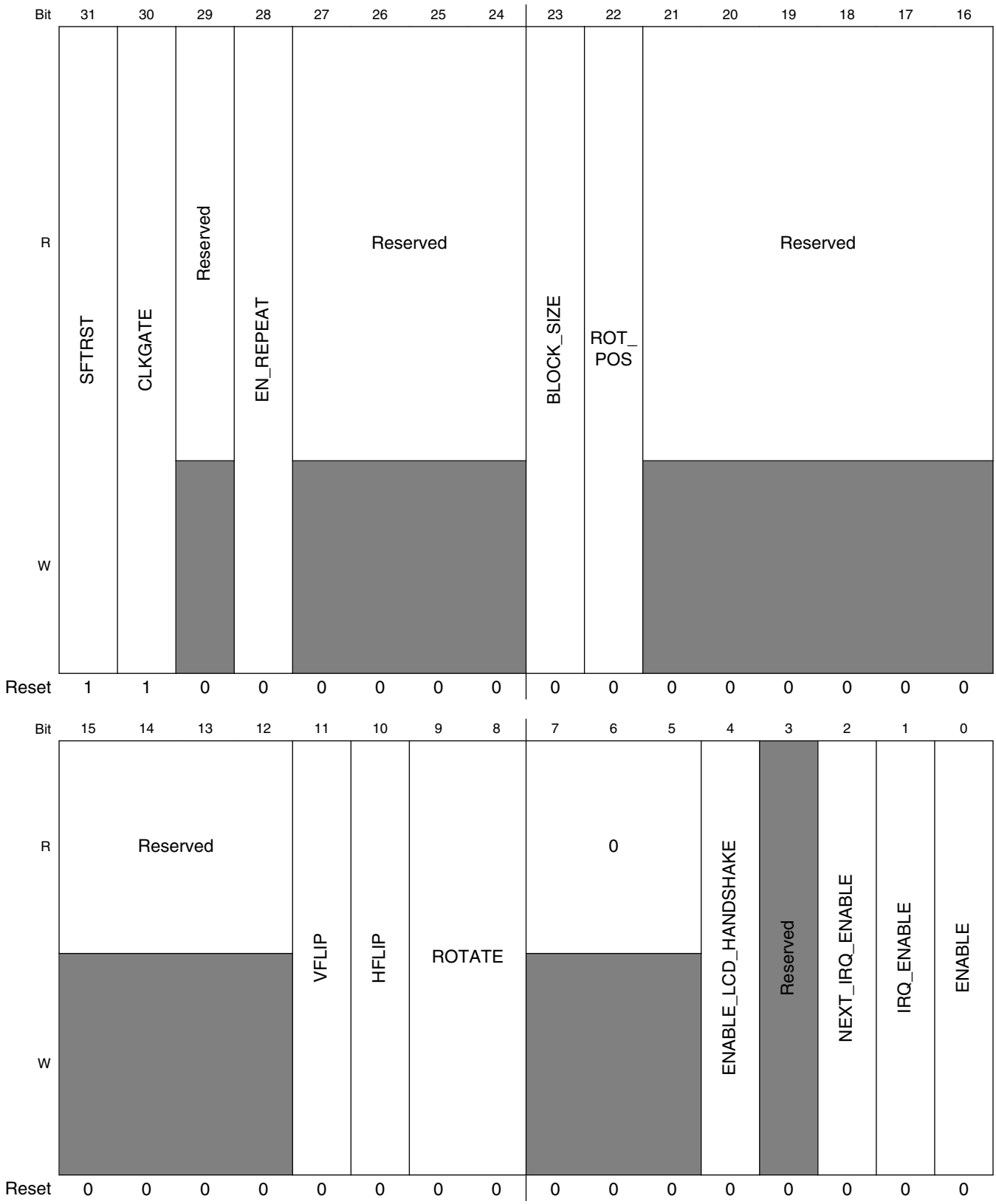
PXP_CTRL_TOG: 0x00C

The Control register contains the primary controls for the PXP block. The present bits indicate which of the sub-features of the block are present in the hardware.

EXAMPLE

```
PXP_CTRL_SET(BM_PXP_CTRL_SFTRST);
PXP_CTRL_CLR(BM_PXP_CTRL_SFTRST | BM_PXP_CTRL_CLKGATE);
```

Address: 402B_4000h base + 0h offset + (4d × i), where i=0d to 3d



PXP_CTRL*n* field descriptions

Field	Description
31 SFTRST	This bit can be turned on and then off to reset the PXP block to its default state. 0x0 **Enabled** — Normal PXP operation is enabled 0x1 **Disabled** — Clocking with PXP is disabled and held in its reset (lowest power) state. This is the default value.
30 CLKGATE	This bit must be set to zero for normal operation. When set to one it gates off the clocks to the block. 0x0 **Normal** — Normal operation 0x1 **Gated** — All clocks to PXP is gated-off
29 Reserved	This field is reserved. Always set to zero.
28 EN_REPEAT	Enable the PXP to run continuously. This bit should be set when the LCDIF handshake mode is enabled so that the next frame is automatically generated for the next screen refresh cycle. If it not set and the handshake mode is enabled, the CPU will have to initiate the PXP for the next refresh cycle. When the PXP NEXT feature is used, it has priority over the REPEAT mode, in that the new register settings are fetched first, and then the next PXP operation will continue. 0x1 **Repeat** — PXP will repeat based on the current configuration register settings 0x0 **Complete** — PXP will complete the process and enter the idle state ready to accept the next frame to be processed
27–24 Reserved	This field is reserved. Always set to zero.
23 BLOCK_SIZE	Select the block size to process. 0x0 **8X8** — Process 8x8 pixel blocks. 0x1 **16X16** — Process 16x16 pixel blocks.
22 ROT_POS	This bit controls where rotation will occur in the PXP datapath. Setting this bit to 1'b0 will place the rotation resources at the output stage of the PXP data path. Image compositing will occur before pixels are processed for rotation. Setting this bit to a 1'b1 will place the rotation resources before image composition. Only the PS can be rotated in this configuration and AS will not be rotated.
21–12 Reserved	This field is reserved. Always set to zero.
11 VFLIP	Indicates that the output buffer should be flipped vertically (effect applied before rotation). 0x0 **Disabled** — Vertical Flip is disabled 0x1 **Enabled** — Vertical Flip is enabled
10 HFLIP	Indicates that the output buffer should be flipped horizontally (effect applied before rotation). 0x0 **Disabled** — Horizontal Flip is disabled 0x1 **Enabled** — Horizontal Flip is enabled
9–8 ROTATE	Indicates the clockwise rotation to be applied at the output buffer. The rotation effect is defined as occurring after the FLIP_X and FLIP_Y permutation. 0x0 **ROT_0** — 0x1 **ROT_90** — 0x2 **ROT_180** — 0x3 **ROT_270** —
7–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

PXP_CTRL*n* field descriptions (continued)

Field	Description
4 ENABLE_LCD_ HANDSHAKE	Enable handshake with LCD controller. When this is set, the PXP will not process an entire framebuffer, but will instead process rows of NxN blocks in a double-buffer handshake with the LCDIF. This enables the use of the onboard SRAM for a partial frame buffer.
3 Reserved	This field is reserved.
2 NEXT_IRQ_ ENABLE	Next command interrupt enable. When set, the PXP will issue an interrupt when a queued command initiated by a write to the PXP_NEXT register has been loaded into the PXP's registers. This interrupt also indicates that a new command may now be queued. 0x0 **Disabled** — 0x1 **Enabled** —
1 IRQ_ENABLE	Interrupt enable **NOTE:** When using the PXP_NEXT functionality to reprogram the PXP, the new value of this bit will be used and may therefore enable or disable an interrupt unintentionally. 0x1 **Enabled** — PXP interrupt is enabled 0x0 **Disabled** — PXP interrupt is disabled
0 ENABLE	Enables PXP operation with specified parameters. The ENABLE bit will remain set while the PXP is active and will be cleared after the current operation completes. Software should use the IRQ bit in the PXP_STAT when polling for PXP completion. 0x1 **Enabled** — PXP is enabled 0x0 **Disabled** — PXP is disabled

36.6.2 Status Register (PXP_STAT*n*)

The PXP Interrupt Status register provides interrupt status information.

PXP_STAT: 0x010

PXP_STAT_SET: 0x014

PXP_STAT_CLR: 0x018

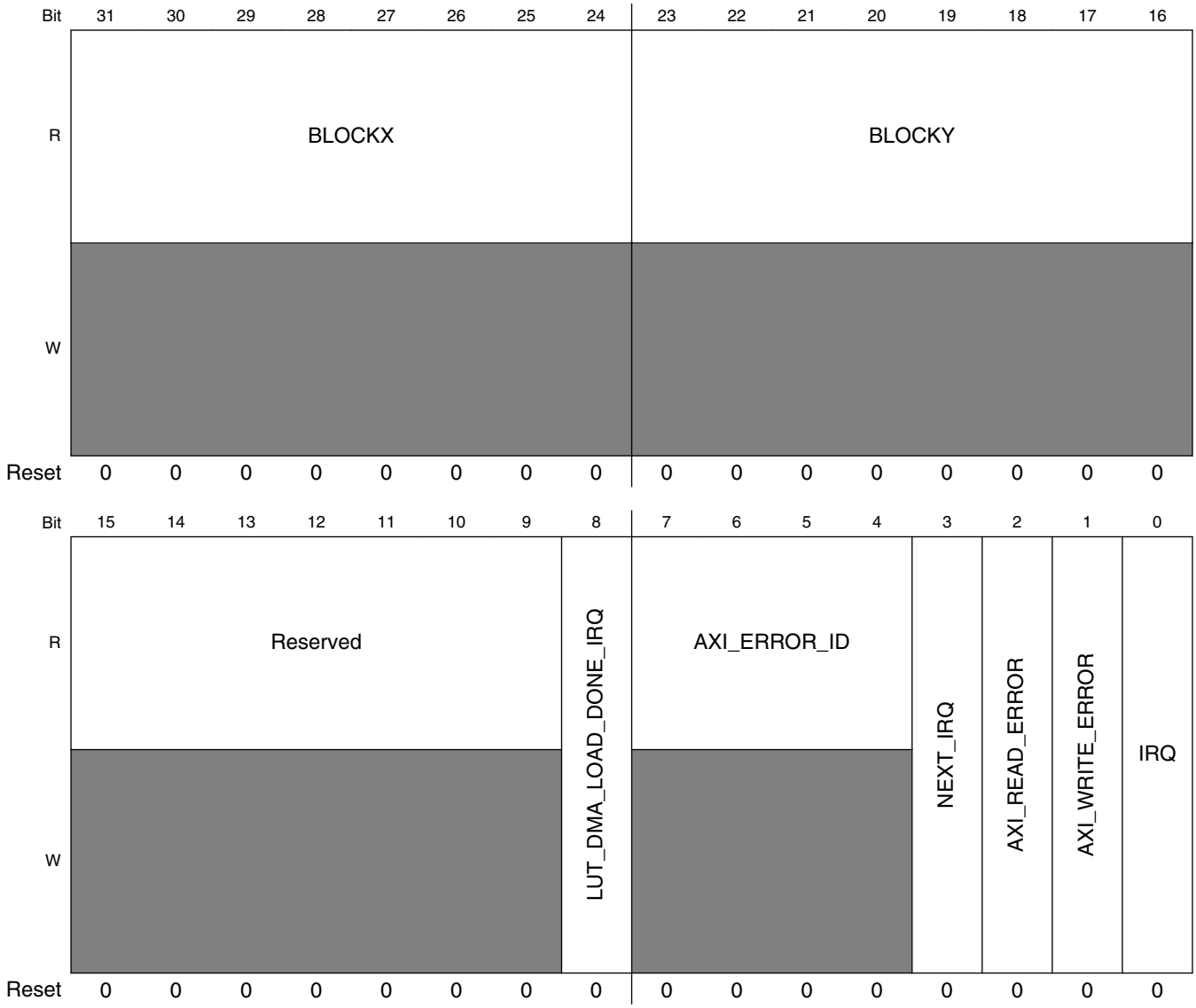
PXP_STAT_TOG: 0x01C

This register provides PXP interrupt status and the current X/Y block coordinate that is being processed.

EXAMPLE

```
PXP_STAT_CLR(BM_PXP_STAT_IRQ);  // clear CSC interrupt
```

Address: 402B_4000h base + 10h offset + (4d × i), where i=0d to 3d



PXP_STAT*n* field descriptions

Field	Description
31–24 BLOCKX	Indicates the X coordinate of the block currently being rendered.
23–16 BLOCKY	Indicates the X coordinate of the block currently being rendered.
15–9 Reserved	This field is reserved. Always set to zero.
8 LUT_DMA_ LOAD_DONE_ IRQ	Indicates that the LUT DMA transfer has completed. 0x0 **Active** — LUT DMA LOAD transfer is active 0x1 **Complete** — LUT DMA LOAD transfer is complete
7–4 AXI_ERROR_ID	Indicates the AXI ID of the failing bus operation.

Table continues on the next page...

PXP_STAT*n* field descriptions (continued)

Field	Description
3 NEXT_IRQ	Indicates that a command issued with the "Next Command" functionality has been issued and that a new command may be initiated with a write to the PXP_NEXT register.
2 AXI_READ_ ERROR	Indicates PXP encountered an AXI read error and processing has been terminated. 0x0 **Normal** — AXI read is normal 0x1 **Error** — AXI read error has occurred
1 AXI_WRITE_ ERROR	Indicates PXP encountered an AXI write error and processing has been terminated. 0x0 **Normal** — AXI write is normal 0x1 **Error** — AXI write error has occurred
0 IRQ	Indicates current PXP interrupt status. The IRQ is routed through the pxp_irq when the IRQ_ENABLE bit in the control register is set. 0x0 **No interrupt** — PXP interrupt not generated 0x1 **Interrupt generated** — PXP interrupt is generated

36.6.3 Output Buffer Control Register (PXP_OUT_CTRL*n*)

The OUT_CTRL register contains controls for the Output Buffer.

PXP_OUT_CTRL: 0x020

PXP_OUT_CTRL_SET: 0x024

PXP_OUT_CTRL_CLR: 0x028

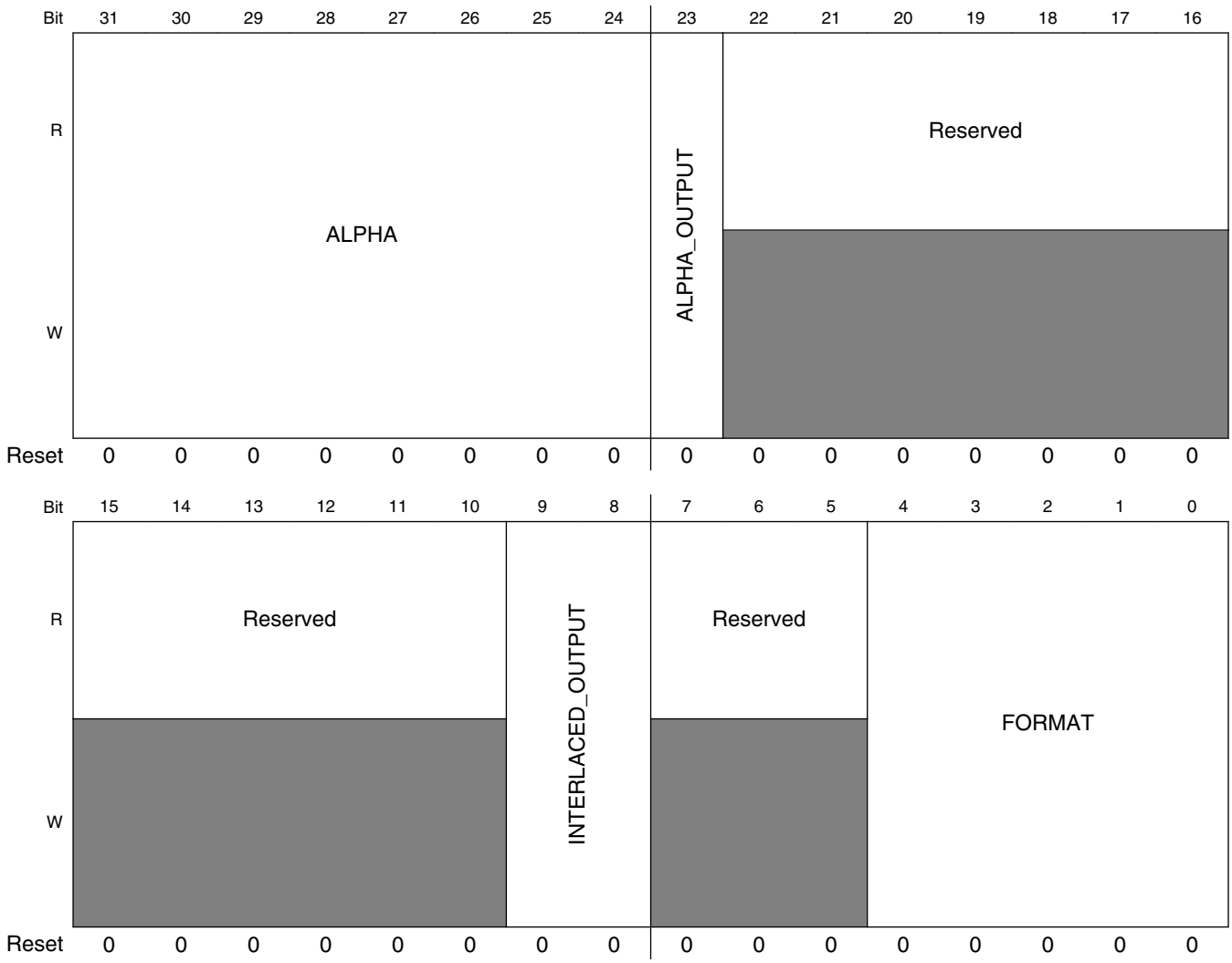
PXP_OUT_CTRL_TOG: 0x02C

The Control register contains the primary controls for the PXP block. The present bits indicate which of the sub-features of the block are present in the hardware.

EXAMPLE

```
PXP_CTRL_SET(BM_PXP_CTRL_SFTRST);
PXP_CTRL_CLR(BM_PXP_CTRL_SFTRST | BM_PXP_CTRL_CLKGATE);
```

Address: 402B_4000h base + 20h offset + (4d × i), where i=0d to 3d



PXP_OUT_CTRL*n* field descriptions

Field	Description
31–24 ALPHA	When generating an output buffer with an alpha component, the value in this field will be used when enabled to override the alpha passed through the pixel data pipeline.
23 ALPHA_ OUTPUT	Indicates that alpha component in output buffer pixels should be overwritten by PXP_OUT_CTRL[ALPHA]. If 0, retain their alpha value from the computed alpha for that pixel. 0x0 **Retain** — 0x1 **Overwritten** —
22–10 Reserved	This field is reserved. Always set to zero.
9–8 INTERLACED_ OUTPUT	Determines how the PXP writes it's output data. Output interlacing should not be used in conjunction with input interlacing. Splitting frames into fields is most efficient using output interlacing. 2-plane output formats AND interlaced output is NOT supported. 0x0 **PROGRESSIVE** — All data written in progressive format to the OUTBUF Pointer. 0x1 **FIELD0** — Interlaced output: only data for field 0 is written to the OUTBUF Pointer.

Table continues on the next page...

PXP_OUT_CTRL*n* field descriptions (continued)

Field	Description
	0x2 **FIELD1** — Interlaced output: only data for field 1 is written to the OUTBUF2 Pointer. 0x3 **INTERLACED** — Interlaced output: data for field 0 is written to OUTBUF and data for field 1 is written to OUTBUF2.
7–5 Reserved	This field is reserved. Always set to zero.
FORMAT	Output framebuffer format. The UV byte lanes are synonymous with CbCr byte lanes for YUV output pixel formats. For example, the YUV2P420 format should be selected when the output is YCbCr 2-plane 420 output format. 0x0 **ARGB8888** — 32-bit pixels 0x4 **RGB888** — 32-bit pixels (unpacked 24-bit pixel in 32 bit DWORD.) 0x5 **RGB888P** — 24-bit pixels (packed 24-bit format) 0x8 **ARGB1555** — 16-bit pixels 0x9 **ARGB4444** — 16-bit pixels 0xC **RGB555** — 16-bit pixels 0xD **RGB444** — 16-bit pixels 0xE **RGB565** — 16-bit pixels 0x10 **YUV1P444** — 32-bit pixels (1-plane XYUV unpacked) 0x12 **UYVY1P422** — 16-bit pixels (1-plane U0,Y0,V0,Y1 interleaved bytes) 0x13 **VYUY1P422** — 16-bit pixels (1-plane V0,Y0,U0,Y1 interleaved bytes) 0x14 **Y8** — 8-bit monochrome pixels (1-plane Y luma output) 0x15 **Y4** — 4-bit monochrome pixels (1-plane Y luma, 4 bit truncation) 0x18 **YUV2P422** — 16-bit pixels (2-plane UV interleaved bytes) 0x19 **YUV2P420** — 16-bit pixels (2-plane UV) 0x1A **YVU2P422** — 16-bit pixels (2-plane VU interleaved bytes) 0x1B **YVU2P420** — 16-bit pixels (2-plane VU)

36.6.4 Output Frame Buffer Pointer (PXP_OUT_BUF)

Output Framebuffer Pointer. This register points to the beginning of the output frame buffer. This pointer is used for progressive format and field 0 when generating interlaced output.

This register is used by the logic to point to the current output location for the output frame buffer.

EXAMPLE

```
PXP_OUT_BUF_WR( buffer );
```

Address: 402B_4000h base + 30h offset = 402B_4030h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_OUT_BUF field descriptions

Field	Description
ADDR	Current address pointer for the output frame buffer. The address can have any byte alignment. 64B alignment is recommended for optimal performance.

36.6.5 Output Frame Buffer Pointer #2 (PXP_OUT_BUF2)

Output Framebuffer Pointer #2. This register points to the beginning of the output frame buffer for either field 1 when generating interlaced output or for the UV buffer when in YUV 2-plane output modes. Both interlaced output AND 2-plane output modes are not supported in a single PXP operation. This register is not used as the pointer to the second buffer when in LCDIF_HANDSHAKE mode.

This register is used by the logic to point to the current output location for the field 1 or UV output frame buffer.

EXAMPLE

```
PXP_OUT_BUF_WR( field0 );  // buffer for interlaced field 0
PXP_OUT_BUF2_WR( field1 ); // buffer for interlaced field 1
```

Address: 402B_4000h base + 40h offset = 402B_4040h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_OUT_BUF2 field descriptions

Field	Description
ADDR	Current address pointer for the output frame buffer. The address can have any byte alignment. 64B alignment is recommended for optimal performance.

36.6.6 Output Buffer Pitch (PXP_OUT_PITCH)

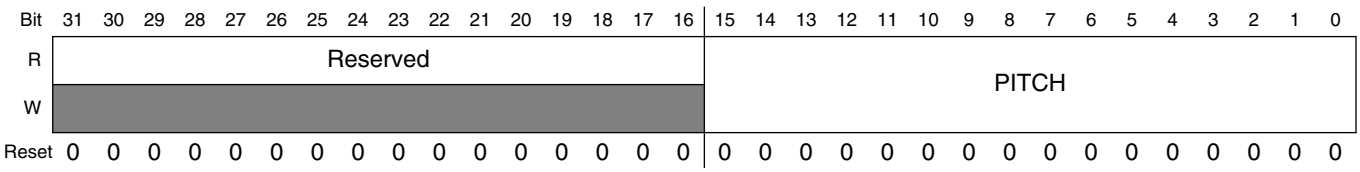
This register contains the output buffer pitch in bytes.

Any byte value will indicate the vertical pitch. This value will be used in output pixel address calculations.

EXAMPLE

```
PXP_OUT_PITCH_WR( 68 * 4 );  // The output buffer pitch is 68 pixels times 32 bits per pixel
```

Address: 402B_4000h base + 50h offset = 402B_4050h



PXP_OUT_PITCH field descriptions

Field	Description
31–16 Reserved	This field is reserved. Always set to zero.
PITCH	Indicates the number of bytes in memory between two vertically adjacent pixels.

36.6.7 Output Surface Lower Right Coordinate (PXP_OUT_LRC)

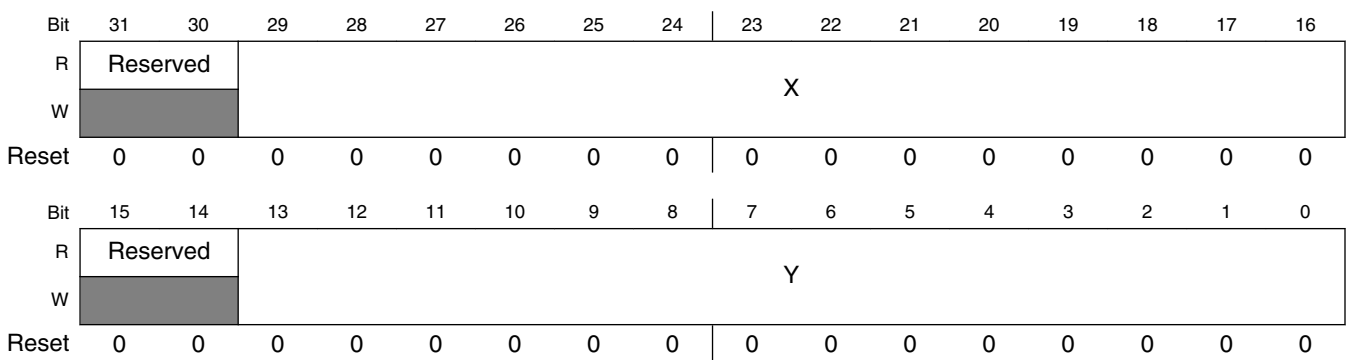
This register contains the size, or lower right coordinate, of the output buffer NOT rotated. It is implied that the upper left coordinate of the output surface is always [0,0]. When rotating the framebuffer, the PXP will automatically swap the X/Y, or WIDTH/HEIGHT, to accomodate the rotated size.

This register sets the size of the output frame buffer in pixels, not blocks. The frame buffer need not be a multiple of NxN pixels. Partial blocks will be written for output frame buffer sizes that are not divisable by N pixels in either dimension.

EXAMPLE

```
PXP_OUT_LRC[X]=319;   // set width of output frame buffer to 320 pixels
PXP_OUT_LRC[Y]=243;  // set height of output frame buffer to 244 pixels which is not
divisible by block size N

PXP_OUT_LRC_WR( BF_PXP_OUT_LRC_X(319)   |   BF_PXP_OUT_LRC_Y(243) );
```

Address: 402B_4000h base + 60h offset = 402B_4060h



PXP_OUT_LRC field descriptions

Field	Description
31–30 Reserved	This field is reserved. Always set to zero.
29–16 X	Indicates number of horizontal PIXELS in the output surface (non-rotated). The output buffer pixel width minus 1 should be programmed. The image size is not required to be a multiple of 8 pixels. The PXP will clip the pixel output at this boundary.
15–14 Reserved	This field is reserved. Always set to zero.
Y	Indicates the number of vertical PIXELS in the output surface (non-rotated). The output buffer pixel height minus 1 should be programmed. The image size is not required to be a multiple of 8 pixels. The PXP will clip the pixel output at this boundary.

36.6.8 Processed Surface Upper Left Coordinate (PXP_OUT_PS_ULC)

This register contains the upper left pixel coordinate for the Processed Surface in the OUTPUT buffer.

This register contains the upper left coordinate of the Processed Surface in the output frame buffer (in pixels). Values that are within the PXP_OUT_LRC X,Y extents are valid. The lowest valid value for these fields is 0,0. If the value of the PXP_OUT_PS_ULC is greater than the PXP_OUT_LRC, then no PS pixels will be fetched from memory, but only PXP_PS_BACKGROUND pixels will be processed by the PS engine. Pixel locations that are greater than or equal to the PS upper left coordinates, less than or equal to the PS lower right coordinates, and within the PXP_OUT_LRC extents will use the PS to render pixels into the output buffer.

EXAMPLE

```
PXP_OUT_PS_ULC_WR(0,0x0002_0002); // Processed Surface upper left coordinate at (X,Y) =
2,2.  The PS surface will not effect pixels in the first and second row and column of the
output buffer.
```

Address: 402B_4000h base + 70h offset = 402B_4070h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved		X													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved		Y													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_OUT_PS_ULC field descriptions

Field	Description
31–30 Reserved	This field is reserved. Always set to zero.
29–16 X	This field indicates the upper left X-coordinate (in pixels) of the processed surface (PS) in the output buffer.
15–14 Reserved	This field is reserved. Always set to zero.
Y	This field indicates the upper left Y-coordinate (in pixels) of the processed surface in the output buffer.

36.6.9 Processed Surface Lower Right Coordinate (PXP_OUT_PS_LRC)

This register contains the lower right extent for the Processed Surface in the OUTPUT buffer.

This register contains the lower right coordinate of the Processed Surface in the output frame buffer (in pixels). Values that are within the PXP_OUT_LRC X,Y extents are valid. The lowest valid value for these fields is 0,0. Pixel locations that are greater than or equal to the PS upper left coordinates, less than or equal to the PS lower right coordinates, and within the PXP_OUT_LRC extents will use the PS to render pixels into the output buffer.

EXAMPLE

```
PXP_OUT_PS_ULC_WR(0,0x03FF_03FF); // With this UL/LR pair of pixel coordinates, only one
pixel at OUT[X,Y]=1023,1023 will use the PS to contribute to its value.
PXP_OUT_PS_LRC_WR(0,0x03FF_03FF);
```

Address: 402B_4000h base + 80h offset = 402B_4080h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved		X													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved		Y													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_OUT_PS_LRC field descriptions

Field	Description
31–30 Reserved	This field is reserved. Always set to zero.
29–16 X	This field indicates the lower right X-coordinate (in pixels) of the processed surface (PS) in the output frame buffer.
15–14 Reserved	This field is reserved. Always set to zero.
Y	This field indicates the lower right Y-coordinate (in pixels) of the processed surface in the output frame buffer.

36.6.10 Alpha Surface Upper Left Coordinate (PXP_OUT_AS_ULC)

This register contains the upper left location for the Alpha Surface in the output buffer.

This register contains the upper left coordinate of AS in the output frame buffer (in pixels). Values that are within the PXP_OUT_LRC X,Y extents are valid. The lowest valid value for these fields is 0,0. Pixel locations that are greater than or equal to the upper left coordinates will use the AS to render pixels in the output buffer.

EXAMPLE

```
PXP_OUT_AS_ULC_WR(0,0x0001_0001); // Alpha Surface upper left coordinate at (X,Y) = 1,1.
The AS surface will not effect pixels in the first row or first column of the output buffer.
```

Address: 402B_4000h base + 90h offset = 402B_4090h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved								X							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								Y							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_OUT_AS_ULC field descriptions

Field	Description
31–30 Reserved	This field is reserved. Always set to zero.
29–16 X	This field indicates the upper left X-coordinate (in pixels) of the alpha surface (AS) in the output frame buffer.

Table continues on the next page...

PXP_OUT_AS_ULC field descriptions (continued)

Field	Description
15–14 Reserved	This field is reserved. Always set to zero.
Y	This field indicates the upper left Y-coordinate (in pixels) of the alpha surface in the output frame buffer.

36.6.11 Alpha Surface Lower Right Coordinate (PXP_OUT_AS_LRC)

This register contains the lower right extent for Alpha Surface in the output buffer.

This register contains the lower right coordinate of AS in the output frame buffer (in pixels). Values that are within the PXP_OUT_LRC X,Y extents are valid. The lowest valid value for these fields is 0,0. Pixel locations that are less than or equal to the lower right coordinates will use the AS to render pixels in the output buffer.

EXAMPLE

```
PXP_AS_LRC_WR(0,0x03FF_03FF); // Alpha Surface lower right coordinate at (X,Y) = 1023,1023.
```

Address: 402B_4000h base + A0h offset = 402B_40A0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved		X													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved		Y													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_OUT_AS_LRC field descriptions

Field	Description
31–30 Reserved	This field is reserved. Always set to zero.
29–16 X	This field indicates the lower right X-coordinate (in pixels) of the alpha surface (AS) in the output frame buffer.
15–14 Reserved	This field is reserved. Always set to zero.
Y	This field indicates the lower right Y-coordinate (in pixels) of the alpha surface in the output frame buffer.

36.6.12 Processed Surface (PS) Control Register (PXP_PS_CTRL*n*)

The PS_CTRL register contains controls for the Processed Surface Buffer.

PXP_PS_CTRL: 0x0B0

PXP_PS_CTRL_SET: 0x0b4

PXP_PS_CTRL_CLR: 0x0B8

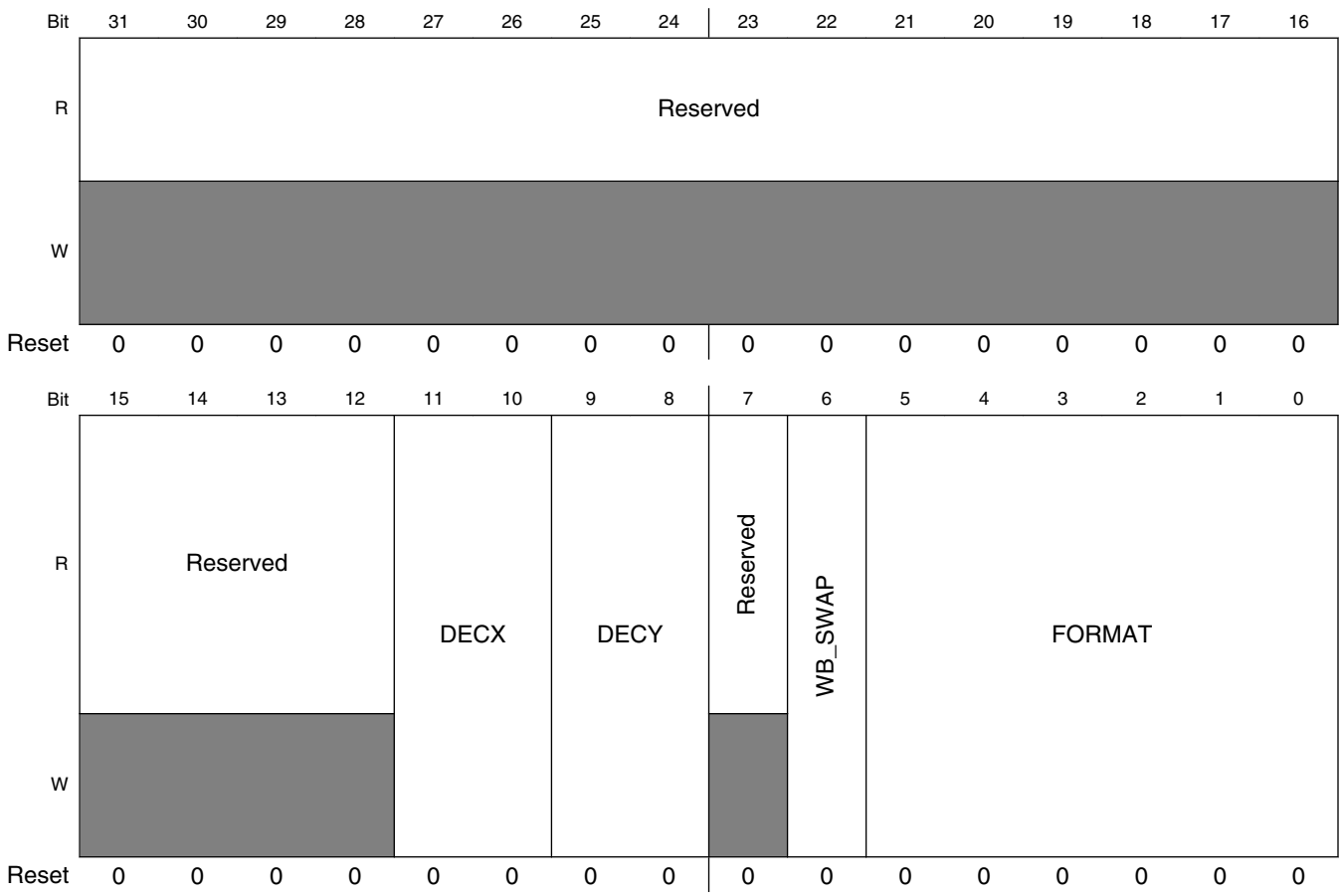
PXP_PS_CTRL_TOG: 0x0BC

The Control register contains the primary controls for the PXP block. The present bits indicate which of the sub-features of the block are present in the hardware.

EXAMPLE

```
PXP_CTRL_SET(BM_PXP_CTRL_SFTRST);
PXP_CTRL_CLR(BM_PXP_CTRL_SFTRST | BM_PXP_CTRL_CLKGATE);
```

Address: 402B_4000h base + B0h offset + (4d × i), where i=0d to 3d



PXP_PS_CTRL*n* field descriptions

Field	Description
31–12 Reserved	This field is reserved. Always set to zero.
11–10 DECX	Horizontal pre decimation filter control. 0x0 **DISABLE** — Disable pre-decimation filter. 0x1 **DECX2** — Decimate PS by 2. 0x2 **DECX4** — Decimate PS by 4. 0x3 **DECX8** — Decimate PS by 8.
9–8 DECY	Verticle pre decimation filter control. 0x0 **DISABLE** — Disable pre-decimation filter. 0x1 **DECY2** — Decimate PS by 2. 0x2 **DECY4** — Decimate PS by 4. 0x3 **DECY8** — Decimate PS by 8.
7 Reserved	This field is reserved. Always set to zero.
6 WB_SWAP	Swap bytes in words. For each 16 bit word, the two bytes will be swapped. 0x0 **Disabled** — Byte swap is disabled 0x1 **Enabled** — Byte swap is enabled
FORMAT	PS buffer format. To select between YUV and YCbCr formats, see bit 31 of the CSC1_COEF0 register. 0x4 **RGB888_ARGB8888** — 32-bit pixels (unpacked 24-bit format with/without alpha at high 8bits) 0xC **RGB555_ARGB1555** — 16-bit pixels with/without alpha at high 1bit 0xD **RGB444_ARGB4444** — 16-bit pixels with/without alpha at high 4 bits 0xE **RGB565** — 16-bit pixels 0x10 **YUV1P444** — 32-bit pixels (1-plane XYUV unpacked) 0x12 **UYVY1P422** — 16-bit pixels (1-plane U0,Y0,V0,Y1 interleaved bytes) 0x13 **VYUY1P422** — 16-bit pixels (1-plane V0,Y0,U0,Y1 interleaved bytes) 0x14 **Y8** — 8-bit monochrome pixels (1-plane Y luma output) 0x15 **Y4** — 4-bit monochrome pixels (1-plane Y luma, 4 bit truncation) 0x18 **YUV2P422** — 16-bit pixels (2-plane UV interleaved bytes) 0x19 **YUV2P420** — 16-bit pixels (2-plane UV) 0x1A **YVU2P422** — 16-bit pixels (2-plane VU interleaved bytes) 0x1B **YVU2P420** — 16-bit pixels (2-plane VU) 0x1E **YUV422** — 16-bit pixels (3-plane format) 0x1F **YUV420** — 16-bit pixels (3-plane format) 0x24 **RGBA8888** — 2-bit pixels with alpha at the low 8 bits 0x2C **RGBA5551** — 16-bit pixels with alpha at the low 1bits 0x2D **RGBA4444** — 16-bit pixels with alpha at the low 4 bits

36.6.13 PS Input Buffer Address (PXP_PS_BUF)

PS Input Buffer Address. This should be programmed to the starting address of the RGB data or Y (luma) data for the PS plane.

This register contains the pointer to the Luma/RGB buffer. If the application requires an offset into the PS buffer, then this address can be set so that the desired offset is achieved. Any byte address is valid. For best performance, 64B alignment is recommended.

EXAMPLE

```
PXP_PS_BUF_WR(image_rgb); // RGB image
PXP_PS_BUF_WR(image_y);   // Y (luma) image data
PXP_PS_UBUF_WR(image_u);  // U (Cb) image data
PXP_PS_VBUF_WR(image_v);  // V (Cr) image data
```

Address: 402B_4000h base + C0h offset = 402B_40C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_PS_BUF field descriptions

Field	Description
ADDR	Address pointer for the PS RGB or Y (luma) input buffer.

36.6.14 PS U/Cb or 2 Plane UV Input Buffer Address (PXP_PS_UBUF)

PS Chroma (U/Cb/UV) Input Buffer Address. This register points to the beginning of the PS U/Cb input buffer. In two plane operation, this register points to the beginning of the PS UV chroma input buffer.

This register contains the pointer to the Chroma U/Cb or 2 plane UV buffer. This register is unused when processing 1-plane buffer formats. If the application requires an offset into the PS buffer, then this address can be set so that the desired offset is achieved. Any byte address is valid. For best performance, 64B alignment is recommended.

EXAMPLE

```
PXP_PS_BUF_WR(image_y);   // Y (luma) image data
PXP_PS_UBUF_WR(image_u);  // U (Cb) image data
PXP_PS_VBUF_WR(image_v);  // V (Cr) image data
```

Address: 402B_4000h base + D0h offset = 402B_40D0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_PS_UBUF field descriptions

Field	Description
ADDR	Address pointer for the PS U/Cb or 2 plane UV Chroma input buffer.

36.6.15 PS V/Cr Input Buffer Address (PXP_PS_VBUF)

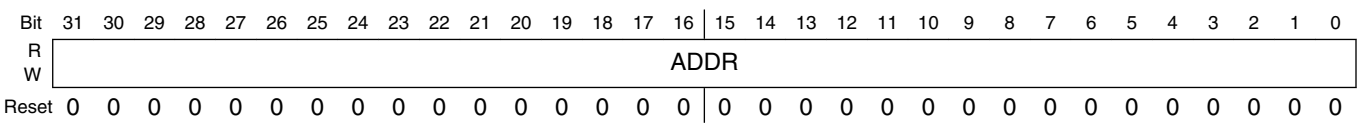
PS Chroma (V/Cr) Input Buffer Address. This register points to the beginning of the PS V/Cr input buffer. In one or two plane operation, this register is not used. In monochrome modes Y8 and Y4, the low 16 bits are used as the U/V data in the datapath instead of sourcing U/V data from external buffers. In this case, it represents a fixed value for U/V data.

This register contains the pointer to the Chroma V/Cr buffer. For Y8/Y4 modes, the low 16 bits are used as the monochrome U and V values in the data path. Bits [15:8] represent the U data byte, and bits]7:0] represent the V data byte. Other than with Y8/Y4 input buffer formats, this register is unused when processing 1 or 2-plane buffer formats. If the application requires an offset into the PS buffer, then this address can be set so that the desired offset is achieved. Any byte address is valid. For best performance, 64B alignment is recommended.

EXAMPLE

```
PXP_PS_BUF_WR(image_y);   // Y (luma) image data
PXP_PS_UBUF_WR(image_u);  // U (Cb) image data
PXP_PS_VBUF_WR(image_v);  // V (Cr) image data
```

Address: 402B_4000h base + E0h offset = 402B_40E0h



PXP_PS_VBUF field descriptions

Field	Description
ADDR	Address pointer for the PS V/Cr Chroma input buffer.

36.6.16 Processed Surface Pitch (PXP_PS_PITCH)

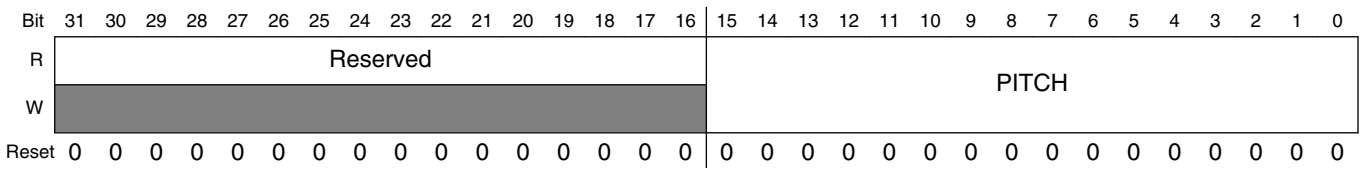
This register contains the processed surface pitch in bytes.

Any byte value will indicate the vertical pitch of the PS source frame buffer. This value will be used in PS pixel address calculations. This value has no relation to the UL and LR registers. It specifies how many bytes are between two vertically adjacent pixels in the input PS surface. For multi-plane formats, the Y buffer pitch should be programmed. For 2-plane YUV422, the UV pitch is the same as the Y pitch. For 3-plane YUV422, the U and V pitch is 1/2 the Y pitch. For 2-plane YUV420, the UV pitch is 1/2 the Y pitch. For 3-plane YUV420, the U and V pitch is 1/4 the Y pitch. All source buffers should comply with these U and V resolution reductions with respect to their Y source buffers.

EXAMPLE

```
PXP_PS_PITCH_WR( 64 * 4 );  // The output buffer pitch is 64 pixels times 32 bits per pixel
```

Address: 402B_4000h base + F0h offset = 402B_40F0h



PXP_PS_PITCH field descriptions

Field	Description
31–16 Reserved	This field is reserved. Always set to zero.
PITCH	Indicates the number of bytes in memory between two vertically adjacent pixels.

36.6.17 PS Background Color (PXP_PS_BACKGROUND)

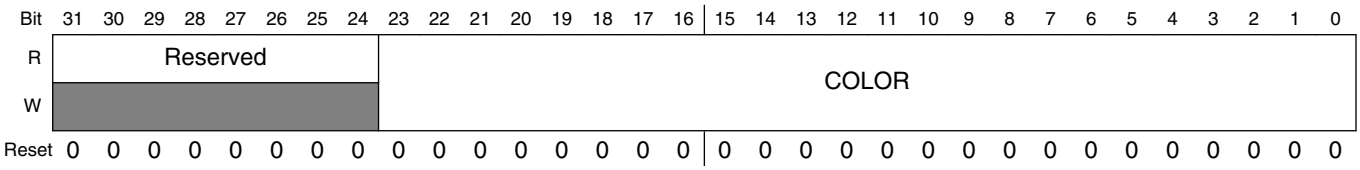
PS Background Pixel Color. This register provides a pixel value used when processing pixels outside of the region specified by the PS Coordinate registers. This value can effectively be used to set the color of the letterboxing region around the PS image.

This register contains a pixel value to be used for any PS pixels that fall outside the PS extents. This is effectively a background or letterbox color. The CSC1 control and datapath pixel format should be considered when selecting the background color.

EXAMPLE

```
PXP_PS_BACKGROUND_WR(0x00000000);  // letterbox is black
PXP_PS_BACKGROUND_WR(0x00800000);  // letterbox is dark red
PXP_PS_BACKGROUND_WR(0x00008000);  // letterbox is dark green
PXP_PS_BACKGROUND_WR(0x00000080);  // letterbox is dark blue
```

Address: 402B_4000h base + 100h offset = 402B_4100h



PXP_PS_BACKGROUND field descriptions

Field	Description
31–24 Reserved	This field is reserved. Always set to zero.
COLOR	Background color (in 24bpp format) for any pixels not within the buffer range specified by the PS ULC/ LRC.

36.6.18 PS Scale Factor Register (PXP_PS_SCALE)

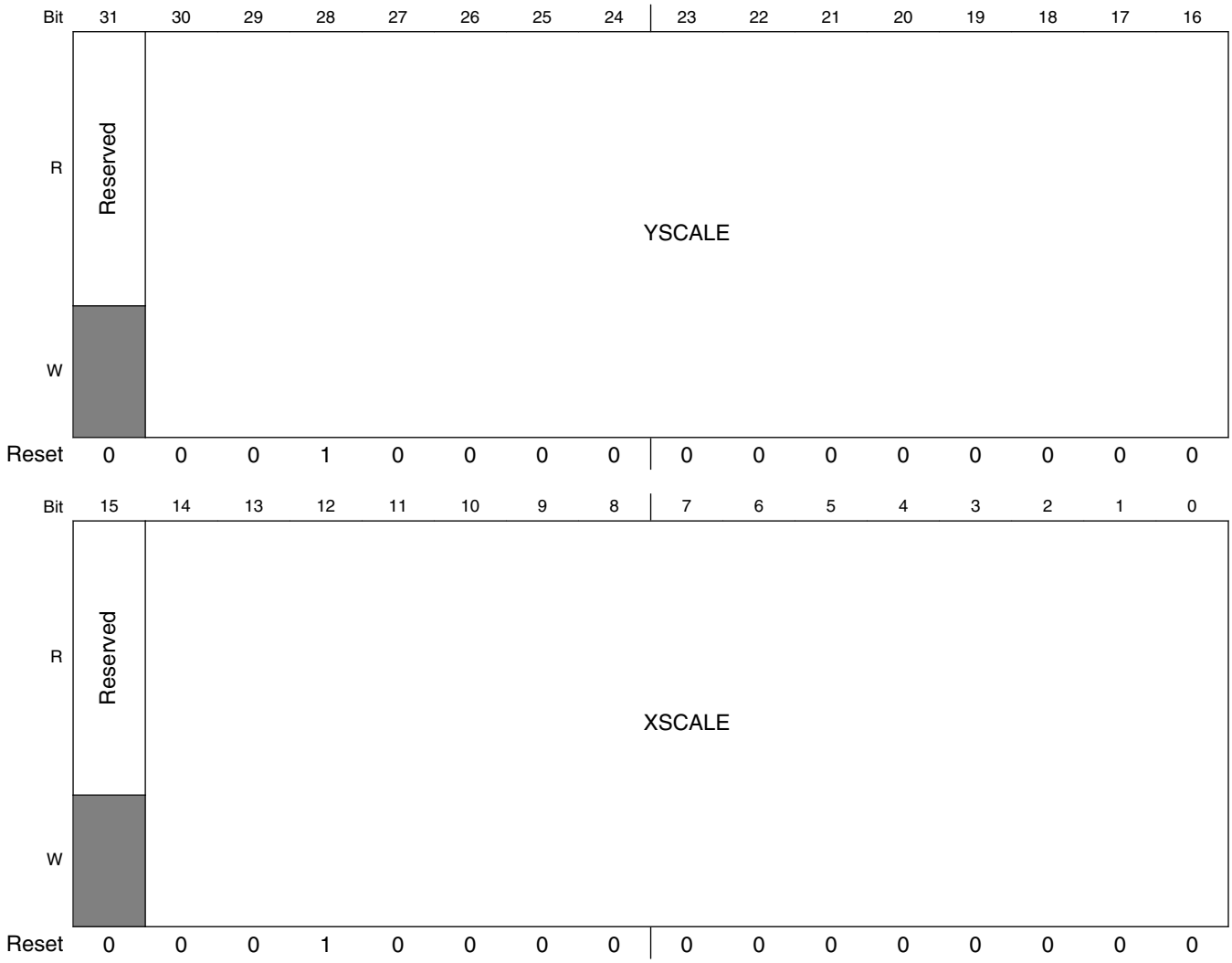
PS Scale Factor. This register provides the scale factor for the PS buffer.

The maximum down scaling factor is 1/2 such that the output image in either axis is 1/2 the size of the source. The maximum up scaling factor is 2^12 for either axis. The reciprocal of the scale factor should be loaded into this register. To reduce the PS buffer by a factor of two in the output frame buffer, a value of 10.0000_0000_0000 should be loaded into this register. To scale up by a factor of 4, the value of 1/4, or 00.0100_0000_0000, should be loaded into this register. To scale up by 8/5, the value of 00.1010_0000_0000 should be loaded.

EXAMPLE

```
PXP_PS_SCALE_WR(0x10001000);  // 1:1  scaling (0x1.000)
PXP_PS_SCALE_WR(0x08000800);  // 2x   scaling (0x0.800)
PXP_PS_SCALE_WR(0x20002000);  // 1/2x scaling (0x2.000)
```

Address: 402B_4000h base + 110h offset = 402B_4110h



PXP_PS_SCALE field descriptions

Field	Description
31 Reserved	This field is reserved. Always set to zero.
30–16 YSCALE	This is a two bit integer and 12 bit fractional representation (##.####_####_####) of the Y scaling factor for the PS source buffer. The maximum value programmed should be 2 since scaling down by a factor greater than 2 is not supported with the bilinear filter. Decimation and the bilinear filter should be used together to achieve scaling by more than a factor of 2.
15 Reserved	This field is reserved. Always set to zero.
XSCALE	This is a two bit integer and 12 bit fractional representation (##.####_####_####) of the X scaling factor for the PS source buffer. The maximum value programmed should be 2 since scaling down by a factor greater than 2 is not supported with the bilinear filter. Decimation and the bilinear filter should be used together to achieve scaling by more than a factor of 2.

36.6.19 PS Scale Offset Register (PXP_PS_OFFSET)

PS Scale Offset. This register provides the initial scale offset for the PS buffer.

The X and Y offset provides the ability to access the source image with a per sub-pixel granularity. This provides the capability to use all source pixels to effect the output PS image. The fixed offset values can be used for sub-pixel adjustments in the bilinear scaling filter. For example, when scaling an image down by a factor of 2, an initial offset of 0x0 would result in sub-sampling every other pixel. If a fixed offset of 0x800 (1/2), all pixels are used in scaling the final output pixel value. In this case, the first output pixel would be the sum of (1/2*P0) + (1/2*P1). This fixed offset is applied after the decimation filter stage, and before the bilinear filter stage.

EXAMPLE

```
PXP_PS_SCALE_WR(0x2000_2000);   // 1/2x scaling (0x2.000)
PXP_PS_OFFSET_WR(0x0800_0800);  // half-pixel offset in both X and Y to ensure averaging
versus pixel decimation
```

Address: 402B_4000h base + 120h offset = 402B_4120h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				YOFFSET												Reserved				XOFFSET											
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_PS_OFFSET field descriptions

Field	Description
31–28 Reserved	This field is reserved. Always set to zero.
27–16 YOFFSET	This is a 12 bit fractional representation (0.####_####_####) of the Y scaling offset. This represents a fixed pixel offset which gets added to the scaled address to determine source data for the scaling engine.
15–12 Reserved	This field is reserved. Always set to zero.
XOFFSET	This is a 12 bit fractional representation (0.####_####_####) of the X scaling offset. This represents a fixed pixel offset which gets added to the scaled address to determine source data for the scaling engine.

36.6.20 PS Color Key Low (PXP_PS_CLRKEYLOW)

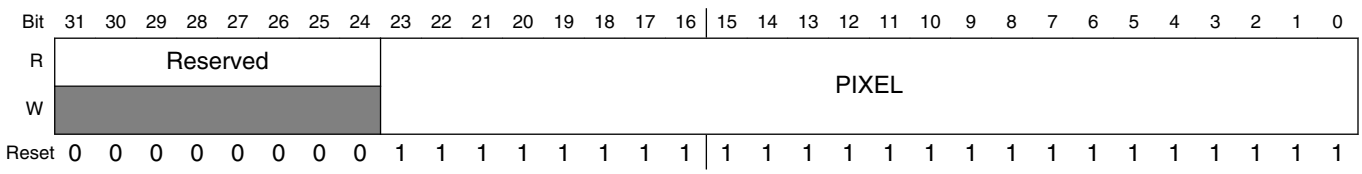
This register contains the color key low value for the PS buffer.

When processing an image, if the PXP finds a pixel in the PS buffer with a color that falls in the range between PXP_PS_CLRKEYLOW and PXP_PS_CLRKEYHIGH, it will insert the pixel from the AS channel. If the current AS pixel is letterboxed or if the AS also matches its colorkey range, the PXP_PS_BACKGROUND color is passed down the pixel pipeline.

EXAMPLE

```
                                    // colorkey values between
PXP_PS_CLRKEYLOW_WR (0x008000); // medium green and
PXP_PS_CLRKEYHIGH_WR(0x00FF00); // light green
```

Address: 402B_4000h base + 130h offset = 402B_4130h



PXP_PS_CLRKEYLOW field descriptions

Field	Description
31–24 Reserved	This field is reserved. Always set to zero.
PIXEL	Low range of color key applied to PS buffer. To disable PS colorkeying, set the low colorkey to 0xFFFFFF and the high colorkey to 0x000000.

36.6.21 PS Color Key High (PXP_PS_CLRKEYHIGH)

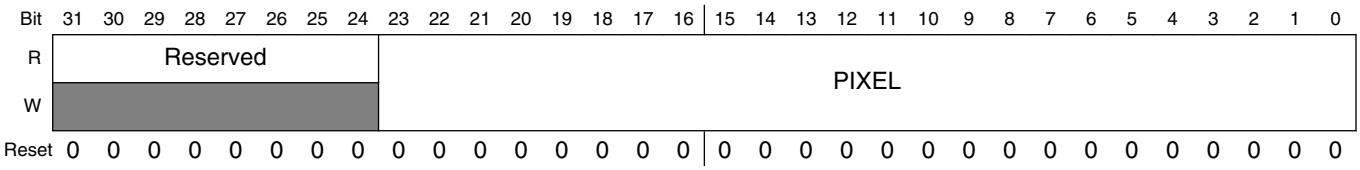
This register contains the color key high value for the PS buffer.

When processing an image, if the PXP finds a pixel in the PS buffer with a color that falls in the range between PXP_PS_CLRKEYLOW and PXP_PS_CLRKEYHIGH, it will insert the pixel from the AS channel. If the current AS pixel is letterboxed or if the AS also matches its colorkey range, the PXP_PS_BACKGROUND color is passed down the pixel pipeline.

EXAMPLE

```
                                    // colorkey values between
PXP_PS_CLRKEYLOW_WR (0x008000); // medium green and
PXP_PS_CLRKEYHIGH_WR(0x00FF00); // light green
```

Address: 402B_4000h base + 140h offset = 402B_4140h



PXP_PS_CLRKEYHIGH field descriptions

Field	Description
31–24 Reserved	This field is reserved. Always set to zero.
PIXEL	High range of color key applied to PS buffer. To disable PS colorkeying, set the low colorkey to 0xFFFFFF and the high colorkey to 0x000000.

36.6.22 Alpha Surface Control (PXP_AS_CTRL)

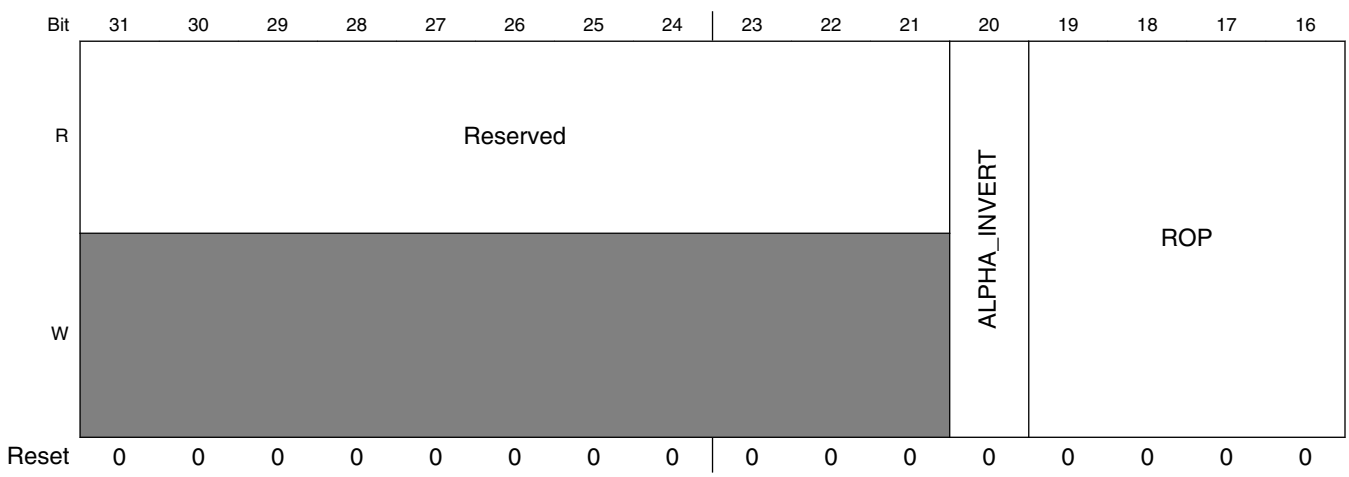
This register contains buffer control for the Alpha Surface 0 input buffer.

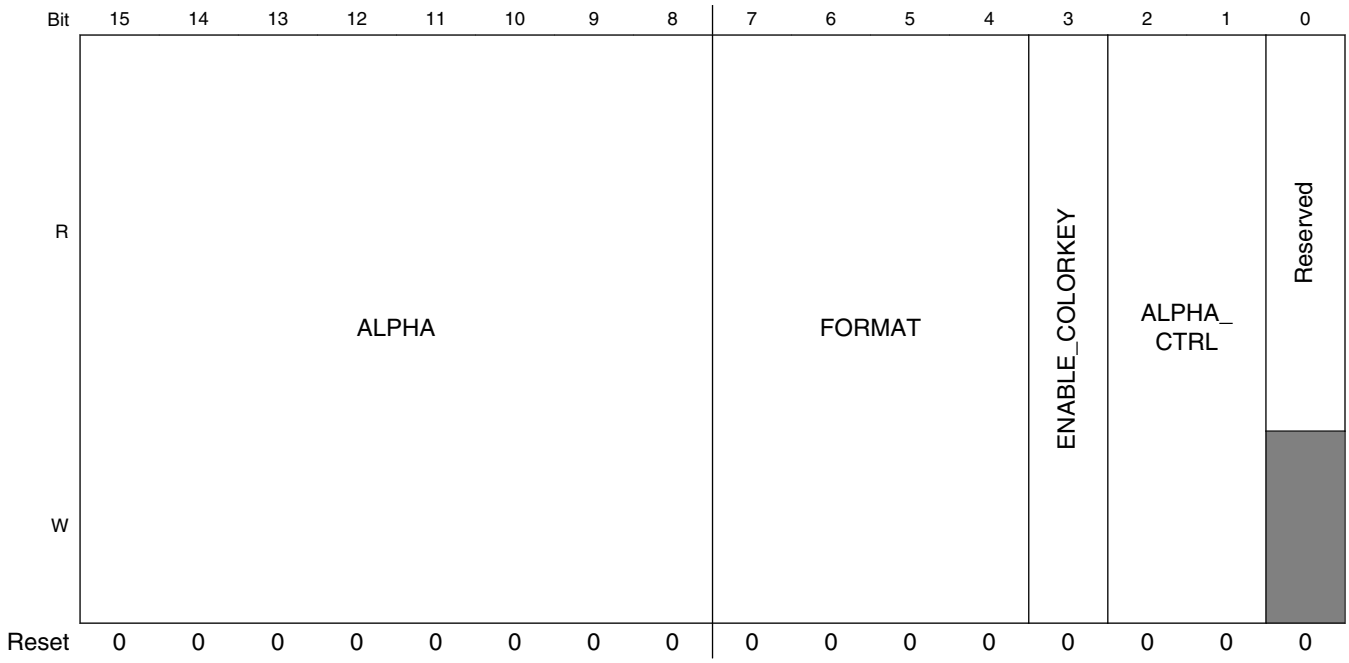
The Alpha Surface Parameter register provides additional controls for AS.

EXAMPLE

```
u32 asparam;
      asparam  = BF_PXP_ASPARAM_ENABLE    (1);
      asparam |= BF_PXP_ASPARAM_ALPHA_CTRL(BV_PXP_ASPARAM_ALPHA_CTRL__ROPs);
      asparam |= BF_PXP_ASPARAM_FORMAT    (BV_PXP_ASPARAM_FORMAT__ARGB8888);
      asparam |= BF_PXP_ASPARAM_ROP       (BV_PXP_ASPARAM_ROP__XORAS);
PXP_ASPARAM_WR(0,asparam);  // enable alpha surface to perform XOR ROP using RGB8888 AS
pixel format
```

Address: 402B_4000h base + 150h offset = 402B_4150h





PXP_AS_CTRL field descriptions

Field	Description
31–21 Reserved	This field is reserved. Always set to zero.
20 ALPHA_INVERT	Setting this bit to logic 0 will not alter the alpha value. A logic 1 will invert the alpha value and apply (1-alpha) for image composition. 0x0 **Not inverted** — 0x1 **Inverted** —
19–16 ROP	Indicates a raster operation to perform when enabled. Raster operations are enabled through the ALPHA_CTRL field. 0x0 **MASKAS** — AS AND PS 0x1 **MASKNOTAS** — nAS AND PS 0x2 **MASKASNOT** — AS AND nPS 0x3 **MERGEAS** — AS OR PS 0x4 **MERGENOTAS** — nAS OR PS 0x5 **MERGEASNOT** — AS OR nPS 0x6 **NOTCOPYAS** — nAS 0x7 **NOT** — nPS 0x8 **NOTMASKAS** — AS NAND PS 0x9 **NOTMERGEAS** — AS NOR PS 0xA **XORAS** — AS XOR PS 0xB **NOTXORAS** — AS XNOR PS
15–8 ALPHA	Alpha modifier used when the ALPHA_MULTIPLY or ALPHA_OVERRIDE values are programmed in PXP_AS_CTRL[ALPHA_CTRL]. The output alpha value will either be replaced (ALPHA_OVERRIDE) or scaled (ALPHA_MULTIPLY) when selected.
7–4 FORMAT	Indicates the input buffer format for AS.

Table continues on the next page...

PXP_AS_CTRL field descriptions (continued)

Field	Description
	0x0 **ARGB8888** — 32-bit pixels with alpha 0x1 **RGBA888** — 2-bit pixel with alpha at low 8 bits 0x4 **RGB888** — 32-bit pixels without alpha (unpacked 24-bit format) 0x8 **ARGB1555** — 16-bit pixels with alpha 0x9 **ARGB4444** — 16-bit pixels with alpha 0xA **RGBA5551** — 16-bit pixel with alpha at low 1 bit 0xB **RGBA4444** — 16-bit pixel with alpha at low 4 bits 0xC **RGB555** — 16-bit pixels without alpha 0xD **RGB444** — 16-bit pixels without alpha 0xE **RGB565** — 16-bit pixels without alpha
3 ENABLE_ COLORKEY	Indicates that colorkey functionality is enabled for this alpha surface. Pixels found in the alpha surface colorkey range will be displayed as transparent (the PS pixel will be used). 0x0 **Disabled** — 0x1 **Enabled** —
2–1 ALPHA_CTRL	Determines how the alpha value is constructed for this alpha surface. Indicates that the value in the ALPHA field should be used instead of the alpha values present in the input pixels. 0x0 **Embedded** — Indicates that the AS pixel alpha value will be used to blend the AS with PS. The ALPHA field is ignored. 0x1 **Override** — Indicates that the value in the ALPHA field should be used instead of the alpha values present in the input pixels. 0x2 **Multiply** — Indicates that the value in the ALPHA field should be used to scale all pixel alpha values. Each pixel alpha is multiplied by the value in the ALPHA field. 0x3 **ROPs** — Enable ROPs. The ROP field indicates an operation to be performed on the alpha surface and PS pixels.
0 Reserved	This field is reserved. Always set to zero.

36.6.23 Alpha Surface Buffer Pointer (PXP_AS_BUF)

Alpha Surface 0 Buffer Address Pointer. This register points to the beginning of the Alpha Surface 0 input buffer.

This register is used to indicate the base address of the AS buffer.

EXAMPLE

```
u32* alpha_ptr;
PXP_ASn_WR(0,alpha_ptr);
```

Address: 402B_4000h base + 160h offset = 402B_4160h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ADDR																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PXP_AS_BUF field descriptions

Field	Description
ADDR	Address pointer for the alpha surface 0 buffer.

36.6.24 Alpha Surface Pitch (PXP_AS_PITCH)

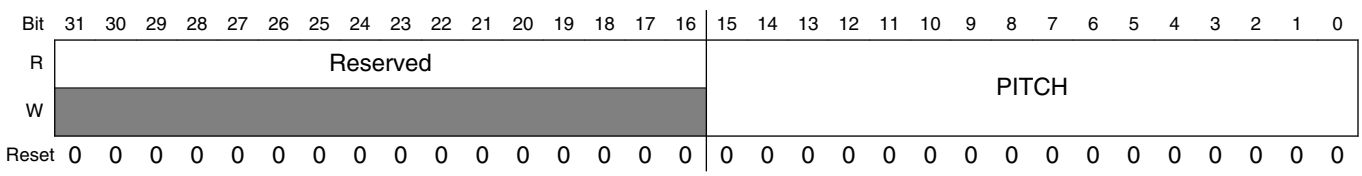
This register contains the alpha surface pitch in bytes.

Any byte value will indicate the vertical pitch. This value will be used in AS pixel address calculations. This value has no relation to the UL and LR registers. It specifies how many bytes are between two vertically adjacent pixels in the input AS surface.

EXAMPLE

```
PXP_AS_PITCH_WR( 1920 * 4 );  // The output buffer pitch is HD resolution at 32 bits per
pixel
```

Address: 402B_4000h base + 170h offset = 402B_4170h



PXP_AS_PITCH field descriptions

Field	Description
31–16 Reserved	This field is reserved. Always set to zero.
PITCH	Indicates the number of bytes in memory between two vertically adjacent pixels.

36.6.25 Overlay Color Key Low (PXP_AS_CLRKEYLOW)

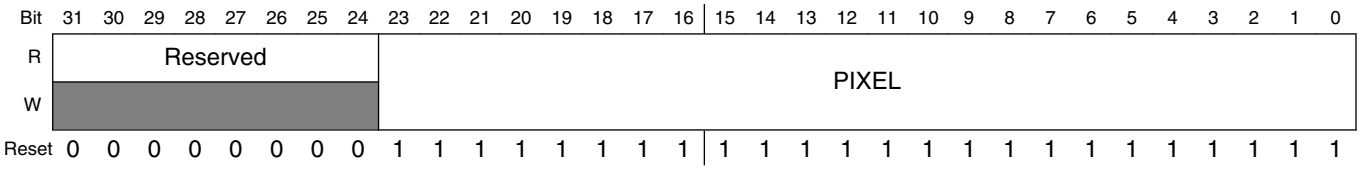
This register contains the color key low value for the AS buffer.

When processing an image, the if the PXP finds a pixel in the current overlay image with a color that falls in the range from the ASCOLORKEYLOW to ASCOLORKEYHIGH range, it will use the PS pixel value for that location. If no PS image is present or if the PS image also matches its colorkey range, the PS background color is used. Colorkey operations are higher priority than alpha or ROP operations.

EXAMPLE

```
                                    // colorkey values between
PXP_AS_CLRKEYLOW_WR (0x000000); // black and
PXP_AS_CLRKEYHIGH_WR(0x800000); // medium red
```

Address: 402B_4000h base + 180h offset = 402B_4180h



PXP_AS_CLRKEYLOW field descriptions

Field	Description
31–24 Reserved	This field is reserved. Always set to zero.
PIXEL	Low range of RGB color key applied to AS buffer. Each overlay has an independent colorkey enable.

36.6.26 Overlay Color Key High (PXP_AS_CLRKEYHIGH)

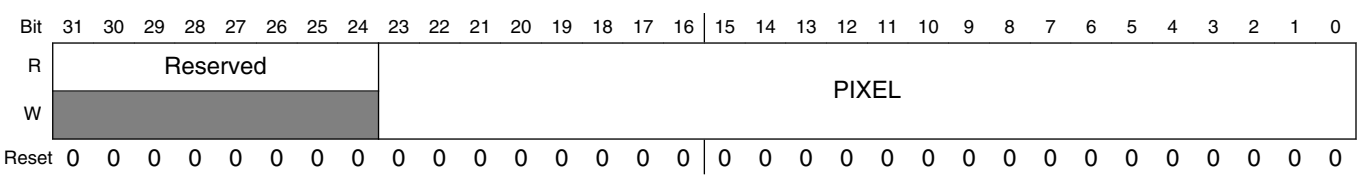
This register contains the color key high value for the AS buffer.

When processing an image, the if the PXP finds a pixel in the current overlay image with a color that falls in the range from the ASCOLORKEYLOW to ASCOLORKEYHIGH range, it will use the PS pixel value for that location. If no PS image is present or if the PS image also matches its colorkey range, the PS background color is used. Colorkey operations are higher priority than alpha or ROP operations.

EXAMPLE

```
                                    // colorkey values between
PXP_AS_CLRKEYLOW_WR (0x000000); // black and
PXP_AS_CLRKEYHIGH_WR(0x800000); // medium red
```

Address: 402B_4000h base + 190h offset = 402B_4190h



PXP_AS_CLRKEYHIGH field descriptions

Field	Description
31–24 Reserved	This field is reserved. Always set to zero.

Table continues on the next page...

PXP_AS_CLRKEYHIGH field descriptions (continued)

Field	Description
PIXEL	High range of RGB color key applied to AS buffer. Each overlay has an independent colorkey enable.

36.6.27 Color Space Conversion Coefficient Register 0 (PXP_CSC1_COEF0)

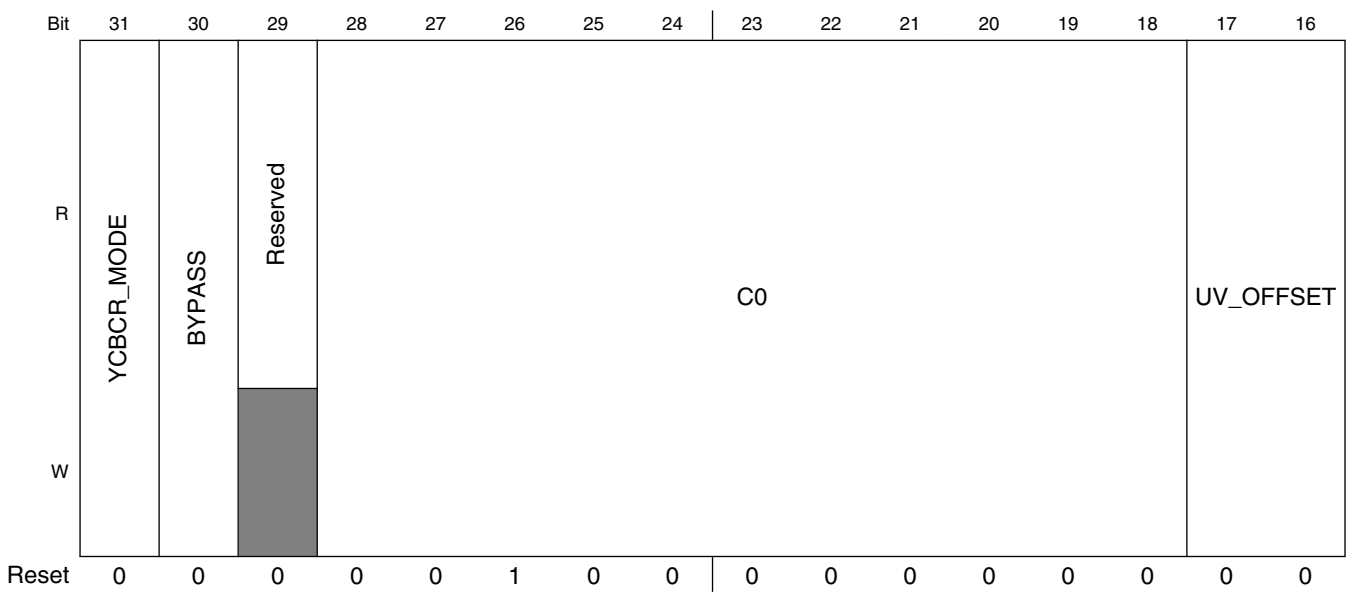
This register contains color space conversion coefficients in two's compliment notation.

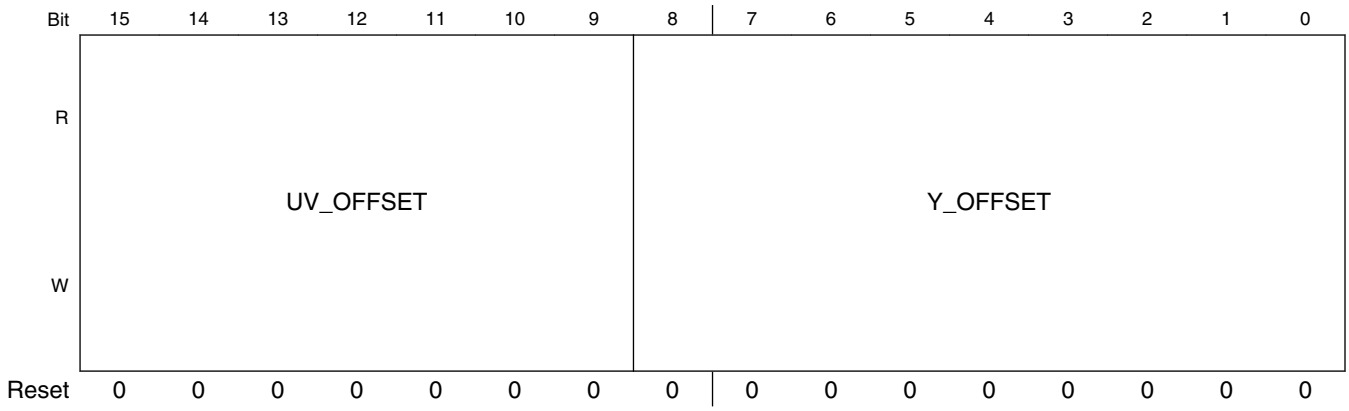
The coefficient 0 register contains coefficients used in the color space conversion algorithm. The Y and UV offsets are added to the source buffer to normalize them before the conversion. C0 is the coefficient that is used to multiply the luma component of the data for all three RGB components.

EXAMPLE

```
//  The equations used for Colorspace conversion are:
//     R = C0*(Y+YOFFSET)                    + C1(V+UV_OFFSET)
//     G = C0*(Y+YOFFSET) + C3(U+UV_OFFSET) + C2(V+UV_OFFSET)
//     R = C0*(Y+YOFFSET) + C4(U+UV_OFFSET)

   PXP_CSCCOEF0_WR(0x04030000); // YUV coefficients: C0, Yoffset, UVoffset
   PXP_CSCCOEF1_WR(0x01230208); // YUV coefficients: C1, C4
   PXP_CSCCOEF2_WR(0x076B079b); // YUV coefficients: C2, C3
```

Address: 402B_4000h base + 1A0h offset = 402B_41A0h





PXP_CSC1_COEF0 field descriptions

Field	Description
31 YCBCR_MODE	Set to 1 when performing YCbCr conversion to RGB. This bit changes the behavior of the scaler when performing U/V scaling. 0 YUV to RGB 1 YCbCr to RGB
30 BYPASS	Bypass the CSC unit in the scaling engine. When set to logic 1, bypass is enabled and the output pixels will be in the YUV/YCbCr color space. When set to logic 0, the CSC unit is enabled and the pixels will be converted based on the programmed coefficients.
29 Reserved	This field is reserved. Always set to zero.
28–18 C0	Two's compliment Y multiplier coefficient. YUV=0x100 (1.000) YCbCr=0x12A (1.164)
17–9 UV_OFFSET	Two's compliment phase offset implicit for CbCr data. Generally used for YCbCr to RGB conversion. YCbCr=0x180, YUV=0x000 (typically -128 or 0x180 to indicate normalized -0.5 to 0.5 range)
Y_OFFSET	Two's compliment amplitude offset implicit in the Y data. For YUV, this is typically 0 and for YCbCr, this is typically -16 (0x1F0)

36.6.28 Color Space Conversion Coefficient Register 1 (PXP_CSC1_COEF1)

This register contains color space conversion coefficients in two's compliment notation.

The Coeffient 1 register contains coefficients used in the color space conversion algorithm. C1 is the coefficient that is used to multiply the chroma (Cr/V) component of the data for the red component. C4 is the coefficient that is used to multiply the chroma (Cb/U) component of the data for the blue component. Both values should be coded as a two's compliment fixed point number with 8 bits right of the decimal.

EXAMPLE

```
PXP_CSCCOEF0_WR(0x04030000); // YUV coefficients: C0, Yoffset, UVoffset
PXP_CSCCOEF1_WR(0x01230208); // YUV coefficients: C1, C4
```

```
PXP_CSCCOEF2_WR(0x076B079b); // YUV coefficients: C2, C3
```

Address: 402B_4000h base + 1B0h offset = 402B_41B0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved					C1											Reserved					C4											
W																																
Reset	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0

PXP_CSC1_COEF1 field descriptions

Field	Description
31–27 Reserved	This field is reserved. Always set to zero.
26–16 C1	Two's compliment Red V/Cr multiplier coefficient. YUV=0x123 (1.140) YCbCr=0x198 (1.596)
15–11 Reserved	This field is reserved. Always set to zero.
C4	Two's compliment Blue U/Cb multiplier coefficient. YUV=0x208 (2.032) YCbCr=0x204 (2.017)

36.6.29 Color Space Conversion Coefficient Register 2 (PXP_CSC1_COEF2)

This register contains color space conversion coefficients in two's compliment notation.

The Coefficient 2 register contains coefficients used in the color space conversion algorithm. C2 is the coefficient that is used to multiply the chroma (Cr/V) component of the data for the green component. C3 is the coefficient that is used to multiply the chroma (Cb/U) component of the data for the green component. Both values should be coded as a two's compliment fixed point number with 8 bits right of the decimal.

EXAMPLE

```
// NOTE: The default values for the CSCCOEF2 register are incorrect.  C2 should be 0x76B and
C3 should be 0x79C for proper operation.

   PXP_CSCCOEF0_WR(0x04030000); // YUV coefficients: C0, Yoffset, UVoffset
   PXP_CSCCOEF1_WR(0x01230208); // YUV coefficients: C1, C4
   PXP_CSCCOEF2_WR(0x076B079b); // YUV coefficients: C2, C3
```

Address: 402B_4000h base + 1C0h offset = 402B_41C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved					C2											Reserved					C3											
W																																
Reset	0	0	0	0	0	1	1	1	1	0	0	1	1	0	1	1	0	0	0	0	0	1	1	1	0	1	1	0	1	1	0	0

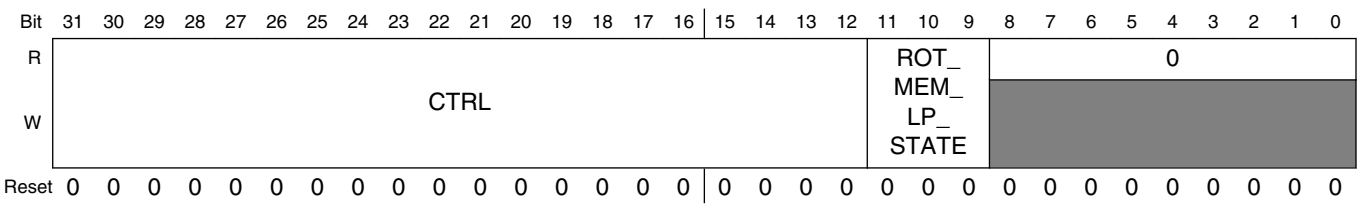
PXP_CSC1_COEF2 field descriptions

Field	Description
31–27 Reserved	This field is reserved. Always set to zero.
26–16 C2	Two's complement Green V/Cr multiplier coefficient. YUV=0x76B (-0.581) YCbCr=0x730 (-0.813)
15–11 Reserved	This field is reserved. Always set to zero.
C3	Two's complement Green U/Cb multiplier coefficient. YUV=0x79C (-0.394) YCbCr=0x79C (-0.392)

36.6.30 PXP Power Control Register (PXP_POWER)

This register controls power states for PXP memories.

Address: 402B_4000h base + 320h offset = 402B_4320h



PXP_POWER field descriptions

Field	Description
31–12 CTRL	Power control for the PXP.
11–9 ROT_MEM_LP_ STATE	Select the low power state of the Rotation (ROT) memory. 0x0 **NONE** — Memory is not in low power state. 0x1 **LS** — Light Sleep Mode. Low leakage mode, maintain memory contents. 0x2 **DS** — Deep Sleep Mode. Low leakage mode, maintain memory contents. 0x4 **SD** — Shut Down Mode. Shut Down periphery and core, no memory retention.
Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

36.6.31 Next Frame Pointer (PXP_NEXT)

This register contains a pointer to a data structure used to reload the PXP registers at the end of the current frame.

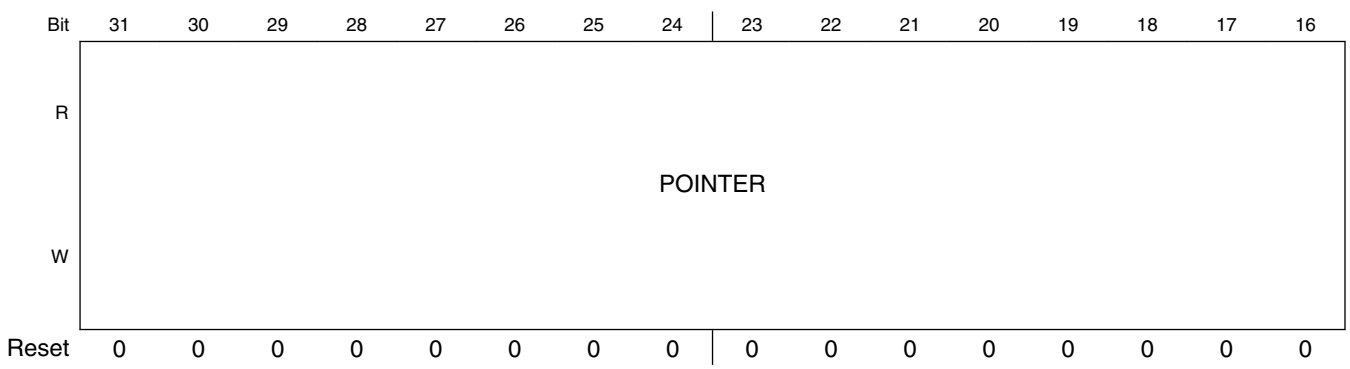
To enable this functionality, software must write this register while the PXP is processing the current data frame (if the PXP is currently idle, this will also initiate an immediate load of registers from the pointer). The process of writing this register (WRITE

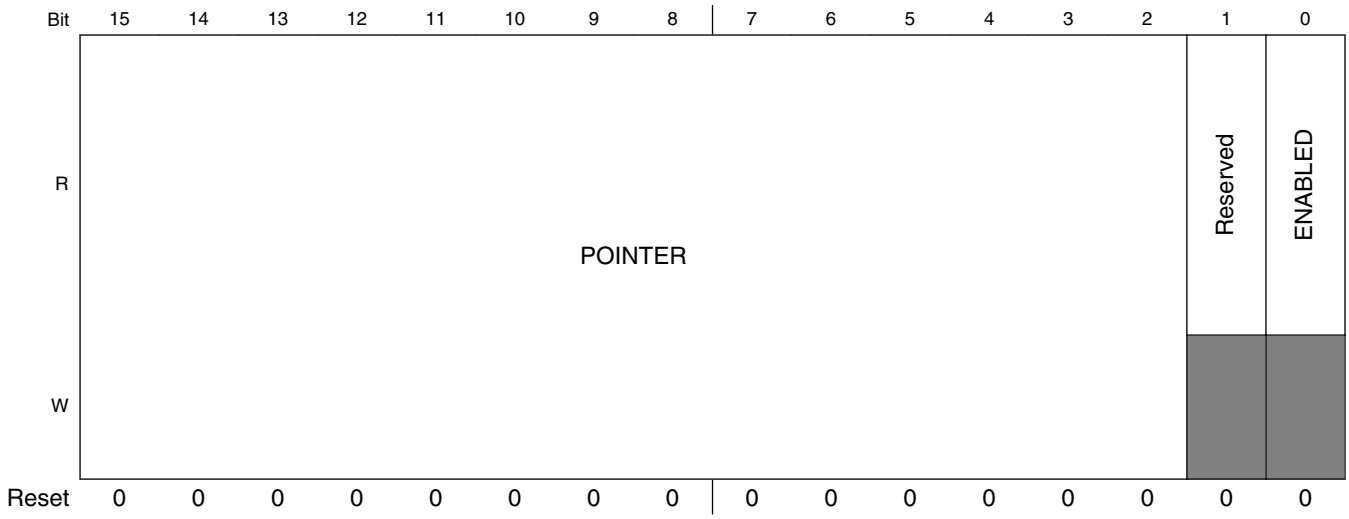
operation) will set a semaphore in hardware to notify the control logic that a register reload operation must be performed when the current frame processing is complete. At the end of a frame, the PXP will fetch the register settings from this location, signal an interrupt to software, then proceed with rendering the next frame of data. Software may cancel the reload operation by issuing a CLEAR operation to this register. SET and TOGGLE operations should not be used when addressing this register. All registers will be reloaded with the exception of the following: STAT, CSCCOEFn, NEXT. All other registers will be loaded in the order they appear in the register map. After the pointer's contents have been loaded into the PXP's registers, the NEXT_IRQ interrupt will be issued (see the PXP_STATUS register).

EXAMPLE

```
// create register command structure in memory
  u32* pxp_commands0[48], pxp_commands1;
  u32  rc;

// initialize control structure for frame 0
  pxp_commands0[0] = ...; // CTRL
  pxp_commands0[1] = ...; // OUT Buffer
  ...
  pxp_commands0[47] = ..; // Overlay7 param2

// initialize control structure for frame 1
  pxp_commands1[0] = ...; // CTRL
  pxp_commands1[1] = ...; // OUT Buffer
  ...
  pxp_commands1[47] = ..; // Overlay7 param2

// poll until a command isn't queued
  while (rc=PXP_NEXT_RD() & BM_PXP_NEXT_ENABLED );
  PXP_NEXT_WR(pxp_commands0);  // enable PXP operation 0 via command pointer

// poll until first command clears
  while (rc=PXP_NEXT_RD() & BM_PXP_NEXT_ENABLED );
  PXP_NEXT_WR(pxp_commands1);  // enable PXP operation 1 via command pointer
```

Address: 402B_4000h base + 400h offset = 402B_4400h



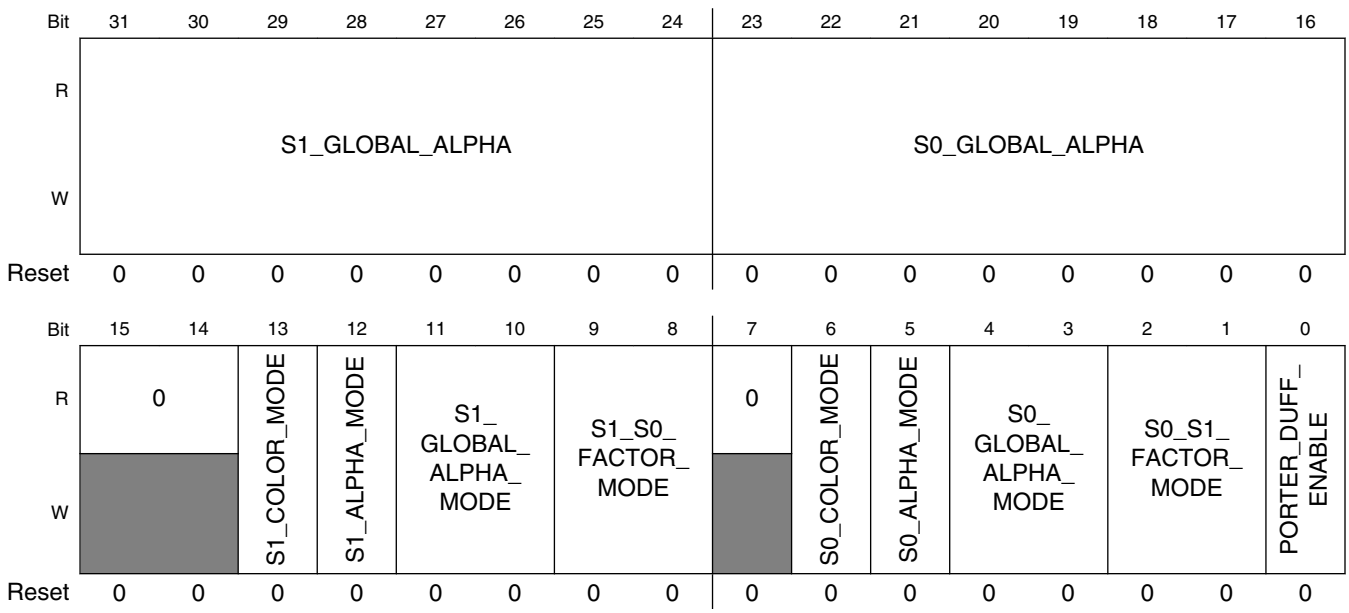


PXP_NEXT field descriptions

Field	Description
31–2 POINTER	A pointer to a data structure containing register values to be used when processing the next frame. The pointer must be 32-bit aligned and should reside in on-chip or off-chip memory.
1 Reserved	This field is reserved. Always set to zero.
0 ENABLED	Indicates that the "next frame" functionality has been enabled. This bit reflects the status of the hardware semaphore indicating that a reload operation is pending at the end of the current frame.

36.6.32 PXP Alpha Engine A Control Register. (PXP_PORTER_DUFF_CTRL)

Address: 402B_4000h base + 440h offset = 402B_4440h



PXP_PORTER_DUFF_CTRL field descriptions

Field	Description
31–24 S1_GLOBAL_ALPHA	s1 global alpha
23–16 S0_GLOBAL_ALPHA	s0 global alpha
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13 S1_COLOR_MODE	s1 color mode 0x0 **Original pixel** — 0x1 **Scaled pixel** —
12 S1_ALPHA_MODE	s1 alpha mode (Porter-Duff Alpha mode) 0x0 **Straight mode** — 0x1 **Inverted mode** —
11–10 S1_GLOBAL_ALPHA_MODE	s1 global alpha mode (Porter-Duff Global Alpha mode) 0x0 **Global alpha** — 0x1 **Local alpha** —

Table continues on the next page...

PXP_PORTER_DUFF_CTRL field descriptions (continued)

Field	Description
	0x2 **Scaled alpha** — 0x3 **Scaled alpha** —
9–8 S1_S0_ FACTOR_MODE	s1 to s0 factor mode (Porter-Duff factor mode) 0x0 **1** — 0x1 **0** — 0x2 **Straight alpha** — 0x3 **Inverse alpha** —
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 S0_COLOR_ MODE	s0 color mode (Porter-Duff color mode) 0x0 **Original pixel** — 0x1 **Scaled pixel** —
5 S0_ALPHA_ MODE	s0 alpha mode (Porter-Duff alpha mode) 0x0 **Straight mode** — 0x1 **Inverted mode** —
4–3 S0_GLOBAL_ ALPHA_MODE	s0 global alpha mode 0x0 **Global alpha** — 0x1 **Local alpha** — 0x2 **Scaled alpha** — 0x3 **Scaled alpha** —
2–1 S0_S1_ FACTOR_MODE	s0 to s1 factor mode 0x0 **1** — 0x1 **0** — 0x2 **Straight alpha** — 0x3 **Inverse alpha** —
0 PORTER_ DUFF_ENABLE	Porter-Duff Enable 0x0 **Disabled** — 0x1 **Enabled** —

Chapter 37
Audio Overview

37.1 Audio Overview

The audio subsystem consists of the following modules: SAI-1, SAI-2, SAI-3, SPDIF and MQS. In addition, the IOMUX must be appropriately configured to get signals in and out of the chip.

Audio Module Overview provides an overview of each of the audio modules, followed by a module-specific section.

37.1.1 Audio Module Overview

The following figure shows a high level block diagram of the audio subsystem.

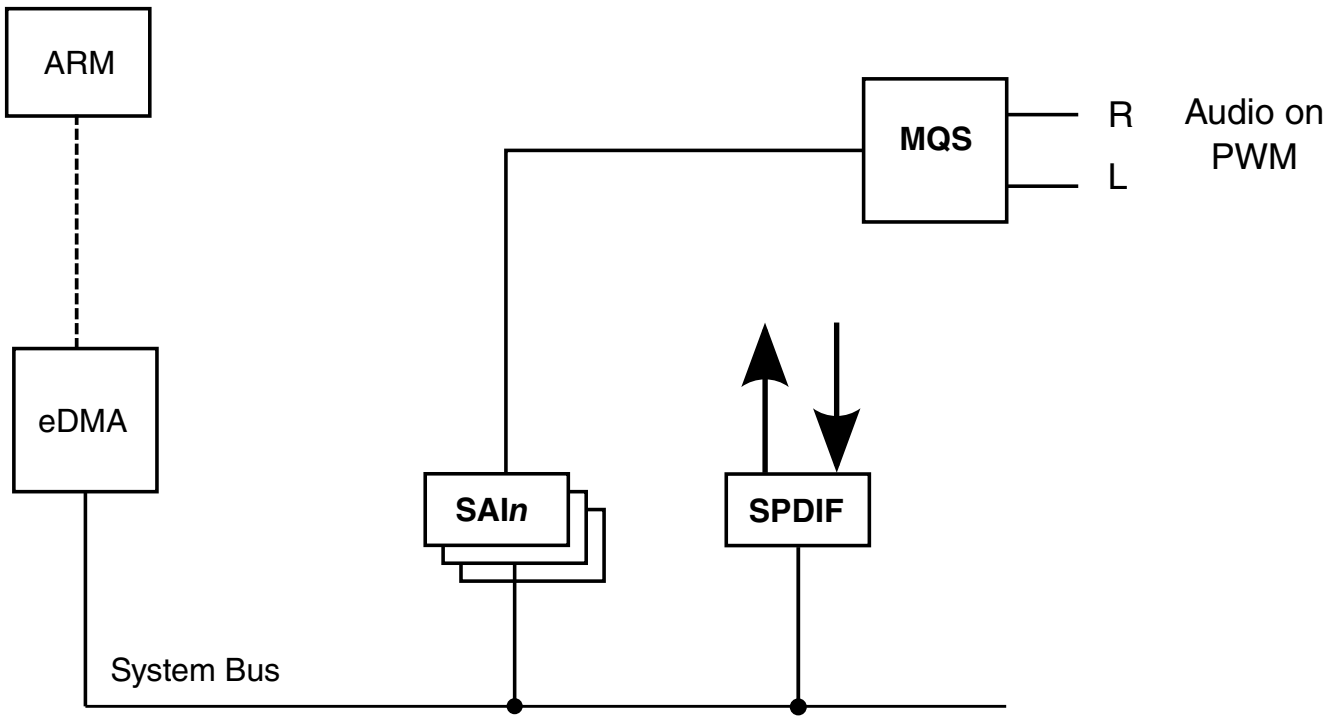


Figure 37-1. Audio peripherals block diagram

SAI*n* are synchronous serial interfaces used to transfer audio data. They can be accessed by both the eDMA and Arm CPUs. Their input/output are connected to the pads through IOMUX.

MQS (medium quality speaker) is used to convert the I2S audio data from SAI to PWM signals that can drive external speaker directly. Its audio source comes from SAI-3 and its output is connected to pads through IOMUX.

The Sony/Philips digital interface (SPDIF) audio module is a stereo transceiver that allows the processor to receive and transmit digital audio over it. The SPDIF receiver section includes a frequency measurement block that allows the precise measurement of an incoming sampling frequency. A recovered clock is provided by the SPDIF receiver section and may be used to drive both internal and external components in the system.

The audio interfaces are summarized as the table below.

Table 37-1. Audio Interface Summary

Interface	Function	RX Data Line	TX Data Line
SAI-1	External audio	4	4
SAI-2	External audio	1	1
SAI-3	External audio	1	1
MQS	External audio	0	2

Table continues on the next page...

Table 37-1. Audio Interface Summary (continued)

Interface	Function	RX Data Line	TX Data Line
SPDIF	External audio	1	1

SAI-1 is used for multi-channel audio interface, which supports up to 8-channels audio input or 8-channels audio output at 384 kHz/32-bit. SAI-2, SAI-3 can be used for stereo audio input and output up to 384 kHz/32-bit. Also SAI-3 is able to drive MQS directly as a low-cost audio output.

To reduce IO count and keep the flexibilities of supporting multiple RX and TX data lines application, SAI-1 has following options on data pin multiplexing. The multiplexing is done in the IOMUX. The following table is not an exhastive list of options, but does show that there can be up to 5 SAI-1 RX/TX data pins.

Table 37-2. SAI-1 TX/RX Data Pin Multiplexing

Options	Number of RX Data Pins	Number of TX Data Pins
0	1	4
1	2	3
2	3	2
3	4	1

SAI-1 to IOMUXC connection is implemented as shown in the diagram below.

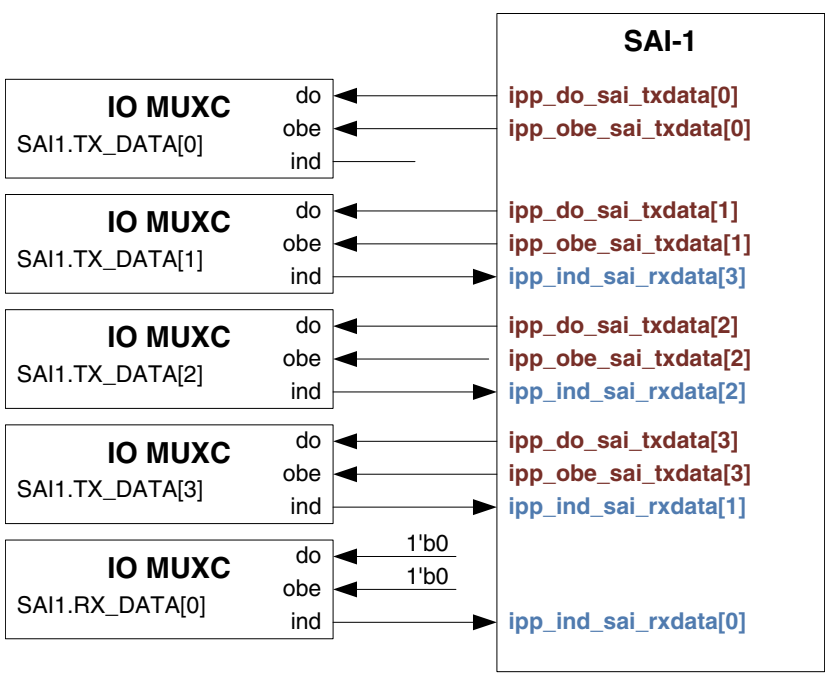


Figure 37-2. SAI-1 TX/RX Data Multiplexing

In the SAI-1 TX/RX Data Multiplexing figure:

- 'do' in the signal names indicate data transmitted to the PAD from SAI
- 'ind' in the signal name indicate data sensed from PAD and received by SAI
- 'obe' in the signal name indicate transmit enable

For example, if a pin is configured to use SAI-1 TXD3, then the PAD level is driven from SAI-1 TXD3 when transmit is enabled (obe=1). When the signal works as a receive function, the PAD level is sensed and sent to SAI-1 RXD1.

Detailed clock multiplexing scheme is shown in the following figure.

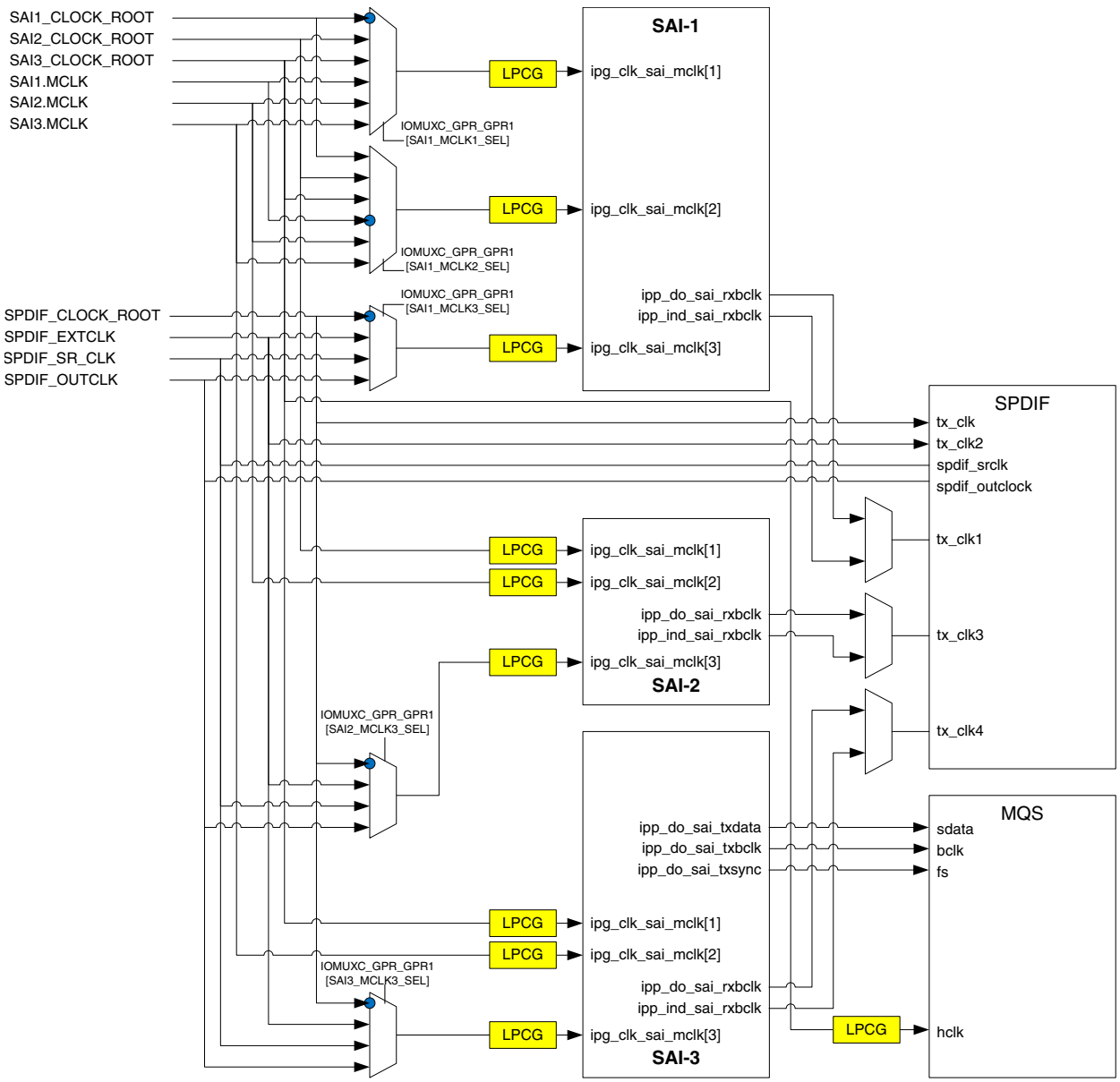


Figure 37-3. Audio subsystem clocking diagram

37.1.2 Medium Quality Sound (MQS)

MQS is used to generate medium quality audio via a standard GPIO in the pinmux. The user can connect stereo speakers or headphones to a power amplifier without an additional DAC chip.

- 2-channel, LSB-valid 16 bit, MSB shift-out first serial data (sdata)
- Frame sync aligned with the left channel data
- 44.1 kHz or 48 kHz I2S signals from SAI3
- SNR target as no more than 20 dB for the signals below 10 kHz
- Signals over 10 kHz have worse THD+N values

37.1.3 Synchronous Audio Interface (SAI)

- Transmitter with independent Bit Clock and Frame Sync supporting 1 data line
- Receiver with independent Bit Clock and Frame Sync supporting 1 data line
- Maximum Frame Size of 32 Words
- Word size programmable from 8-bits to 32-bits
- Word size configured separately for first word and remaining words in frame
- Asynchronous FIFO for each Transmit and Receive data line
- Graceful restart after FIFO Error

37.1.4 Sony/Philips Digital Interface (SPDIF)

The Sony/Philips Digital Interface (SPDIF) module is a stereo transceiver that allows the processor to receive and transmit digital audio over it using the IEC60958 standard, consumer format. The chip provides a single SPDIF receiver with one input, and one SPDIF transmitter with one output.

The SPDIF transceiver allows the handling of both SPDIF channel status (CS) and User (U) data and features a frequency measurement block that allows the precise measurement of the incoming sampling frequency.

The clock recovered by the SPDIF receiver is provided to drive both internal and external components in the system such as the SPDIF transmitter, SAI ports, as well as external A/Ds or D/As, with clocking control provided via related registers.

The SPDIF is composed of two parts: SPDIF Receiver and SPDIF Transmitter. The SPDIF receiver extracts the audio data from each SPDIF frame and places the data in two 16-word-deep FIFOs, one FIFO for the left channel, the other FIFO for the right channel.

The FIFOs support programmable watermark levels so that FIFO Full service request can be triggered when the combined number of data words stored in both FIFOs is 2, 8, 16 or 32 words. It is recommended to program the watermark level to trigger a FIFO Full service request when 16 word locations are filled. For optimal performance when servicing the FIFO Full service request, the FIFOs should be read alternately, starting with the left channel FIFO. The Channel Status and User Bits are also extracted from each frame and placed in the corresponding registers. The SPDIF receiver provides a bypass option for direct transfer of the SPDIF input signal to the SPDIF transmitter.

For the SPDIF transmitter, the audio data is provided by the processor via the SPDIFTxLeft and SPDIFTxRight registers, and the data is stored in two 16-word-deep FIFOs, one for the right channel, the other for the left channel. The FIFOs support programmable watermark levels so that FIFO Empty service request can be triggered when the combined number of empty data words locations in both FIFOs is 8, 16, 24 or 32 words. It is recommended to program the watermark level to trigger a FIFO Empty service request when 16 word locations are empty. For optimal performance when servicing the FIFO Empty service request, the FIFOs should be written alternately, starting with the left channel FIFO. The Channel Status bits are also provided via the corresponding registers. The SPDIF transmitter generates an SPDIF output bitstream in the biphase mark format (IEC 60958), which consists of audio data, channel status and user bits.

The data handled by the SPDIF module is 24-bit wide. The 24-bit SPDIF data is aligned in the 24 least significant bits of the 32-bit shared peripheral bus data word. The 8 most significant bits of the 32-bit word are ignored by the SPDIF Transmitter when data is being stored in the Transmit FIFOs from the peripheral bus. The 8 most significant bits of the 32-bit word are zeroed by the SPDIF Receiver module when the data is being read from the Receiver FIFOs to the peripheral bus.

Note that 16-bit data is left-aligned in the 24-bit word format of the SPDIF. This means that when receiving 16-bit data, it will be located in the middle two bytes of the 32-bit peripheral bus data word, while the 8 bits of the MSB and the 8 bits of the LSB will be zero. When 16-bit data is to be transmitted, the 32-bit word to be written to the SPDIF Transmit FIFOs should be created as follows: the 16-bit data should be located in the middle two bytes of the 32-bit data word and the 8 bits of the LSB must be set to zero, while the 8 bits of the MSB will be ignored.

The SPDIF Transmit clock is generated by the SPDIF internal clock generator module and the clock sources are from outside of the SPDIF block. The clock sources should provide a clock that is at least 64 x Fs, where Fs is the sampling frequency. The external clock source should provide at least 128 x Fs. Clocks of higher frequency may be provided as long as the multiplication factor is a power of 2 (for example, 128x, 256x or 512x). Also, clock frequency precision of 100ppm or better should be provided.

Chapter 38
Synchronous Audio Interface (SAI)

38.1 Chip-specific SAI information

Table 38-1. Reference links to related information

Topic	Related module or subsystem	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Audio Subsystem	Audio Subsystem	Audio Subsystem Overview
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
Interrupts, DMA Events and XBAR Assignments	-	Interrupts, DMA Events and XBAR Assignments

NOTE

Number of TX and RX data lines for SAI1 and SAI2/SAI3 are 4, and 1 respectively.

NOTE

In this device, it does NOT support synchronous mode between different SAI peripherals.

MCLK sources are determined outside of the SAI block, as defined in the below table. For additional IOMUXC GPR information, please see the IOMUXC chapter. For more details about the features and clocking of the different SAI instances, see the Audio Overview chapter.

Table 38-2. SAI MCLK Information

SAI Instance	MCLK	MCLK sources
SAI1	MCLK1/MCLK2	From IOMUX GPR1: • sai1_clk_root • sai2_clk_root • sai3_clk_root • sai1_ipg_clk_sai_mclk • sai2_ipg_clk_sai_mclk • sai3_ipg_clk_sai_mclk
	MCLK3	From IOMUX GPR1: • spdif0_clk_root • spdif_tx_clk2 • spdif_srclk • spdif_outclock
SAI2	MCLK1	sai2_clk_root
	MCLK2	SAI2_MCLK (pin)
	MCLK3	From IOMUX GPR1: • spdif0_clk_root • spdif_tx_clk2 • spdif_srclk • spdif_outclock
SAI3	MCLK1	sai3_clk_root
	MCLK2	SAI3_MCLK (pin)
	MCLK3	From IOMUX GPR1: • spdif0_clk_root • spdif_tx_clk2 • spdif_srclk • spdif_outclock

38.2 Overview

The Synchronous Audio Interface (SAI) provides an interface that supports full-duplex serial interfaces with frame synchronization such as I^2S, AC97, TDM, and codec/DSP interfaces.

38.2.1 Block diagram

The following block diagram also shows the module clocks.

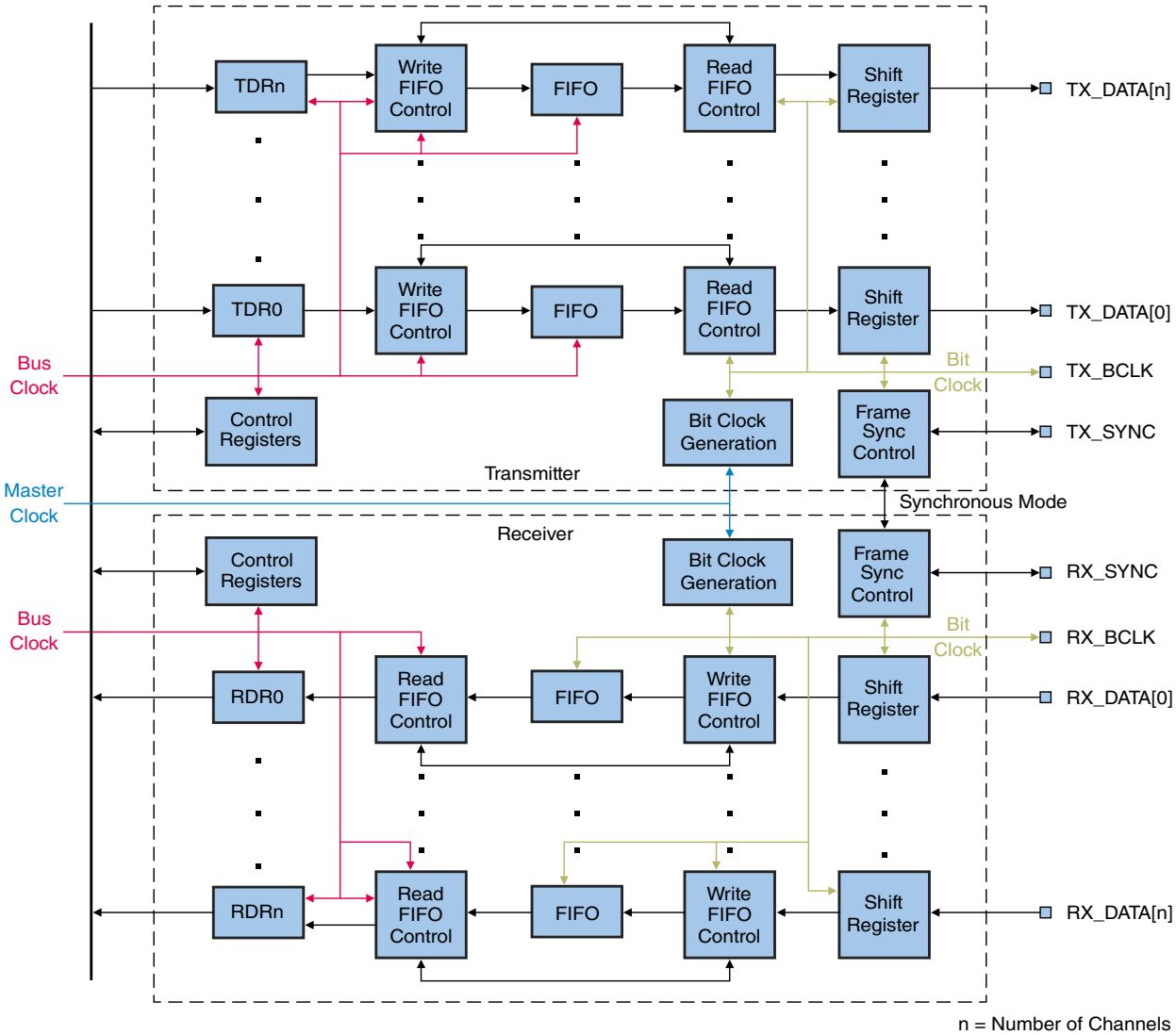


Figure 38-1. SAI block diagram

38.2.2 Features

Some of the features are not supported across all SAI instances; see the chip-specific SAI information in the first section of this chapter.

SAI1 pins: 1 is transmit only, 1 is receive only, and 3 can either transmit or receive. So you don't get 4 transmit and 4 receive, at this implies.

- Transmitter with independent bit clock and frame sync supporting 4 data lines
- Receiver with independent bit clock and frame sync supporting 4 data lines
- Each data line can support a maximum frame size of 32 words
- 8- to 32-bit word size
- Word size configured separately for the first word and remaining words in a frame
- Asynchronous 32 x 32-bit FIFO for each transmit and receive data line supporting:
  - Graceful restart after FIFO error

- Automatic restart after FIFO error without software intervention
- 8-bit and 16-bit data packing into each 32-bit FIFO word
- Multiple data line FIFOs combining into single data line FIFO

38.3 Functional description

This section provides a complete functional description of the block.

38.3.1 Modes of operation

Module power modes include Run mode, Stop modes, and Debug mode.

38.3.1.1 Run mode

In Run mode, the SAI transmitter and receiver operate normally.

38.3.1.2 Stop modes

In Stop mode, the SAI transmitter and/or receiver can continue operating provided the appropriate Stop Enable bit = 1 (TCSR[STOPE] and/or RCSR[STOPE], respectively), and provided the transmitter and/or receiver is/are using an externally generated bit clock or an Audio Master Clock that remains operating in Stop mode. The SAI transmitter and/or receiver can generate an asynchronous interrupt to wake the CPU from Stop mode.

In Stop mode, if the Transmitter Stop Enable (TCSR[STOPE]) bit is clear, the transmitter is disabled after completing the current transmit frame, and, if the Receiver Stop Enable (RCSR[STOPE]) bit is clear, the receiver is disabled after completing the current receive frame. Entry into Stop mode is prevented (not acknowledged) while waiting for the transmitter and receiver to be disabled at the end of the current frame.

38.3.1.3 Debug mode

In Debug mode, the SAI transmitter and/or receiver continues operating if the Debug Enable bit is set. When TCSR[DBGE] or RCSR[DBGE] bit is clear and Debug mode is entered, the SAI is disabled after completing the current transmit or receive frame. Debug mode does not affect the transmitter and receiver bit clocks.

38.3.2 Synchronous modes

The SAI transmitter and receiver can operate synchronously to each other.

38.3.2.1 Synchronous mode

The SAI transmitter and receiver can be configured to operate with synchronous bit clock and frame sync.

If both the transmitter and receiver use the transmitter bit clock and frame sync:

- Configure the transmitter for asynchronous operation and the receiver for synchronous operation.
- Enable the receiver in synchronous mode only after both the transmitter and receiver are enabled.
- Enable the transmitter last and disable the tranmitter first.

If both the transmitter and receiver use the receiver bit clock and frame sync:

- Configure the receiver for asynchronous operation and the transmitter for synchronous operation.
- Enable the transmitter in synchronous mode only after both the receiver and transmitter are enabled.
- Enable the receiver last and disable the receiver first.

When operating in synchronous mode, the transmitter and receiver share only the bit clock, frame sync, and transmitter/receiver enable. Otherwise, the transmitter and receiver operate independently, although the configuration registers must be configured consistently across both the transmitter and receiver.

38.3.3 Frame sync configuration

When enabled, the SAI continuously transmits and/or receives frames of data. Each frame consists of a fixed number of words and each word consists of a fixed number of bits. Within each frame, any given word can be masked causing the receiver to ignore that word and the transmitter to tri-state for the duration of that word.

The frame sync signal indicates the start of each frame. A valid frame sync requires a rising edge (if active high) or falling edge (if active low) to be detected and the transmitter or receiver cannot be busy with a previous frame. A valid frame sync is also ignored (slave mode) or not generated (master mode) for the first four bit clock cycles after enabling the transmitter or receiver.

The transmitter and receiver frame sync can be configured independently with any of the following options:

- Generate externally or internally
- Require active high or active low
- Assert with the first bit in frame or assert one bit early
- Assert for a duration of between 1 bit clock and the first word length
- Set frame length from 1 to 32 words per frame
- Support word length of 8 to 32 bits per word
    - First word length and remaining word lengths can be configured separately
- Transmit/Receive words MSB first or LSB first

These configuration options cannot be changed after enabling the SAI transmitter or receiver.

38.3.4 Data FIFO

Each transmit and receive channel includes a 32 x 32-bit size FIFO. To access the FIFO data use the SAI Transmit/Receive Data Registers.

38.3.4.1 Data alignment

Data in the FIFO can be aligned anywhere within the 32-bit wide register through the use of the First Bit Shifted configuration field, which selects the bit index (between 31 and 0) of the first bit shifted.

Examples of supported data alignment and the required First Bit Shifted configuration are illustrated in Figure 38-2 for LSB First configurations and Figure 38-3 for MSB First configurations.

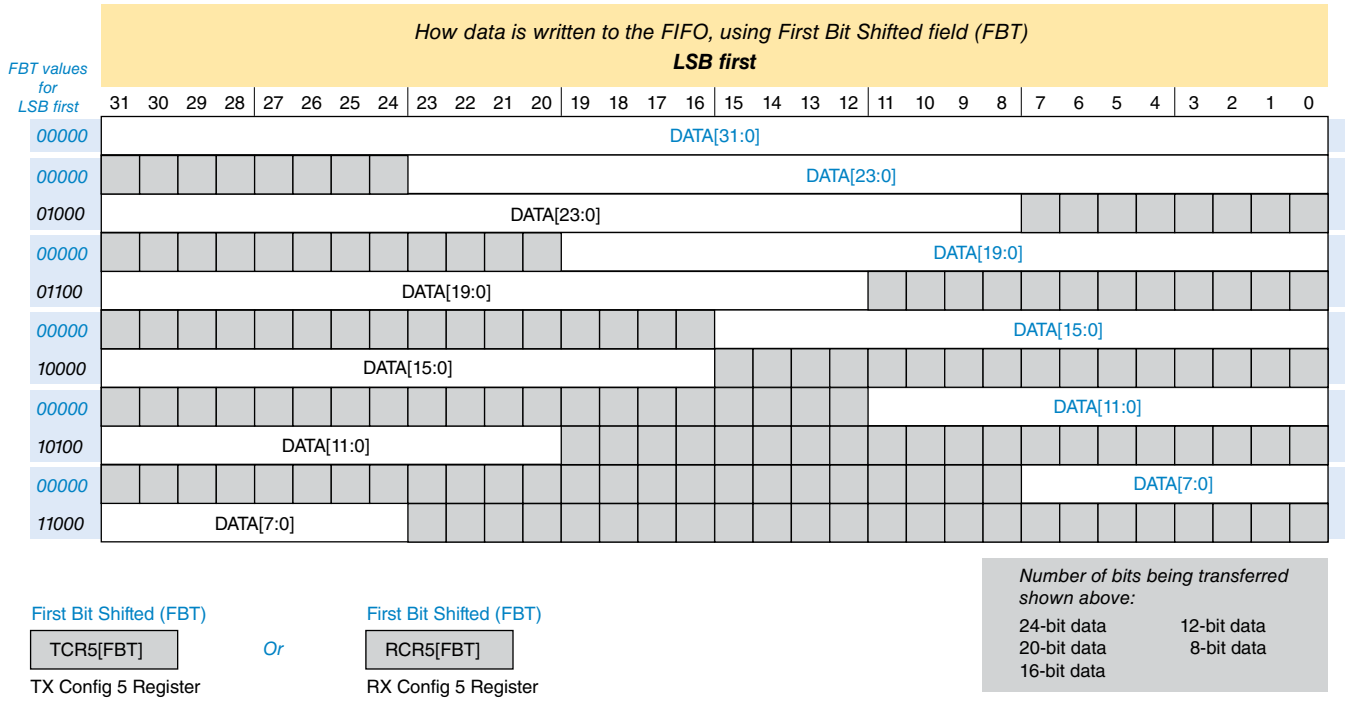


Figure 38-2. SAI first bit shifted, LSB first

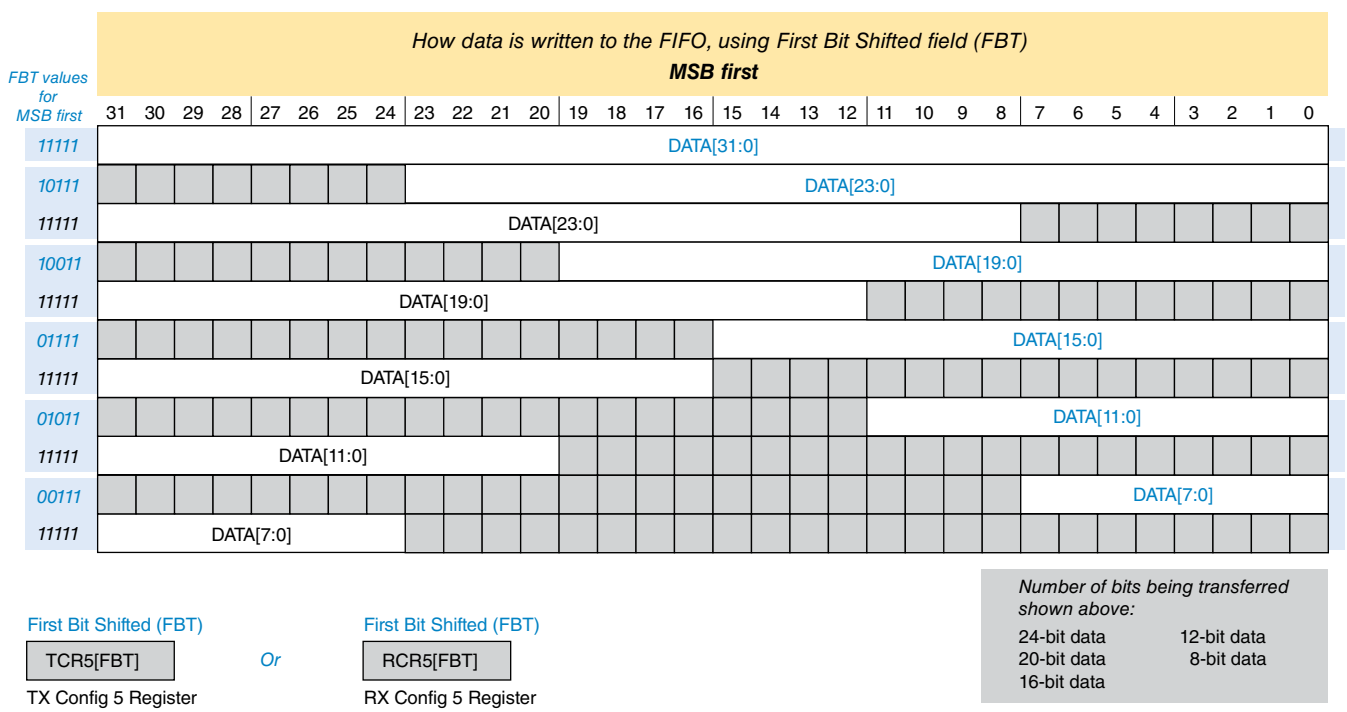


Figure 38-3. SAI first bit shifted, MSB first

38.3.4.2 FIFO pointers

When writing to a Transmit Data Register (TDR*n*), the Write FIFO Pointer (WFP) of the corresponding Transmit FIFO Register (TFR*n*) increments after each valid write. The SAI supports 8-bit, 16-bit, and 32-bit writes to the Transmit Data Register and the FIFO pointer increments after each individual write. Note that 8-bit writes should only be used when transmitting up to 8-bit data; 16-bit writes should only be used when transmitting up to 16-bit data.

- If the Transmit FIFO is full, writes to a Transmit Data Register are ignored.
- If the Transmit FIFO is empty, to avoid a FIFO underrun, write to the Transmit Data Register at least 3 bit clocks before the start of the next unmasked word. Before enabling the transmitter, initialize the Transmit FIFO with data (because after enabling the transmitter, the transmitter starts a new frame, and if no data is in the FIFO, then the transmitter immediately give an error).

When reading a Receive Data Register (RDR*n*), the Read FIFO Pointer (RFP) of the corresponding Receive FIFO Register (RFR*n*) increments after each valid read. The SAI supports 8-bit, 16-bit and 32-bit reads from the RDR and the FIFO pointer will increment after each individual read. Note that 8-bit reads should only be used when receiving up to 8-bit data; 16-bit reads should only be used when receiving up to 16-bit data.

- If the Receive FIFO is empty, reads from a Receive Data Register are ignored.
- If the Receive FIFO is full, to avoid a FIFO overrun, read the Receive Data Register at least 3 bit clocks before the end of an unmasked word.

38.3.4.3 FIFO packing

FIFO packing supports storing multiple 8-bit or 16-bit data words in one 32-bit FIFO word for the transmitter and/or receiver. While this can be emulated by adjusting the number of bits per word and number of words per frame (for example, one 32-bit word per frame versus two 16-bit words per frame), FIFO packing does not require even multiples of words per frame and fully supports word masking. When FIFO packing is enabled, the FIFO pointers only increment when the full 32-bit FIFO word has been written (transmit) or read (receive) by software, supporting scenarios where different words within each frame are loaded/stored in different areas of memory.

Using 16-bit FIFO packing for transmit, the transmit shift register loads at the start of each frame and after every second unmasked transmit word. The first word transmitted is taken from 16-bit word at byte offset 0x0 (the first bit is selected by TCR5[FBT] and must be configured within this 16-bit word) and the second word transmitted is taken from the 16-bit word at byte offset 0x2 (the first bit is selected by TCR5[FBT][3:0]). The transmitter transmits logic zero until the start of the next word once the 16-bit word has been transmitted.

Using 16-bit FIFO packing for receive, the receive shift register is stored after every second unmasked received word, and at the end of each frame if there is an odd number of unmasked received words in each frame. The first word received is stored in the 16-bit word at byte offset 0x0 (the first bit is selected by RCR5[FBT] and must be configured within this 16-bit word) and the second word received is stored in the 16-bit word at byte offset 0x2 (the first bit is selected by RCR5[FBT][3:0]). The receiver ignores received data until the start of the next word once the 16-bit word has been received.

The 8-bit FIFO packing is similar to 16-bit packing except four words are loaded or stored into each 32-bit FIFO word. The first word is loaded/stored in byte offset 0x0, second word in byte offset 0x1, third word in byte offset 0x2 and fourth word in byte offset 0x3. The TCR5[FBT] and/or RCR5[FBT] must be configured within byte offset 0x0.

38.3.4.4 FIFO Combine

FIFO Combining mode allows the separate FIFOs for multiple data channels to be used as a single FIFO for either software accesses or a single data channel or both. Note that the enabled data channels must be contiguous and data channel 0 must be enabled when using FIFO Combine mode.

Combining FIFOs for software access (writing transmit FIFO registers, reading receive FIFO registers) allows a DMA controller or software to read or write multiple FIFOs without incrementing the address that is accessed. Once enabled, the first software access to a FIFO register accesses the first enabled channel FIFO, while the second access to a FIFO register accesses the second enabled channel FIFO. This continues until software accesses the last enabled channel FIFO and the pointer resets back to the first enabled channel FIFO. To reset the pointer manually, software can reset the FIFOs or disable FIFO combining on software accesses.

Combining FIFOs for transmit data channels allows one data channel to use the FIFOs of all enabled channel FIFOs, with identical data output on each enabled data channel. The transmit shift registers for all enabled data channels load at the start of each frame and every N unmasked words (where N is the number of enabled data channels). The first word transmitted loads from the first enabled channel FIFO, while the second word transmitted loads from the second enabled channel FIFO, and so on until the end of the frame. Because the first word in each frame always loads from the first enabled data channel, it is recommended that the number of unmasked words in each frame is evenly divisible by the number of enabled data channels.

Combining FIFOs for receive data channels allows one data channel to use the FIFOs of all enabled channel FIFOs, with received data from channel 0 stored into each enabled data channel. The receive shift register for all enabled data channels are stored after every N unmasked words (where N is the number of enabled data channels). The first word received stores to the first enabled channel FIFO, while the second word received stores to the second enabled channel FIFO, and so on until the end of the frame. Because the first word in each frame always stores to the first enabled data channel, it is recommended that the number of unmasked words in each frame is evenly divisible by the number of enabled data channels.

Note that combining FIFOs for data channels loads or stores each channel FIFO at the same time. This means that FIFO error conditions are only checked every N words (where N is the number of enabled data channels) and that the FIFO warning and request flags assert if any of the enabled data channel meets the warning flag or request flag conditions.

38.3.5 Word mask register

The SAI transmitter and receiver each contain a word mask register (TMR and RMR) that can be used to mask any word in the frame. Because the word mask register is double buffered, software can update it before the end of each frame to mask a particular word in the next frame.

The TMR causes the Transmit Data pin to be tri-stated for the length of each selected word and the transmit FIFO is not read for masked words.

The RMR causes the received data for each selected word to be discarded and not written to the receive FIFO.

38.3.6 Interrupts and DMA requests

The SAI transmitter and receiver generate separate interrupts and separate DMA requests, but support the same status flags. Asynchronous versions of the transmitter and receiver interrupts generate to wake up the CPU from Stop mode. Asynchronous interrupts are only generated when the system clock is gated provided the corresponding BCLK continues.

38.3.6.1 FIFO request flag

The FIFO request flag sets based on the number of entries in the FIFO and the FIFO watermark configuration.

The transmit FIFO request flag sets when the number of entries in any of the enabled transmit FIFOs is less than or equal to the transmit FIFO watermark configuration and clears when the number of entries in each enabled transmit FIFO is greater than the transmit FIFO watermark configuration.

The receive FIFO request flag sets when the number of entries in any of the enabled receive FIFOs is greater than the receive FIFO watermark configuration and clears when the number of entries in each enabled receive FIFO is less than or equal to the receive FIFO watermark configuration.

The FIFO request flag can generate an interrupt or a DMA request.

38.3.6.2 FIFO warning flag

The FIFO warning flag sets based on the number of entries in the FIFO.

The transmit warning flag sets when the number of entries in any of the enabled transmit FIFOs is empty and clears when the number of entries in each enabled transmit FIFO is not empty.

The receive warning flag sets when the number of entries in any of the enabled receive FIFOs is full and clears when the number of entries in each enabled receive FIFO is not full.

The FIFO warning flag can generate an Interrupt or a DMA request.

38.3.6.3 FIFO error flag

The Transmit FIFO Error Flag (TCSR[FEF] = 1]) sets when any of the enabled transmit FIFOs underrun. After the error flag sets, all enabled transmit channels transmit zero data before TCSR[FEF] clears.

When FIFO Continue on Error is enabled (TCR4[FCONT] = 1), the FIFO continues transmitting data following an underrun without software intervention. To ensure that data transmits in the correct order, the transmitter continues from the same word number in the frame that caused the FIFO to underrun, but only after new data writes to the transmit FIFO. Software should still clear the TCSR[FEF] flag, but without reinitializing the transmit FIFOs.

The Receive FIFO Error Flag sets (RCSR[FEF] = 1) when any of the enabled receive FIFOs overflow. After the flag is set, all enabled receive channels discard received data until RCSR[FEF] clears and the next receive frame starts. Empty all enabled receive FIFOs before RCSR[FEF] clears.

When Receive FIFO Continue on Error is enabled (RCR4[FCONT] = 1) the FIFO continues receiving data following an overflow without software intervention. To ensure that data is received in the correct order, the receiver continues from the same word number in the frame that caused the FIFO to overflow, but only after data has been read from the receive FIFO. Software should still clear the RCSR[FEF] flag, but without emptying the receive FIFOs.

The FIFO Error Flag can generate only an interrupt.

38.3.6.4 Sync error flag

The Sync Error Flag, TCSR[SEF] or RCSR[SEF], sets when configured for an externally generated frame sync and the external frame sync asserts when the transmitter or receiver is busy with the previous frame. The external frame sync assertion is ignored and the sync error flag sets. When the Sync Error Flag sets, the transmitter or receiver continues checking for frame sync assertion when idle or at the end of each frame.

The Sync Error Flag can only generate an interrupt.

38.3.6.5 Word start flag

The Word Start Flag sets (TCSR[WSF] = 1) at the start of the second bit clock for the selected word, as configured by the Word Flag Configuration field (TCR3[WDFL]).

The Word Start Flag can generate an interrupt only.

38.3.7 SAI clocking

The SAI clocks include:

- Audio master clock
- Bit clock
- Bus clock

38.3.7.1 Audio master clock

The audio master clock generates the bit clock when the receiver or transmitter is configured for an internally generated bit clock. The transmitter and receiver can independently select between the bus clock and up to three audio master clocks to generate the bit clock.

The audio master clock generation and selection is chip-specific. Refer to chip-specific clocking information about how the audio master clocks are generated.

38.3.7.2 Bit clock

The SAI transmitter and receiver support asynchronous free-running bit clocks that can be generated internally from an audio master clock or supplied externally. There is also the option for synchronous bit clock and frame sync operation between the receiver and transmitter or between multiple SAI peripherals.

- If both transmitter and receiver are configured for asynchronous operation, then the transmitter and receiver will each use *their own* bit clock and frame sync.
- If the *transmitter* is configured for asynchronous mode and the receiver is configured for synchronous mode, then both transmitter and receiver will use the *transmitter* bit clock and frame sync.
- If the *receiver* is configured for asynchronous mode and the transmitter is configured for synchronous mode, then both transmitter and receiver will use the *receiver* bit clock and frame sync.

Note that the software configures synchronous or asynchronous mode, and that choice selects the bit clock/frame sync used.

Externally generated bit clocks must be:

- Enabled before the SAI transmitter or receiver is enabled
- Disabled after the SAI transmitter or receiver is disabled and completes its current frames

If the SAI transmitter or receiver uses an externally generated bit clock in asynchronous mode and that bit clock is generated by an SAI that is disabled in stop mode, then the transmitter or receiver should be disabled by software before entering stop mode. This issue does not apply when the transmitter or receiver is in a synchronous mode because all synchronous SAIs are enabled and disabled simultaneously.

38.3.7.3 Bus clock

The bus clock is used by the control and configuration registers and to generate synchronous interrupts and DMA requests.

NOTE

Although there is no minimum bus clock frequency specified, the bus clock frequency must be fast enough (relative to the bit clock frequency) to ensure that the FIFOs can be serviced, without generating either a transmitter FIFO underrun or receiver FIFO overflow condition.

38.3.8 SAI resets

The SAI is asynchronously reset on system reset. The SAI has a software reset and a FIFO reset.

38.3.8.1 Software reset

The SAI transmitter includes a software reset that resets all transmitter internal logic, including the bit clock generation, status flags, and FIFO pointers. This software reset does not reset the configuration registers. The software reset remains asserted until cleared by software.

The SAI receiver includes a software reset that resets all receiver internal logic, including the bit clock generation, status flags and FIFO pointers. This software reset does not reset the configuration registers. The software reset remains asserted until cleared by software.

38.3.8.2 FIFO reset

The SAI transmitter includes a FIFO reset that synchronizes the FIFO write pointer to the same value as the FIFO read pointer. This FIFO reset empties the FIFO contents and is used after TCSR[FEF] is set, and before the FIFO is re-initialized and TCSR[FEF] is cleared. The FIFO reset is asserted for one cycle only.

The SAI transmitter can also reset the FIFO of individual data channels by setting the appropriate TCR3[CFR] bit. This should only be done when the corresponding TCR3[TCE] bit is clear.

The SAI receiver includes a FIFO reset that synchronizes the FIFO read pointer to the same value as the FIFO write pointer. This empties the FIFO contents and is to be used after the RCSR[FEF] is set and any remaining data has been read from the FIFO, and before the RCSR[FEF] is cleared. The FIFO reset is asserted for one cycle only.

The SAI receiver can also reset the FIFO of individual data channels by setting the appropriate RCR3[CFR] bit. This should only be done when the corresponding RCR3[RCE] bit is clear.

38.4 External signals

Name	Function	I/O
TX_BCLK	Transmit Bit Clock. The bit clock is an input when externally generated and an output when internally generated.	I/O
TX_SYNC	Transmit Frame Sync. The frame sync is an input sampled synchronously by the bit clock when externally generated and an output generated synchronously by the bit clock when internally generated.	I/O
TX_DATA[3:0]	Transmit Data. The transmit data is generated synchronously by the bit clock and is tristated whenever not transmitting a word.	O
RX_BCLK	Receive Bit Clock. The bit clock is an input when externally generated and an output when internally generated.	I/O
RX_SYNC	Receive Frame Sync. The frame sync is an input sampled synchronously by the bit clock when externally generated and an output generated synchronously by the bit clock when internally generated.	I/O
RX_DATA[3:0]	Receive Data. The receive data is sampled synchronously by the bit clock.	I

38.5 Memory map and register definition

A read or write access to an address from offset 0x100 and above will result in a bus error.

38.5.1 SAI register descriptions

38.5.1.1 SAI memory map

SAI1 base address: 4038_4000h

SAI2 base address: 4038_8000h

SAI3 base address: 4038_C000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Version ID (VERID)	32	RO	0300_0000h
4h	Parameter (PARAM)	32	RO	Table 38-2
8h	Transmit Control (TCSR)	32	RW	0000_0000h
Ch	Transmit Configuration 1 (TCR1)	32	RW	0000_0000h
10h	Transmit Configuration 2 (TCR2)	32	RW	0000_0000h
14h	Transmit Configuration 3 (TCR3)	32	RW	0000_0000h
18h	Transmit Configuration 4 (TCR4)	32	RW	0000_0000h
1Ch	Transmit Configuration 5 (TCR5)	32	RW	0000_0000h
20h - 2Ch	Transmit Data (TDR0 - TDR3)	32	WORZ	See description
40h - 4Ch	Transmit FIFO (TFR0 - TFR3)	32	RO	See description
60h	Transmit Mask (TMR)	32	RW	0000_0000h
88h	Receive Control (RCSR)	32	RW	0000_0000h
8Ch	Receive Configuration 1 (RCR1)	32	RW	0000_0000h
90h	Receive Configuration 2 (RCR2)	32	RW	0000_0000h
94h	Receive Configuration 3 (RCR3)	32	RW	0000_0000h
98h	Receive Configuration 4 (RCR4)	32	RW	0000_0000h
9Ch	Receive Configuration 5 (RCR5)	32	RW	0000_0000h
A0h - ACh	Receive Data (RDR0 - RDR3)	32	RO	See description
C0h - CCh	Receive FIFO (RFR0 - RFR3)	32	RO	See description
E0h	Receive Mask (RMR)	32	RW	0000_0000h

38.5.1.2 Version ID (VERID)

38.5.1.2.1 Offset

Register	Offset
VERID	0h

38.5.1.2.2 Function

Contains version numbers for the module design and feature set.

38.5.1.2.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	MAJOR								MINOR							
W																
Reset	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FEATURE															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

38.5.1.2.4 Fields

Field	Function
31-24 MAJOR	Major Version Number This read only field returns the major version number for the specification.
23-16 MINOR	Minor Version Number This read only field returns the minor version number for the specification.
15-0 FEATURE	Feature Specification Number This read only field returns the feature set number. 0000000000000000b - Standard feature set.

38.5.1.3 Parameter (PARAM)

38.5.1.3.1 Offset

Register	Offset
PARAM	4h

38.5.1.3.2 Function

Contains parameter values that were implemented in the module.

38.5.1.3.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0												FRAME			
W																
Reset	See Register reset values.															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0				FIFO				0				DATALINE			
W																
Reset	See Register reset values.															

38.5.1.3.4 Register reset values

Register	Reset value
PARAM	SAI1: 0005_0504h SAI2,SAI3: 0005_0501h

38.5.1.3.5 Fields

Field	Function
31-20 —	Reserved
19-16 FRAME	Frame Size The maximum number of slots per frame is 2^FRAME.
15-12 —	Reserved
11-8 FIFO	FIFO Size The number of words in each FIFO is 2^FIFO.
7-4 —	Reserved
3-0 DATALINE	Number of Datalines The number of datalines implemented.

38.5.1.4 Transmit Control (TCSR)

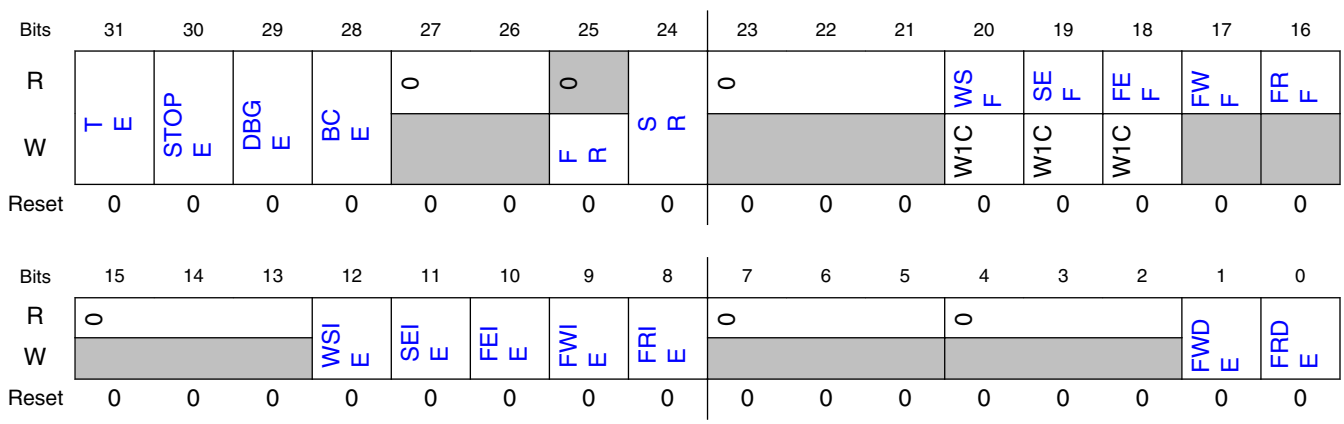
38.5.1.4.1 Offset

Register	Offset
TCSR	8h

38.5.1.4.2 Function

Contains transmitter enable fields including resets, error and interrupt enable fields, and error flag fields.

38.5.1.4.3 Diagram



38.5.1.4.4 Fields

Field	Function
31 TE	Transmitter Enable Enables/disables the transmitter. When software clears this field, the transmitter remains enabled, and this bit remains set, until the end of the current frame. 0b - Transmitter is disabled. 1b - Transmitter is enabled, or transmitter has been disabled and has not yet reached end of frame.
30 STOPE	Stop Enable Configures transmitter operation in Stop mode. 0b - Transmitter disabled in Stop mode. 1b - Transmitter enabled in Stop mode.
29	Debug Enable

Table continues on the next page...

Field	Function
DBGE	Enables/disables transmitter operation in Debug mode. The transmit bit clock is not affected by debug mode. 0b - Transmitter is disabled in Debug mode, after completing the current frame. 1b - Transmitter is enabled in Debug mode.
28 BCE	Bit Clock Enable Enables the transmit bit clock, separately from the TE. This field is automatically set whenever TE is set. When software clears this field, the transmit bit clock remains enabled, and this bit remains set, until the end of the current frame. 0b - Transmit bit clock is disabled. 1b - Transmit bit clock is enabled.
27-26 —	Reserved
25 FR	FIFO Reset Empties the FIFO, and sets the FIFO read and write pointers to the same value, which may or may not be zero. Reading this field will always return zero. FIFO pointers should only be reset when the transmitter is disabled or the FIFO error flag is set. 0b - No effect. 1b - FIFO reset.
24 SR	Software Reset When set, resets the internal transmitter logic including the FIFO read and write pointers. Software-visible registers are not affected, except for the status registers. 0b - No effect. 1b - Software reset.
23-21 —	Reserved
20 WSF	Word Start Flag Indicates that the start of the configured word has been detected. Write a logic 1 to this field to clear this flag. 0b - Start of word not detected. 1b - Start of word detected.
19 SEF	Sync Error Flag Indicates that an error in the externally-generated frame sync has been detected. Write a logic 1 to this field to clear this flag. 0b - Sync error not detected. 1b - Frame sync error detected.
18 FEF	FIFO Error Flag Indicates that an enabled transmit FIFO has underrun. Write a logic 1 to this field to clear this flag. 0b - Transmit underrun not detected. 1b - Transmit underrun detected.
17 FWF	FIFO Warning Flag Indicates that an enabled transmit FIFO is empty. 0b - No enabled transmit FIFO is empty. 1b - Enabled transmit FIFO is empty.
16 FRF	FIFO Request Flag Indicates that the number of words in an enabled transmit channel FIFO is less than or equal to the transmit FIFO watermark. 0b - Transmit FIFO watermark has not been reached. 1b - Transmit FIFO watermark has been reached.

Table continues on the next page...

Field	Function
15-13 —	Reserved
12 WSIE	Word Start Interrupt Enable Enables/disables word start interrupts. 0b - Disables interrupt. 1b - Enables interrupt.
11 SEIE	Sync Error Interrupt Enable Enables/disables sync error interrupts. 0b - Disables interrupt. 1b - Enables interrupt.
10 FEIE	FIFO Error Interrupt Enable Enables/disables FIFO error interrupts. 0b - Disables the interrupt. 1b - Enables the interrupt.
9 FWIE	FIFO Warning Interrupt Enable Enables/disables FIFO warning interrupts. 0b - Disables the interrupt. 1b - Enables the interrupt.
8 FRIE	FIFO Request Interrupt Enable Enables/disables FIFO request interrupts. 0b - Disables the interrupt. 1b - Enables the interrupt.
7-5 —	Reserved
4-2 —	Reserved
1 FWDE	FIFO Warning DMA Enable Enables/disables DMA requests. 0b - Disables the DMA request. 1b - Enables the DMA request.
0 FRDE	FIFO Request DMA Enable Enables/disables DMA requests. 0b - Disables the DMA request. 1b - Enables the DMA request.

38.5.1.5 Transmit Configuration 1 (TCR1)

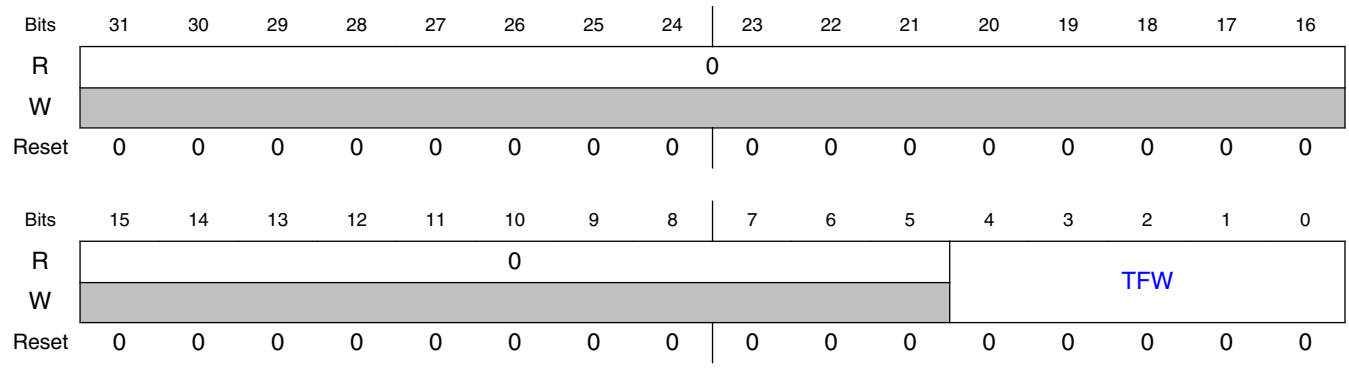
38.5.1.5.1 Offset

Register	Offset
TCR1	Ch

38.5.1.5.2 Function

Configures the watermark level for all enabled transmit channels.

38.5.1.5.3 Diagram



38.5.1.5.4 Fields

Field	Function
31-5 —	Reserved
4-0 TFW	Transmit FIFO Watermark

38.5.1.6 Transmit Configuration 2 (TCR2)

38.5.1.6.1 Offset

Register	Offset
TCR2	10h

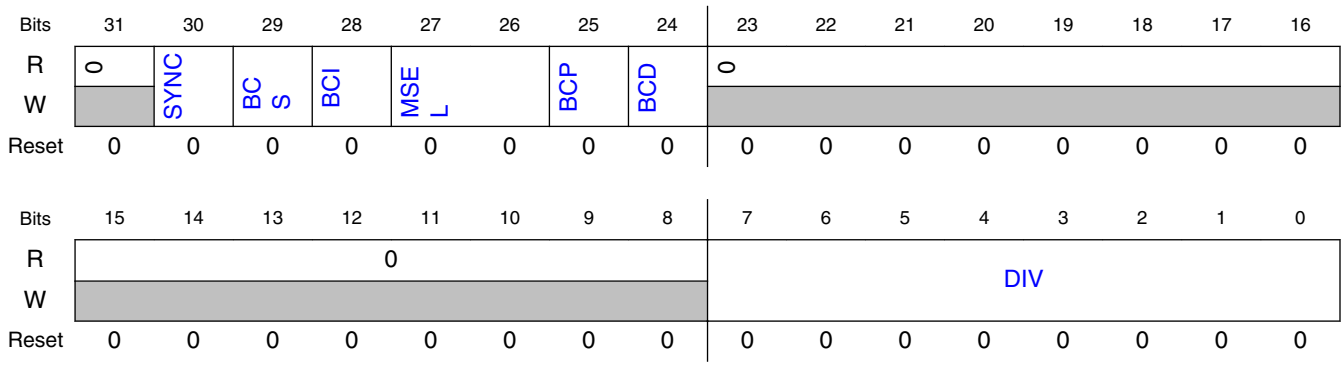
38.5.1.6.2 Function

Contains the SYNC mode and clock setting fields.

NOTE

This register must not be altered when TCSR[TE] is set.

38.5.1.6.3 Diagram



38.5.1.6.4 Fields

Field	Function
31 —	Reserved
30 SYNC	Synchronous Mode Configures between asynchronous and synchronous modes of operation. When configured for a synchronous mode of operation, the receiver must be configured for asynchronous operation. 0b - Asynchronous mode. 1b - Synchronous with receiver.
29 BCS	Bit Clock Swap This field swaps the bit clock used by the transmitter. When the transmitter is configured in asynchronous mode and this bit is set, the transmitter is clocked by the receiver bit clock (RX_BCLK). This allows the transmitter and receiver to share the same bit clock, but the transmitter continues to use the transmit frame sync (TX_SYNC). When the transmitter is configured in synchronous mode, the transmitter BCS field and receiver BCS field must be set to the same value. When both are set, the transmitter and receiver are both clocked by the transmitter bit clock (TX_BCLK) but use the receiver frame sync (RX_SYNC). 0b - Use the normal bit clock source. 1b - Swap the bit clock source.
28 BCI	Bit Clock Input When this field is set and using an internally generated bit clock in either synchronous or asynchronous mode, the bit clock actually used by the transmitter is delayed by the pad output delay (the transmitter is clocked by the pad input as if the clock was externally generated). This has the effect of decreasing the data input setup time, but increasing the data output valid time. The slave mode timing from the datasheet should be used for the transmitter when this bit is set. In synchronous mode, this bit allows the transmitter to use the slave mode timing from the datasheet, while the receiver uses the master mode timing. This field has no effect when configured for an externally generated bit clock . 0b - No effect. 1b - Internal logic is clocked as if bit clock was externally generated.
27-26	MCLK Select

Table continues on the next page...

Field	Function
MSEL	Selects the audio Master Clock option used to generate an internally generated bit clock. This field has no effect when configured for an externally generated bit clock. **NOTE:** Depending on the device, some Master Clock options might not be available. See the chip-specific information for the meaning of each option. 00b - Bus Clock selected. 01b - Master Clock (MCLK) 1 option selected. 10b - Master Clock (MCLK) 2 option selected. 11b - Master Clock (MCLK) 3 option selected.
25 BCP	Bit Clock Polarity Configures the polarity of the bit clock. 0b - Bit clock is active high with drive outputs on rising edge and sample inputs on falling edge. 1b - Bit clock is active low with drive outputs on falling edge and sample inputs on rising edge.
24 BCD	Bit Clock Direction Configures the direction of the bit clock. 0b - Bit clock is generated externally in Slave mode. 1b - Bit clock is generated internally in Master mode.
23-8 —	Reserved
7-0 DIV	Bit Clock Divide Divides down the audio master clock to generate the bit clock when configured for an internal bit clock. The division value is (DIV + 1) * 2.

38.5.1.7 Transmit Configuration 3 (TCR3)

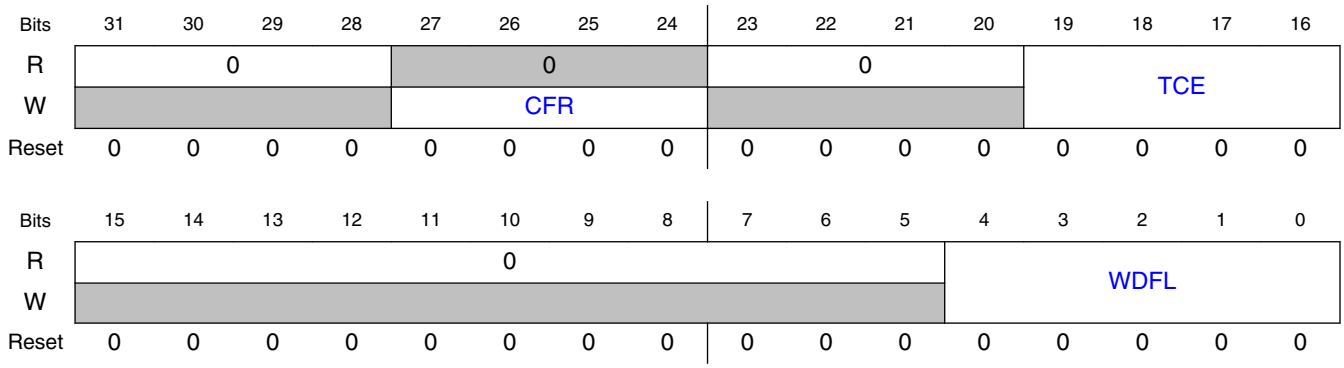
38.5.1.7.1 Offset

Register	Offset
TCR3	14h

38.5.1.7.2 Function

Contains the transmit channel settings.

38.5.1.7.3 Diagram



38.5.1.7.4 Fields

Field	Function
31-28 —	Reserved
27-24 CFR	Channel FIFO Reset Resets the FIFO pointers for a specific channel. Reading this field always returns zero. FIFO pointers should only be reset when a channel is disabled or the FIFO error flag is set. The width of CFR field = the number of transmit channels (call it N). For example, if CFR is 2 bits wide, then bit position 24 refers to transmit channel 1 FIFO pointer and bit position 25 refers to transmit channel 2 FIFO pointer. Setting bit 24 resets transmit channel 1 FIFO pointer, and setting bit 25 enables transmit channel 2 FIFO pointer. Setting bit N will reset transmit channel N FIFO pointer. **NOTE:** When there is only a single channel, there is no need for individual channel FIFO reset (TCR3[CFR]). In the case of a single channel, use the global FIFO reset (TCSR[FR]). 0b - No effect. 1b - Transmit data channel N FIFO is reset. **NOTE:** This field is not supported in every instance. The following table includes only supported registers.
23-20 —	Reserved
19-16 TCE	Transmit Channel Enable Enables the corresponding data channel for transmit operation. Changing TCE field will take effect immediately for generating the FIFO request and warning flags, but at the end of each frame for transmit operation.

Instance	Field supported in	Field not supported in
SAI1	TCR3	—
SAI2	—	TCR3
SAI3	—	TCR3

Table continues on the next page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>The width of TCE field = the number of transmit channels (call it N). For example, if TCE field is 2 bits wide, then bit position 16 refers to transmit channel 1 and bit position 17 refers to transmit channel 2. Setting bit 16 enables transmit channel 1, and setting bit 17 enables transmit channel 2. Setting bit N will enable transmit channel N.

0b - Transmit data channel N is disabled.
1b - Transmit data channel N is enabled.

NOTE: This field is not supported in every instance. The following table includes only supported registers.

<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>SAI1</td><td>TCR3</td><td>—</td></tr>
<tr><td>SAI2</td><td>TCR3[16]</td><td>TCR3[19–17]</td></tr>
<tr><td>SAI3</td><td>TCR3[16]</td><td>TCR3[19–17]</td></tr>
</table>
</td></tr>
<tr><td>15-5
—</td><td>Reserved</td></tr>
<tr><td>4-0
WDFL</td><td>Word Flag Configuration

Configures which word sets the start of word flag. The value written must be one less than the word number. For example, writing 0 configures the first word in the frame. When configured to a value greater than TCR4[FRSZ], then the start of word flag is never set.</td></tr>
</table>

38.5.1.8 Transmit Configuration 4 (TCR4)

38.5.1.8.1 Offset

Register	Offset
TCR4	18h

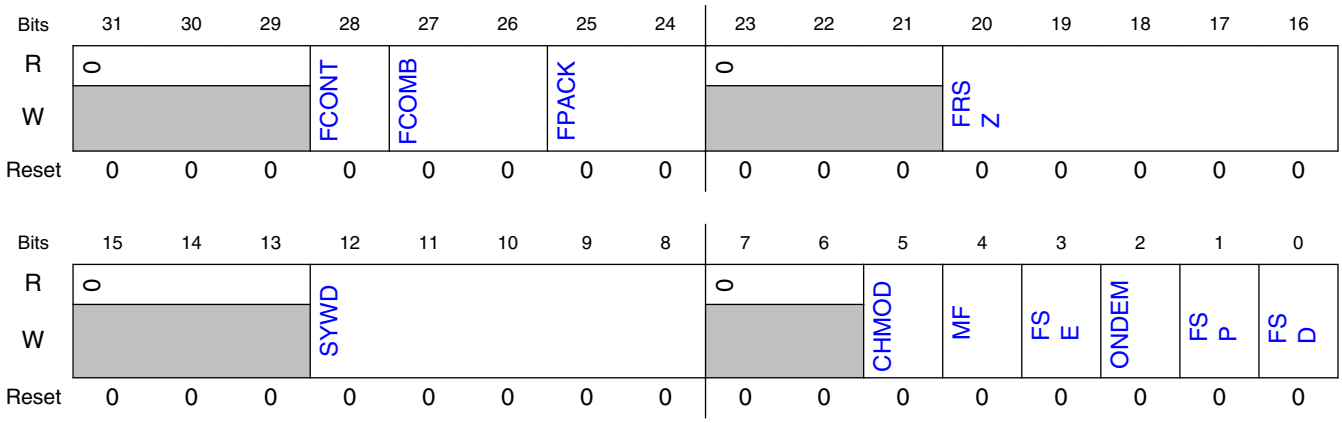
38.5.1.8.2 Function

Contains the transmit fields for FIFO Combine Mode, FIFO Packing Mode, and frame sync settings.

NOTE

This register must not be altered when TCSR[TE] is set.

38.5.1.8.3 Diagram



38.5.1.8.4 Fields

Field	Function
31-29 —	Reserved
28 FCONT	FIFO Continue on Error Configures when the SAI will continue transmitting after a FIFO error has been detected. 0b - On FIFO error, the SAI will continue from the start of the next frame after the FIFO error flag has been cleared. 1b - On FIFO error, the SAI will continue from the same word that caused the FIFO error to set after the FIFO warning flag has been cleared.
27-26 FCOMB	FIFO Combine Mode When FIFO combine mode is enabled for FIFO writes, software writing to any FIFO data register will alternate the write among the enabled data channel FIFOs. For example, if two data channels are enabled then the first write will be performed to the first enabled data channel FIFO and the second write will be performed to the second enabled data channel FIFO. Reseting the FIFO or disabling FIFO combine mode for FIFO writes will reset the pointer back to the first enabled data channel. When FIFO combine mode is enabled for FIFO reads from the transmit shift registers, the transmit data channel output will alternate between the enabled data channel FIFOs. For example, if two data channels are enabled then the first unmasked word will be transmitted from the first enabled data channel FIFO and the second unmasked word will be transmitted from the second enabled data channel FIFO. Since the first word of the frame is always transmitted from the first enabled data channel FIFO, it is recommended that the number of unmasked words per frame is evenly divisible by the number of enabled data channels. **NOTE:** This field is not supported in every instance. The following table includes only supported registers. **Instance** / **Field supported in** / **Field not supported in** SAI1 / TCR4 / — SAI2 / — / TCR4 SAI3 / — / TCR4

Table continues on the next page...

Field	Function
	00b - FIFO combine mode disabled. 01b - FIFO combine mode enabled on FIFO reads (from transmit shift registers). 10b - FIFO combine mode enabled on FIFO writes (by software). 11b - FIFO combine mode enabled on FIFO reads (from transmit shift registers) and writes (by software).
25-24 FPACK	FIFO Packing Mode Enables packing of 8-bit data or 16-bit data into each 32-bit FIFO word. If the word size is greater than 8-bit or 16-bit then only the first 8-bit or 16-bits are loaded from the FIFO. The first word in each frame always starts with a new 32-bit FIFO word and the first bit shifted must be configured within the first packed word. When FIFO packing is enabled, the FIFO write pointer will only increment when the full 32-bit FIFO word has been written by software. 00b - FIFO packing is disabled. 01b - Reserved 10b - 8-bit FIFO packing is enabled. 11b - 16-bit FIFO packing is enabled.
23-21 —	Reserved
20-16 FRSZ	Frame size Configures the number of words in each frame. The value written must be one less than the number of words in the frame. For example, write 0 for one word per frame. The maximum supported frame size is 32 words.
15-13 —	Reserved
12-8 SYWD	Sync Width Configures the length of the frame sync in number of bit clocks. The value written must be one less than the number of bit clocks. For example, write 0 for the frame sync to assert for one bit clock only. The sync width cannot be configured longer than the first word of the frame.
7-6 —	Reserved
5 CHMOD	Channel Mode Configures if transmit data pins are configured for TDM mode or Output mode. 0b - TDM mode, transmit data pins are tri-stated when slots are masked or channels are disabled. 1b - Output mode, transmit data pins are never tri-stated and will output zero when slots are masked or channels are disabled.
4 MF	MSB First Configures whether the LSB or the MSB is transmitted first. 0b - LSB is transmitted first. 1b - MSB is transmitted first.
3 FSE	Frame Sync Early 0b - Frame sync asserts with the first bit of the frame. 1b - Frame sync asserts one bit before the first bit of the frame.
2 ONDEM	On Demand Mode When set, and the frame sync is generated internally, a frame sync is only generated when the FIFO warning flag is clear. 0b - Internal frame sync is generated continuously. 1b - Internal frame sync is generated when the FIFO warning flag is clear.
1	Frame Sync Polarity

Table continues on the next page...

Field	Function
FSP	Configures the polarity of the frame sync. 0b - Frame sync is active high. 1b - Frame sync is active low.
0 FSD	Frame Sync Direction Configures the direction of the frame sync. 0b - Frame sync is generated externally in Slave mode. 1b - Frame sync is generated internally in Master mode.

38.5.1.9 Transmit Configuration 5 (TCR5)

38.5.1.9.1 Offset

Register	Offset
TCR5	1Ch

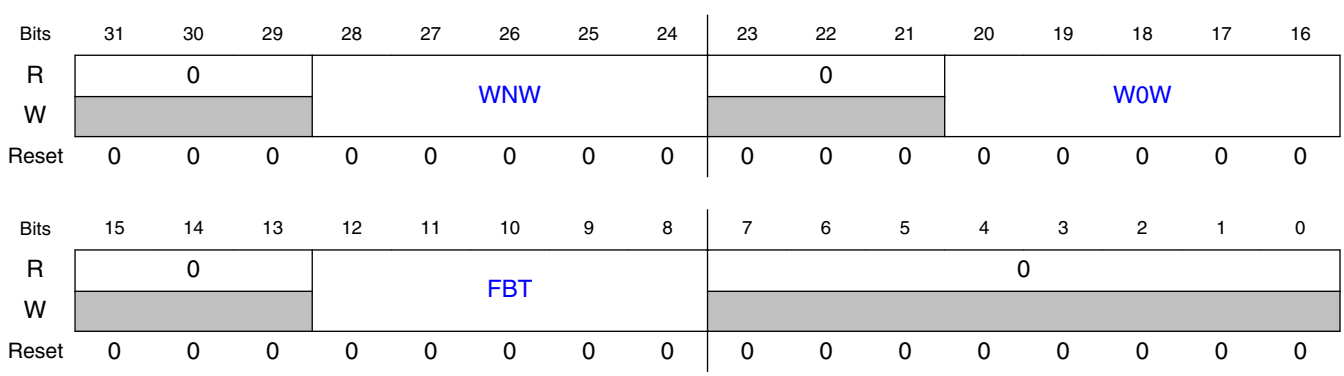
38.5.1.9.2 Function

Contains transmit word width and bit index settings.

NOTE

This register must not be altered when TCSR[TE] is set.

38.5.1.9.3 Diagram



38.5.1.9.4 Fields

Field	Function
31-29 —	Reserved
28-24 WNW	Word N Width Configures the number of bits in each word, for each word except the first in the frame. The value written must be one less than the number of bits per word. Word width of less than 8 bits is not supported.
23-21 —	Reserved
20-16 W0W	Word 0 Width Configures the number of bits in the first word in each frame. The value written must be one less than the number of bits in the first word. Word width of less than 8 bits is not supported if there is only one word per frame.
15-13 —	Reserved
12-8 FBT	First Bit Shifted Configures the bit index for the first bit transmitted for each word in the frame. If configured for MSB First, the index of the next bit transmitted is one less than the current bit transmitted. If configured for LSB First, the index of the next bit transmitted is one more than the current bit transmitted. The value written must be greater than or equal to the word width when configured for MSB First. The value written must be less than or equal to 31-word width when configured for LSB First.
7-0 —	Reserved

38.5.1.10 Transmit Data (TDR0 - TDR3)

38.5.1.10.1 Offset

Register	Offset
TDR0	20h
TDR1	24h
TDR2	28h
TDR3	2Ch

38.5.1.10.2 Function

Writes to this register when the transmit FIFO is not full will push the data written into the transmit data FIFO. Writes to this register when the transmit FIFO is full are ignored.

NOTE

Each module instance supports a different number of registers.

Instance	Register supported	Register not supported
SAI1	TDR0–TDR3	—
SAI2	TDR0	TDR1–TDR3
SAI3	TDR0	TDR1–TDR3

38.5.1.10.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W	TDR															
Reset	See Register reset values.															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0															
W	TDR															
Reset	See Register reset values.															

38.5.1.10.4 Register reset values

Register	Reset value
TDR0	SAI1–SAI3: 0000_0000h
TDR1–TDR3	0000_0000h

38.5.1.10.5 Fields

Field	Function
31-0 TDR	Transmit Data Register

38.5.1.11 Transmit FIFO (TFR0 - TFR3)

38.5.1.11.1 Offset

Register	Offset
TFR0	40h
TFR1	44h
TFR2	48h
TFR3	4Ch

38.5.1.11.2 Function

The MSB of the read and write pointers is used to distinguish between FIFO full and empty conditions. If the read and write pointers are identical, then the FIFO is empty. If the read and write pointers are identical except for the MSB, then the FIFO is full.

NOTE

Each module instance supports a different number of registers.

Instance	Register supported	Register not supported
SAI1	TFR0–TFR3	—
SAI2	TFR0	TFR1–TFR3
SAI3	TFR0	TFR1–TFR3

38.5.1.11.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	WCP	0									WFP					
W																
Reset	See Register reset values.															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0										RFP					
W																
Reset	See Register reset values.															

38.5.1.11.4 Register reset values

Register	Reset value
TFR0	SAI1–SAI3: 0000_0000h
TFR1–TFR3	0000_0000h

38.5.1.11.5 Fields

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td>31
WCP</td><td>Write Channel Pointer
When FIFO Combine mode is enabled for writes, indicates that this data channel is the next FIFO to be written.
NOTE: This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>SAI1</td><td>TFR0–TFR3</td><td>—</td></tr>
<tr><td>SAI2</td><td>—</td><td>TFR0</td></tr>
<tr><td>SAI3</td><td>—</td><td>TFR0</td></tr>
</table>
0b - No effect.
1b - FIFO combine is enabled for FIFO writes and this FIFO will be written on the next FIFO write.</td></tr>
<tr><td>30-22
—</td><td>Reserved</td></tr>
<tr><td>21-16
WFP</td><td>Write FIFO Pointer
FIFO write pointer for transmit data channel.</td></tr>
<tr><td>15-6
—</td><td>Reserved</td></tr>
<tr><td>5-0
RFP</td><td>Read FIFO Pointer
FIFO read pointer for transmit data channel.</td></tr>
</table>

38.5.1.12 Transmit Mask (TMR)

38.5.1.12.1 Offset

Register	Offset
TMR	60h

38.5.1.12.2 Function

This register is double-buffered and updates:

1. When TCSR[TE] is first set
2. At the end of each frame

This allows the masked words in each frame to change from frame to frame.

38.5.1.12.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	TWM															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TWM															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

38.5.1.12.4 Fields

Field	Function
31-0 TWM	Transmit Word Mask Configures whether the transmit word is masked (transmit data pins are tri-stated or drive zero and transmit data not read from FIFO) for the corresponding word in the frame. 00000000000000000000000000000000b - Word N is enabled. 00000000000000000000000000000001b - Word N is masked. The transmit data pins are tri-stated or drive zero when masked.

38.5.1.13 Receive Control (RCSR)

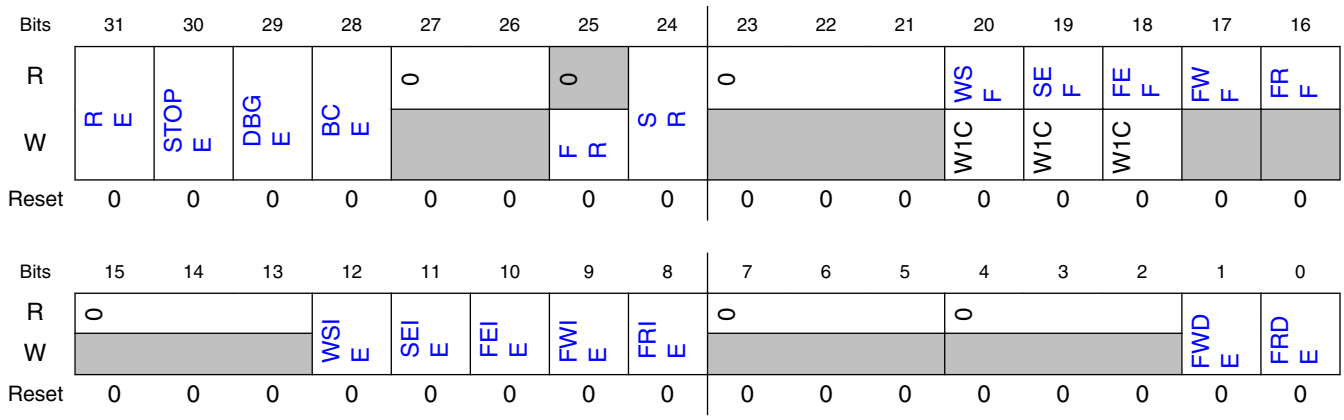
38.5.1.13.1 Offset

Register	Offset
RCSR	88h

38.5.1.13.2 Function

Contains receiver enable fields including resets, error and interrupt enable fields, and error flag fields.

38.5.1.13.3 Diagram



38.5.1.13.4 Fields

Field	Function
31 RE	Receiver Enable Enables/disables the receiver. When software clears this field, the receiver remains enabled, and this bit remains set, until the end of the current frame. 0b - Receiver is disabled. 1b - Receiver is enabled, or receiver has been disabled and has not yet reached end of frame.
30 STOPE	Stop Enable Configures receiver operation in Stop mode. 0b - Receiver disabled in Stop mode. 1b - Receiver enabled in Stop mode.
29 DBGE	Debug Enable Enables/disables receiver operation in Debug mode. The receive bit clock is not affected by Debug mode. 0b - Receiver is disabled in Debug mode, after completing the current frame. 1b - Receiver is enabled in Debug mode.
28 BCE	Bit Clock Enable Enables the receive bit clock, separately from RE. This field is automatically set whenever RE is set. When software clears this field, the receive bit clock remains enabled, and this field remains set, until the end of the current frame. 0b - Receive bit clock is disabled. 1b - Receive bit clock is enabled.
27-26 —	Reserved
25	FIFO Reset

Table continues on the next page...

Field	Function
FR	Empties the FIFO, and sets the FIFO read and write pointers to the same value, which may or may not be zero. Reading this field will always return zero. FIFO pointers should only be reset when the receiver is disabled or the FIFO error flag is set. 0b - No effect. 1b - FIFO reset.
24 SR	Software Reset Resets the internal receiver logic including the FIFO pointers. Software-visible registers are not affected, except for the status registers. 0b - No effect. 1b - Software reset.
23-21 —	Reserved
20 WSF	Word Start Flag Indicates that the start of the configured word has been detected. Write a logic 1 to this field to clear this flag. 0b - Start of word not detected. 1b - Start of word detected.
19 SEF	Sync Error Flag Indicates that an error in the externally-generated frame sync has been detected. Write a logic 1 to this field to clear this flag. 0b - Sync error not detected. 1b - Frame sync error detected.
18 FEF	FIFO Error Flag Indicates that an enabled receive FIFO has overflowed. Write a logic 1 to this field to clear this flag. 0b - Receive overflow not detected. 1b - Receive overflow detected.
17 FWF	FIFO Warning Flag Indicates that an enabled receive FIFO is full. 0b - No enabled receive FIFO is full. 1b - Enabled receive FIFO is full.
16 FRF	FIFO Request Flag Indicates that the number of words in an enabled receive channel FIFO is greater than the receive FIFO watermark. 0b - Receive FIFO watermark not reached. 1b - Receive FIFO watermark has been reached.
15-13 —	Reserved
12 WSIE	Word Start Interrupt Enable Enables/disables word start interrupts. 0b - Disables interrupt. 1b - Enables interrupt.
11 SEIE	Sync Error Interrupt Enable Enables/disables sync error interrupts. 0b - Disables interrupt. 1b - Enables interrupt.
10 FEIE	FIFO Error Interrupt Enable Enables/disables FIFO error interrupts.

Table continues on the next page...

Field	Function
	0b - Disables the interrupt. 1b - Enables the interrupt.
9 FWIE	FIFO Warning Interrupt Enable Enables/disables FIFO warning interrupts. 0b - Disables the interrupt. 1b - Enables the interrupt.
8 FRIE	FIFO Request Interrupt Enable Enables/disables FIFO request interrupts. 0b - Disables the interrupt. 1b - Enables the interrupt.
7-5 —	Reserved
4-2 —	Reserved
1 FWDE	FIFO Warning DMA Enable Enables/disables DMA requests. 0b - Disables the DMA request. 1b - Enables the DMA request.
0 FRDE	FIFO Request DMA Enable Enables/disables DMA requests. 0b - Disables the DMA request. 1b - Enables the DMA request.

38.5.1.14 Receive Configuration 1 (RCR1)

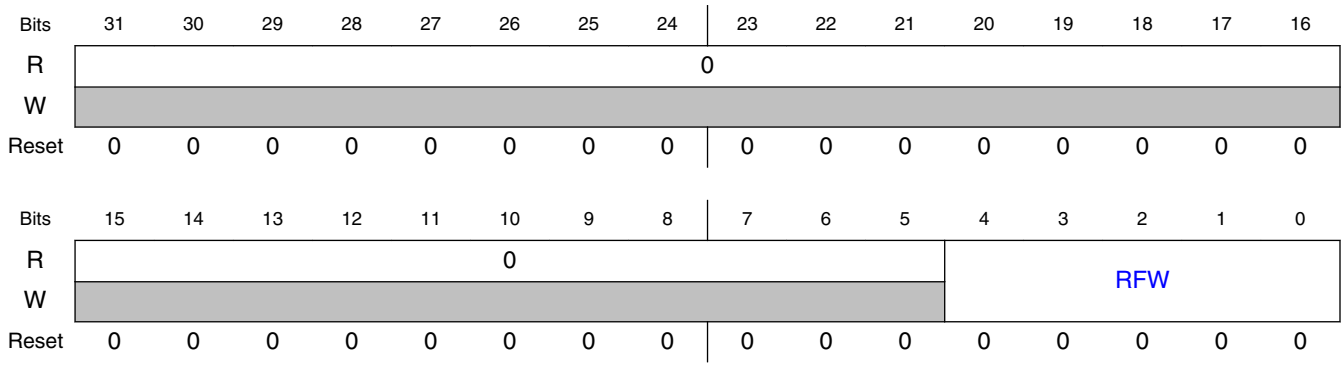
38.5.1.14.1 Offset

Register	Offset
RCR1	8Ch

38.5.1.14.2 Function

Configures the watermark level for all enabled receiver channels.

38.5.1.14.3 Diagram



38.5.1.14.4 Fields

Field	Function
31-5 —	Reserved
4-0 RFW	Receive FIFO Watermark

38.5.1.15 Receive Configuration 2 (RCR2)

38.5.1.15.1 Offset

Register	Offset
RCR2	90h

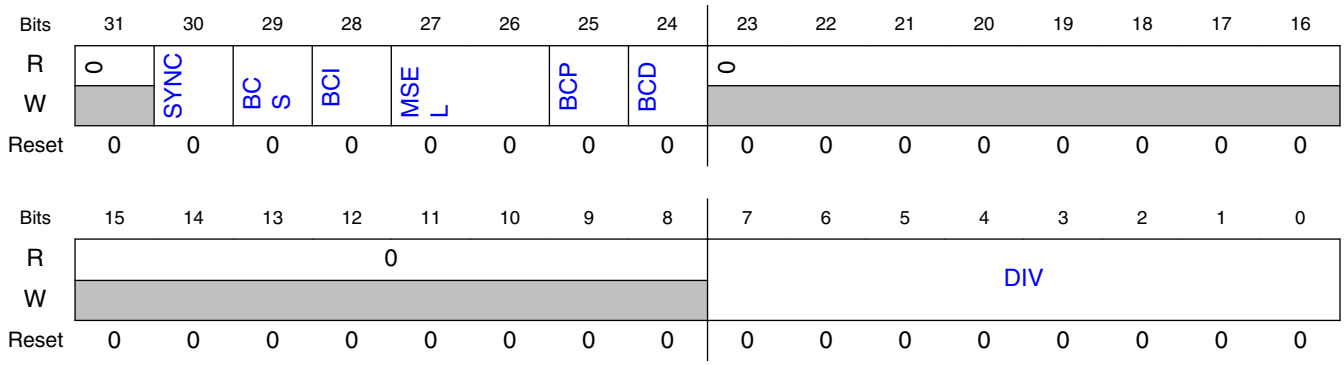
38.5.1.15.2 Function

Contains the SYNC mode and clock setting fields.

NOTE

This register must not be altered when RCSR[RE] is set.

38.5.1.15.3 Diagram



38.5.1.15.4 Fields

Field	Function
31 —	Reserved
30 SYNC	Synchronous Mode Configures between asynchronous and synchronous modes of operation. When configured for a synchronous mode of operation, the transmitter must be configured for asynchronous operation. 0b - Asynchronous mode. 1b - Synchronous with transmitter.
29 BCS	Bit Clock Swap This field swaps the bit clock used by the receiver. When the receiver is configured in asynchronous mode and this bit is set, the receiver is clocked by the transmitter bit clock (TX_BCLK). This allows the transmitter and receiver to share the same bit clock, but the receiver continues to use the receiver frame sync (RX_SYNC). When the receiver is configured in synchronous mode, the transmitter BCS field and receiver BCS field must be set to the same value. When both are set, the transmitter and receiver are both clocked by the receiver bit clock (RX_BCLK) but use the transmitter frame sync (TX_SYNC). 0b - Use the normal bit clock source. 1b - Swap the bit clock source.
28 BCI	Bit Clock Input When this field is set and using an internally generated bit clock in either synchronous or asynchronous mode, the bit clock actually used by the receiver is delayed by the pad output delay (the receiver is clocked by the pad input as if the clock was externally generated). This has the effect of decreasing the data input setup time, but increasing the data output valid time. The slave mode timing from the datasheet should be used for the receiver when this bit is set. In synchronous mode, this bit allows the receiver to use the slave mode timing from the datasheet, while the transmitter uses the master mode timing. This field has no effect when configured for an externally generated bit clock . 0b - No effect. 1b - Internal logic is clocked as if bit clock was externally generated.
27-26 MSEL	MCLK Select

Table continues on the next page...

Field	Function
	Selects the audio Master Clock option used to generate an internally generated bit clock. This field has no effect when configured for an externally generated bit clock. **NOTE:** Depending on the device, some Master Clock options might not be available. See the chip-specific information for the availability and chip-specific meaning of each option. 00b - Bus Clock selected. 01b - Master Clock (MCLK) 1 option selected. 10b - Master Clock (MCLK) 2 option selected. 11b - Master Clock (MCLK) 3 option selected.
25 BCP	Bit Clock Polarity Configures the polarity of the bit clock. 0b - Bit Clock is active high with drive outputs on rising edge and sample inputs on falling edge. 1b - Bit Clock is active low with drive outputs on falling edge and sample inputs on rising edge.
24 BCD	Bit Clock Direction Configures the direction of the bit clock. 0b - Bit clock is generated externally in Slave mode. 1b - Bit clock is generated internally in Master mode.
23-8 —	Reserved
7-0 DIV	Bit Clock Divide Divides down the audio master clock to generate the bit clock when configured for an internal bit clock. The division value is (DIV + 1) * 2.

38.5.1.16 Receive Configuration 3 (RCR3)

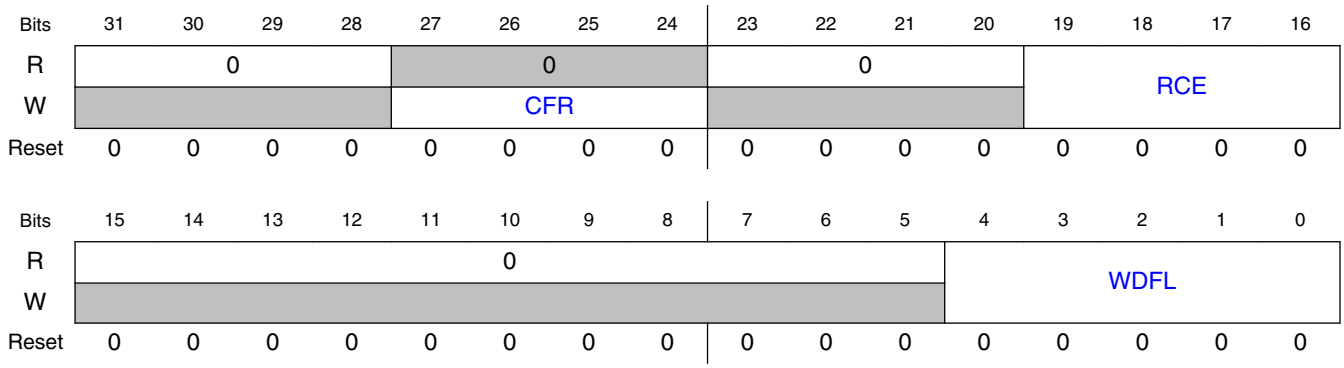
38.5.1.16.1 Offset

Register	Offset
RCR3	94h

38.5.1.16.2 Function

Contains the receive channel settings.

38.5.1.16.3 Diagram



38.5.1.16.4 Fields

Field	Function
31-28 —	Reserved
27-24 CFR	Channel FIFO Reset Resets the FIFO pointers for a specific channel. Reading this field will always return zero. FIFO pointers should only be reset when a channel is disabled or the FIFO error flag is set. The width of CFR field = the number of receive channels (call it N). For example, if CFR is 2 bits wide, then bit position 24 refers to receive channel 1 FIFO pointer and bit position 25 refers to receive channel 2 FIFO pointer. Setting bit 24 resets receive channel 1 FIFO pointer, and setting bit 25 enables receive channel 2 FIFO pointer. Setting bit N will reset receive channel N FIFO pointer. 0b - No effect. 1b - Receive data channel N FIFO is reset. **NOTE:** This field is not supported in every instance. The following table includes only supported registers.
23-20 —	Reserved
19-16 RCE	Receive Channel Enable Enables the corresponding data channel for receive operation. Changing this field will take effect immediately for generating the FIFO request and warning flags, but at the end of each frame for receive operation.

Instance	Field supported in	Field not supported in
SAI1	RCR3	—
SAI2	—	RCR3
SAI3	—	RCR3

Table continues on the next page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>The width of RCE field = the number of receive channels (call it N). For example, if RCE field is 2 bits wide, then bit position 16 refers to receive channel 1 and bit position 17 refers to receive channel 2. Setting bit 16 enables receive channel 1, and setting bit 17 enables receive channel 2. Setting bit N will enable receive channel N.

NOTE: When there is only a single channel, there is no need for individual channel FIFO reset (RCR3[CFR]). In the case of a single channel, use the global FIFO reset (RCSR[FR]).
0b - Receive data channel N is disabled.
1b - Receive data channel N is enabled.

NOTE: This field is not supported in every instance. The following table includes only supported registers.

<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>SAI1</td><td>RCR3</td><td>—</td></tr>
<tr><td>SAI2</td><td>RCR3[16]</td><td>RCR3[19–17]</td></tr>
<tr><td>SAI3</td><td>RCR3[16]</td><td>RCR3[19–17]</td></tr>
</table>
</td></tr>
<tr><td>15-5
—</td><td>Reserved</td></tr>
<tr><td>4-0
WDFL</td><td>Word Flag Configuration

Configures which word the start of word flag is set. The value written should be one less than the word number (for example, write zero to configure for the first word in the frame). When configured to a value greater than the Frame Size field, then the start of word flag is never set.</td></tr>
</table>

38.5.1.17 Receive Configuration 4 (RCR4)

38.5.1.17.1 Offset

Register	Offset
RCR4	98h

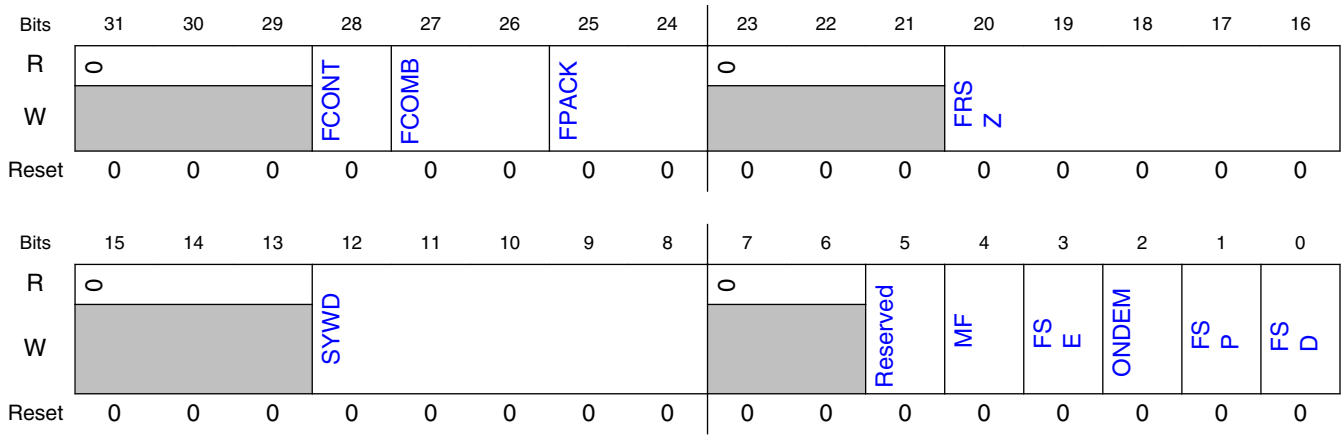
38.5.1.17.2 Function

Contains the receive fields for FIFO Combine Mode, FIFO Packing Mode, and frame sync settings.

NOTE

This register must not be altered when RCSR[RE] is set.

38.5.1.17.3 Diagram



38.5.1.17.4 Fields

Field	Function
31-29 —	Reserved
28 FCONT	FIFO Continue on Error Configures when the SAI will continue receiving after a FIFO error has been detected. 0b - On FIFO error, the SAI will continue from the start of the next frame after the FIFO error flag has been cleared. 1b - On FIFO error, the SAI will continue from the same word that caused the FIFO error to set after the FIFO warning flag has been cleared.
27-26 FCOMB	FIFO Combine Mode When FIFO combine mode is enabled for FIFO reads, software reading any FIFO data register will alternate the read among the enabled data channel FIFOs. For example, if two data channels are enabled then the first read will be performed to the first enabled data channel FIFO and the second read will be performed to the second enabled data channel FIFO. Reseting the FIFO or disabling FIFO combine mode for FIFO reads will reset the pointer back to the first enabled data channel. When FIFO combine mode is enabled for FIFO writes from the receive shift registers, the first enabled data channel input will alternate between the enabled data channel FIFOs. For example, if two data channels are enabled then the first unmasked received word will be stored in the first enabled data channel FIFO and the second unmasked received word will be stored in the second enabled data channel FIFO. Since the first word of the frame is always stored in the first enabled data channel FIFO, it is recommended that the number of unmasked words per frame is evenly divisible by the number of enabled data channels. **NOTE:** This field is not supported in every instance. The following table includes only supported registers. **Instance / Field supported in / Field not supported in** SAI1 / RCR4 / — SAI2 / — / RCR4 SAI3 / — / RCR4

Table continues on the next page...

Field	Function
	00b - FIFO combine mode disabled. 01b - FIFO combine mode enabled on FIFO writes (from receive shift registers). 10b - FIFO combine mode enabled on FIFO reads (by software). 11b - FIFO combine mode enabled on FIFO writes (from receive shift registers) and reads (by software).
25-24 FPACK	FIFO Packing Mode Enables packing of 8-bit data or 16-bit data into each 32-bit FIFO word. If the word size is greater than 8-bit or 16-bit then only the first 8-bit or 16-bits are stored to the FIFO. The first word in each frame always starts with a new 32-bit FIFO word and the first bit shifted must be configured within the first packed word. When FIFO packing is enabled, the FIFO read pointer will only increment when the full 32-bit FIFO word has been read by software. 00b - FIFO packing is disabled 01b - Reserved. 10b - 8-bit FIFO packing is enabled 11b - 16-bit FIFO packing is enabled
23-21 —	Reserved
20-16 FRSZ	Frame Size Configures the number of words in each frame. The value written must be one less than the number of words in the frame. For example, write 0 for one word per frame. The maximum supported frame size is 32 words.
15-13 —	Reserved
12-8 SYWD	Sync Width Configures the length of the frame sync in number of bit clocks. The value written must be one less than the number of bit clocks. For example, write 0 for the frame sync to assert for one bit clock only. The sync width cannot be configured longer than the first word of the frame.
7-6 —	Reserved
5 —	Reserved Software should only write zero to this bit.
4 MF	MSB First Configures whether the LSB or the MSB is received first. 0b - LSB is received first. 1b - MSB is received first.
3 FSE	Frame Sync Early 0b - Frame sync asserts with the first bit of the frame. 1b - Frame sync asserts one bit before the first bit of the frame.
2 ONDEM	On Demand Mode When set, and the frame sync is generated internally, a frame sync is only generated when the FIFO warning flag is clear. 0b - Internal frame sync is generated continuously. 1b - Internal frame sync is generated when the FIFO warning flag is clear.
1 FSP	Frame Sync Polarity Configures the polarity of the frame sync. 0b - Frame sync is active high.

Table continues on the next page...

Field	Function
	1b - Frame sync is active low.
0 FSD	Frame Sync Direction Configures the direction of the frame sync. 0b - Frame Sync is generated externally in Slave mode. 1b - Frame Sync is generated internally in Master mode.

38.5.1.18 Receive Configuration 5 (RCR5)

38.5.1.18.1 Offset

Register	Offset
RCR5	9Ch

38.5.1.18.2 Function

This register must not be altered when RCSR[RE] is set.

38.5.1.18.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0			WNW					0			W0W				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0			FBT					0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

38.5.1.18.4 Fields

Field	Function
31-29 —	Reserved
28-24 WNW	Word N Width

Table continues on the next page...

Field	Function
	Configures the number of bits in each word, for each word except the first in the frame. The value written must be one less than the number of bits per word. Word width of less than 8 bits is not supported.
23-21 —	Reserved
20-16 W0W	Word 0 Width Configures the number of bits in the first word in each frame. The value written must be one less than the number of bits in the first word. Word width of less than 8 bits is not supported if there is only one word per frame.
15-13 —	Reserved
12-8 FBT	First Bit Shifted Configures the bit index for the first bit received for each word in the frame. If configured for MSB First, the index of the next bit received is one less than the current bit received. If configured for LSB First, the index of the next bit received is one more than the current bit received. The value written must be greater than or equal to the word width when configured for MSB First. The value written must be less than or equal to 31-word width when configured for LSB First.
7-0 —	Reserved

38.5.1.19 Receive Data (RDR0 - RDR3)

38.5.1.19.1 Offset

Register	Offset
RDR0	A0h
RDR1	A4h
RDR2	A8h
RDR3	ACh

38.5.1.19.2 Function

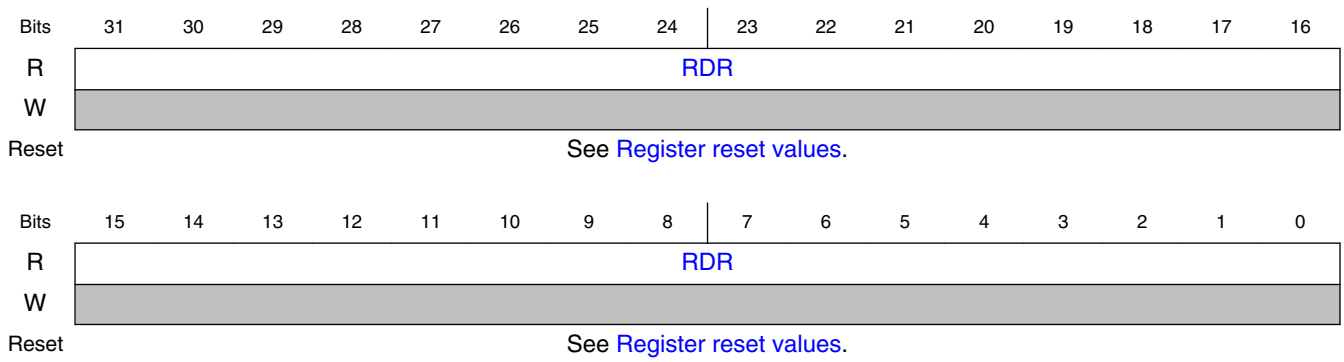
Reads from this register when the receive FIFO is not empty return the data from the top of the receive FIFO. Reads from this register when the receive FIFO is empty are ignored.

NOTE

Each module instance supports a different number of registers.

Instance	Register supported	Register not supported
SAI1	RDR0–RDR3	—
SAI2	RDR0	RDR1–RDR3
SAI3	RDR0	RDR1–RDR3

38.5.1.19.3 Diagram



38.5.1.19.4 Register reset values

Register	Reset value
RDR0	SAI1–SAI3: 0000_0000h
RDR1–RDR3	0000_0000h

38.5.1.19.5 Fields

Field	Function
31-0 RDR	Receive Data Register

38.5.1.20 Receive FIFO (RFR0 - RFR3)

38.5.1.20.1 Offset

Register	Offset
RFR0	C0h
RFR1	C4h
RFR2	C8h
RFR3	CCh

38.5.1.20.2 Function

The MSB of the read and write pointers is used to distinguish between FIFO full and empty conditions. If the read and write pointers are identical, then the FIFO is empty. If the read and write pointers are identical except for the MSB, then the FIFO is full.

NOTE

Each module instance supports a different number of registers.

Instance	Register supported	Register not supported
SAI1	RFR0–RFR3	—
SAI2	RFR0	RFR1–RFR3
SAI3	RFR0	RFR1–RFR3

38.5.1.20.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0										WFP					
W																
Reset	See Register reset values.															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RCP	0									RFP					
W																
Reset	See Register reset values.															

38.5.1.20.4 Register reset values

Register	Reset value
RFR0	SAI1–SAI3: 0000_0000h
RFR1–RFR3	0000_0000h

38.5.1.20.5 Fields

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td>31-22
—</td><td>Reserved</td></tr>
<tr><td>21-16
WFP</td><td>Write FIFO Pointer
FIFO write pointer for receive data channel.</td></tr>
<tr><td>15
RCP</td><td>Receive Channel Pointer
When FIFO Combine mode is enabled for reads, indicates that this data channel is the next FIFO to be read.
NOTE: This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>SAI1</td><td>RFR0–RFR3</td><td>—</td></tr>
<tr><td>SAI2</td><td>—</td><td>RFR0</td></tr>
<tr><td>SAI3</td><td>—</td><td>RFR0</td></tr>
</table>
0b - No effect.
1b - FIFO combine is enabled for FIFO reads and this FIFO will be read on the next FIFO read.</td></tr>
<tr><td>14-6
—</td><td>Reserved</td></tr>
<tr><td>5-0
RFP</td><td>Read FIFO Pointer
FIFO read pointer for receive data channel.</td></tr>
</table>

38.5.1.21 Receive Mask (RMR)

38.5.1.21.1 Offset

Register	Offset
RMR	E0h

38.5.1.21.2 Function

This register is double-buffered and updates:

1. When RCSR[RE] is first set
2. At the end of each frame

This allows the masked words in each frame to change from frame to frame.

38.5.1.21.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	RWM															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RWM															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

38.5.1.21.4 Fields

Field	Function
31-0 RWM	Receive Word Mask Configures whether the receive word is masked (received data ignored and not written to receive FIFO) for the corresponding word in the frame. 00000000000000000000000000000000b - Word N is enabled. 00000000000000000000000000000001b - Word N is masked.

Chapter 39
Medium Quality Sound (MQS)

39.1 Chip-specific MQS information

Table 39-1. Reference links to related information

Topic	Related module or subsystem	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Audio Subsystem	Audio Subsystem	Audio Subsystem Overview
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
Interrupts, DMA Events and XBAR Assignments	-	Interrupts, DMA Events and XBAR Assignments

39.2 Overview

Medium quality sound (MQS) is used to generate medium quality audio via a standard GPIO in the pinmux, allowing the user to connect stereo speakers or headphones to a power amplifier without an additional DAC chip.

39.2.1 Block Diagram

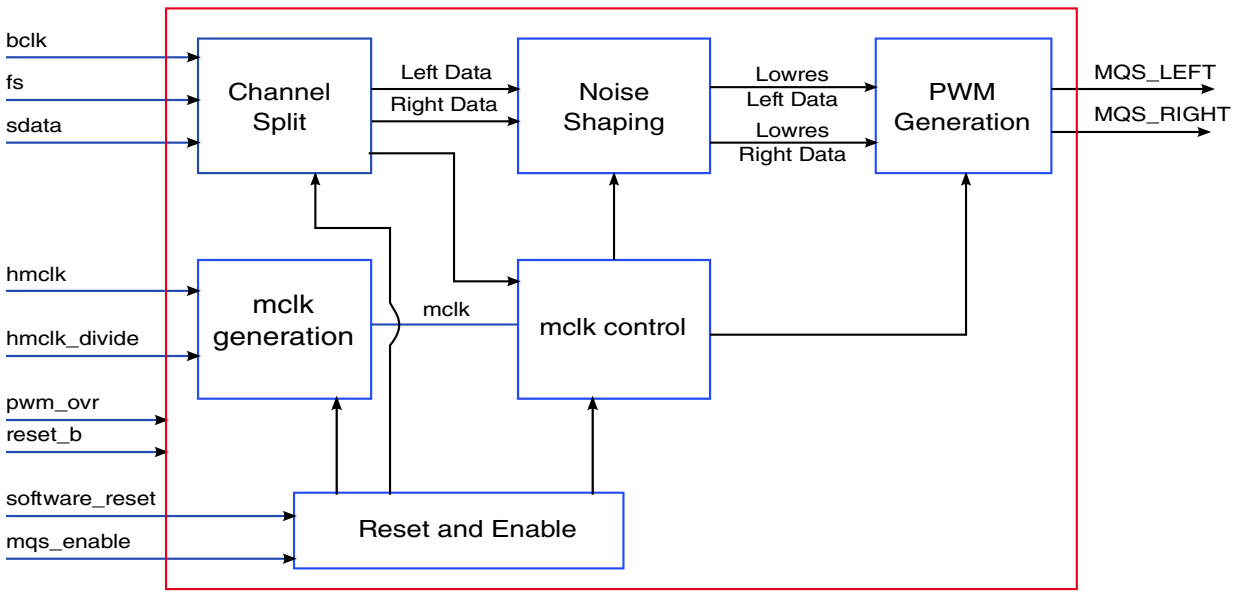


Figure 39-1. Block Diagram

39.2.2 Features

MQS has the following sub-modules:

- Channel Split: Splits the I2S signals into separate left channel and right channel audio data.
- Noise Shaping: Uses the sigma-delta algorithm to generate low-resolution, very high sampling audio, while the audio sampling rate is increased.
- PWM Generation: Generates the bit stream to the GPIO, which is then used to drive the amplifier and then to drive the external speakers or headphones.
- Mclk Generation: Used to generate the master clock (mclk). The frequency of mclk is determined by the final bit duration of PWM generation module.
- Mclk Control:Used as a metronome to co-ordinate the different functional blocks working synchronously.
- Reset and Enable: Used to generate the reset and enable logic to different clock domains.

39.3 Functional Description

MQS accepts the following inputs:

- 2-channel, LSB-valid 16-bit, MSB shift-out first serial data (sdata)

- Frame sync asserting with the first bit of the frame (fs)
- Bit clock used to shift data out on the positive clock edge (bclk)

The signals from the SAI3 are in left_justified format. MQS provides the SNR target as no more than 20dB for the signals below 10 kHz. The signals above 10 kHz will have worse THD+N values.

MQS provides only simple audio reproduction. No internal pop, click or distortion artifact reduction methods are provided.

39.4 External Signals

The following table describes the external signals of MQS:

Table 39-2. MQS External Signals

Signal	Description	Direction
MQS_LEFT	Left signal output	O
MQS_RIGHT	Right signal output	O

39.4.1 Interface Signals

MQS module has the following interface signals.

Signal Name	In/Out	BitWidth	Description	Comments
reset_b	In	1	asynchronous reset	
software_reset	In	1	Software reset	From IOMUXC_GPR2
mqs_enable	In	1	module enable	From IOMUXC_GPR2
pwm_ovr	In	1	PWM oversampling ratio 1=64, 0=32	From IOMUXC_GPR2
hmclk	In	1	Maximum bit clock, used to generate the mclk, divider ratio is controlled by mqs_hmclk_divide	Max 66.5MHzTypical 24.576MHz
hmclk_divide	In	8	Divider ration control for mclk from hmclk	From IOMUXC_GPR2
bclk	In	1	bit clock from I2S signal	
fs	In	1	frame sync clock from I2S signal	
sdata	In	1	serial audio data from I2S signal	

39.5 Application Information

MQS has no internal programmable registers. But it does have some programmability from IOMUXC_GPR2.

Register Bitfields	Description
IOMUXC_GPR2[MQS_OVERSAMPLE]	Used to control the PWM oversampling rate compared with mclk. 1—64, 0—32.
IOMUXC_GPR2[MQS_EN]	MQS enable. 1—Enable MQS, 0—Disable MQS
IOMUXC_GPR2[MQS_SW_RST]	MQS software reset. 1—Enable software reset for MQS 0—Exit software reset for MQS
IOMUXC_GPR2[MQS_CLK_DIV]]	Divider ration control for mclk from hmclk. 0—mclk frequency = hmclk frequency; 1—mclk frequency = ½*hmclk frequency; 2—mclk frequency = 1/3*hmclk frequency; …; n—mclk frequency = 1/(n+1)*hmclk frequency

39.5.1 Usage Model

The user needs to program the SAI3 to output 2-channel LSB-16 bit active left-justified signal, and then program the related IOMUXC_GPR2 bits.

Due to the different devices connected to MQS, and different high frequency behaviors of the connected analog circuits, the user needs choose the appropriate MQS_CLK_DIV and MQS_OVERSAMPLE values for the best audible effects.

Chapter 40
Sony/Philips Digital Interface (SPDIF)

40.1 Chip-specific SPDIF information

Table 40-1. Reference links to related information

Topic	Related module or subsystem	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Audio Subsystem	Audio Subsystem	Audio Subsystem Overview
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
Interrupts, DMA Events and XBAR Assignments	-	Interrupts, DMA Events and XBAR Assignments

40.2 Overview

The Sony/Philips Digital Interface (SPDIF) audio block is a stereo transceiver that allows the processor to receive and transmit digital audio.

40.2.1 Block Diagram

The figure below shows a block diagram of the SPDIF transceiver data paths (receiver and transmitter) and its interface.

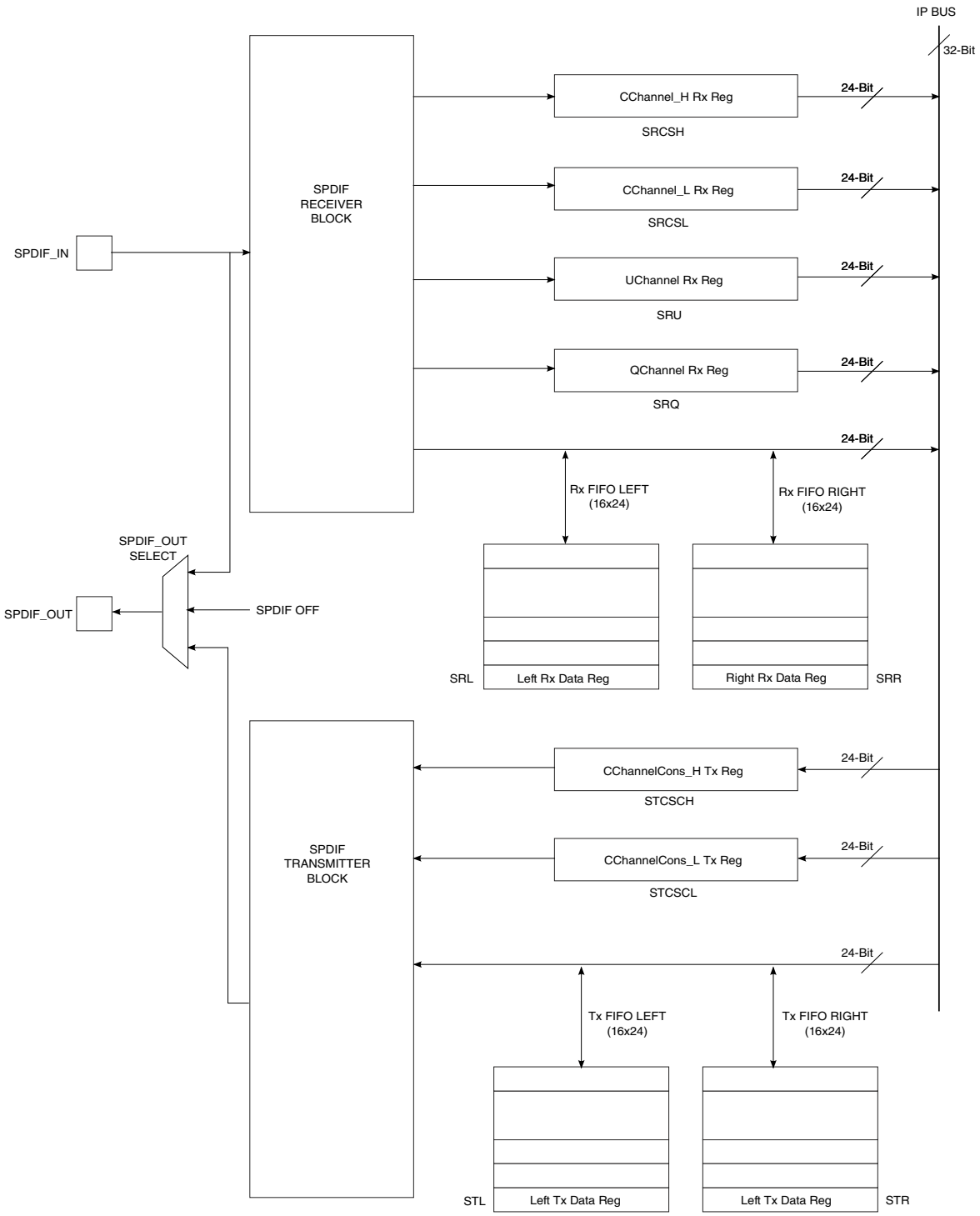


Figure 40-1. SPDIF Transceiver Data Interface Block Diagram

40.2.2 Features

- Allows the handling of both SPDIF channel status (CS) and User (U) data
- Includes a frequency measurement block that allows the precise measurement of an incoming sampling frequency
- Provides a recovered clock to drive both internal components in the system, such as SAI ports, and external components, such as A/Ds or D/As, with clocking control provided via related registers

40.3 Functional Description

The SPDIF is composed of two parts: SPDIF Receiver and SPDIF Transmitter.

The SPDIF receiver extracts the audio data from each SPDIF frame and places the data in the SPDIF Rx left and right FIFOs. The Channel Status and User Bits are also extracted from each frame and placed in the corresponding registers. The SPDIF receiver also provides a bypass option for direct transfer of the SPDIF input signal to the SPDIF transmitter.

For the SPDIF transmitter, the audio data is provided by the processor via the SPDIFTxLeft and SPDIFTxRight registers. The Channel Status bits are also provided via the corresponding registers. The SPDIF transmitter generates a SPDIF output bitstream in the biphase mark format (IEC60958), which consists of audio data, channel status and user bits.

In the SPDIF transmitter, the IEC60958 biphase bit stream is generated on both edges of the SPDIF Transmit clock. The SPDIF Transmit clock is generated by the SPDIF internal clock generate block and the sources are from outside of the SPDIF block. For the SPDIF receiver, it can recover the SPDIF Rx clock.

Figure 40-2 shows the clock structure of the SPDIF transceiver.

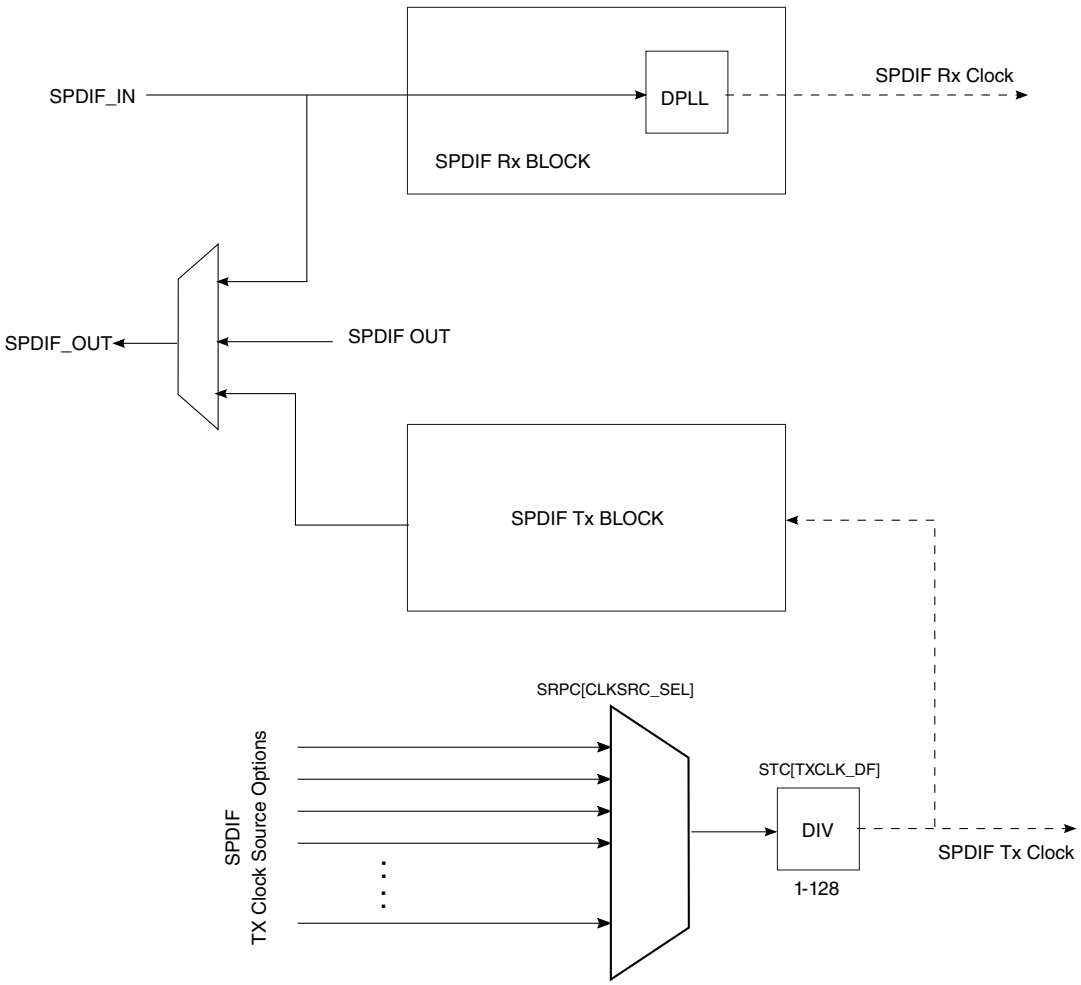


Figure 40-2. SPDIF Transceiver Clock Diagram

40.3.1 SPDIF Receiver

The SPDIF receiver extracts the audio data from each SPDIF frame and places the data in Rx left and right FIFOs.

The Tx left and right FIFOs are 16-deep and 24-bit-wide (equal to the audio data width). The Channel Status and User Bits are also extracted from each frame and placed in corresponding registers. The SPDIF receiver also provides a bypass option for direct transfer of the SPDIF input signal to the SPDIF transmitter.

The SPDIF receiver handles the main data audio stream and recovers the bit clock from the SPDIF input signal. The sample rate can be determined from the frequency measuring block. Additionally, the receiver supports the SPDIF C and U channels. The SPDIF C and U channel data is interfaced directly to memory-mapped registers.

All the data registers are controlled by the Interrupt Control Block and transferred to the memory-mapped IP bus.

40.3.1.1 Audio Data Reception

The SPDIF Receiver block extracts the audio data from the IEC60958 stream, and outputs this via Rx left and right FIFOs to the memory-mapped registers SPDIFRxLeft and SPDIFRxRight.

Data from the SPDIF receiver is buffered in receive FIFO, and can be read by the processor from the memory-mapped registers.

40.3.1.1.1 SPDIF receiver data registers - Behavior on overrun, underrun

The SPDIF Data Receive registers (SPDIFRxLeft and SPDIFRxRight) have individual FIFOs for left and right channel. As a result, there is always the possibility that left and right FIFOs may go out of sync due to FIFO underruns and FIFO overruns that affect only one part (left or right) of any FIFOs. To prevent this from happening, hardware has been added to the device. Two mechanisms to prevent mismatch between the FIFOs are available.

If a SPDIF Data Rx FIFO overrun occurs on e.g. the right half of the FIFO, the sample that caused the overrun is not written to the right half (due to overrun). Special hardware will make sure the next sample is not written to the left half of the FIFO. If the overrun occurs on the left half of the FIFO, the next sample is not written to the right half of the FIFO.

40.3.1.1.2 SPDIF receiver data registers - Automatic resynchronization of FIFOs

An automatic FIFO resynchronization feature is available. It can be enabled and disabled separately for every FIFO. If it is enabled, the hardware will check to see if the left and right FIFOs are in sync. If that is not the case, it will set the filling pointer of the right FIFO to be equal to the filling pointer of the left FIFO.

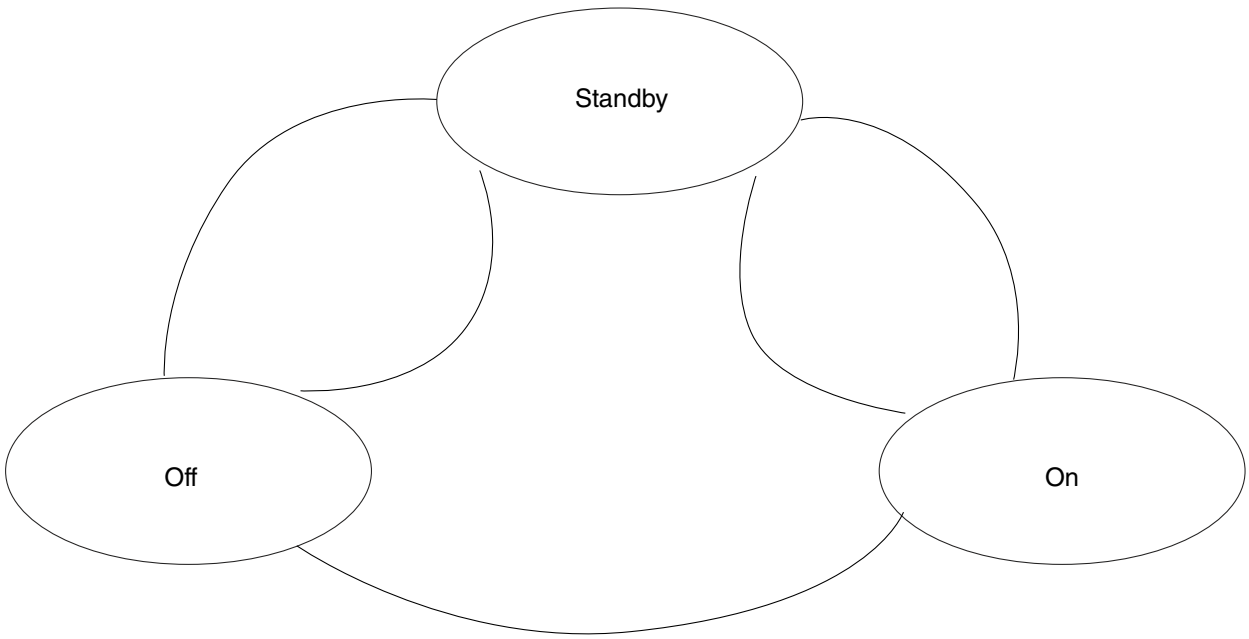


Figure 40-3. FIFO Auto-resync Controller State Machine

The operation is explained from the state diagram shown above. Every FIFO auto-resync controller has a state machine with 3 states: Off, StandBy and On. In the On state, the filling of the left FIFO is compared with the filling of right, and if they are not equal, right is made equal to left, and an interrupt is generated.

The controller will stay in Off state when the feature is disabled. When not disabled, the state machine will go to Off state on any processor read or write to the FIFO. It will go from On or Off to Standby on any left sample read from SPDIF Tx FIFOs, or on any left sample write to SPDIF Rx FIFOs. The controller will go from Standby to On on any right sample read from SPDIF Tx FIFO, or on any right sample write to SPDIF Rx FIFO. There is a control bit in the SPDIF Configuration Register (SCR) to enable/disable the feature for the SPDIF Rx FIFO and SPDIF Tx FIFO.

See Application Information for additional details.

40.3.1.2 Channel Status Reception

40.3.1.2.1 48bits Mode

A total of 48 channel status bits are received in two registers. No interpretation is performed by the SPDIF receiver block.

Channel Status Bits are ordered first bit left. CS-channel MSB bit "0" is located in bit position 23 in the memory-mapped register SPDIFRxCChannel_h. CS-channel bit "23" is considered the LSB bit 0 in the register. C-channel bit 24 to 47 is seen as [23:0] bits of register SPDIFRxCChannel_l.

40.3.1.2.2 Channel Status Interrupt

When the value of a new SPDIF "CS" channel status frame is loaded in the register, an interrupt is generated. The interrupt is cleared when the processor writes the corresponding bit in the InterruptStat register.

40.3.1.3 User Bit Reception

There are two modes for U Channel reception, CD and non-CD. As is decided by USyncMode (bit 1 of CDText_Control register).

40.3.1.3.1 Behavior of U Channel receive interface on incoming CD U Channel Sub-code in SPDIF receiver

This mode is selected if UsyncMode, bit 1 in register CD Text control is set "1".

The CD sub-code stream embedded into the SPDIF U channel consists of a sequence of packets. Every packet is made up 98 "symbols". The first two symbols of every packet are "sync symbols", the other 96 symbols are "data symbols".

Any sequence found in the SPDIF U channel stream starting with a leading one, followed by 7 information bits, is recognized as a "data symbol". Subsequent data symbols are separated by "pauses". During the "pause", "zero bits" are seen on the SPDIF U channel.

Data symbols are coming in MSB first. The MSB is the leading one.

When a "long pause" is seen between 2 subsequent "data symbols", the SPDIF receiver will assume the reception of one or more "sync symbols". Table below gives details.

Table 40-2. Sync Control Bits

Number of U Channel zero bits	Corresponding number of sync symbols
0-1	Unpredictable, not allowed
2-10	0
11-22	1
23-34	2
35-46	3
>45	Unpredictable, not allowed

The recognition of the number of sync symbols derives from the fact that the U channel transmitter in the CD channel decoder will transmit one symbol on average every 12 SPDIF channel bits. On this average rate, there is a maximum tolerance of 5%.

The SPDIF receiver is tolerant of symbol errors. Due to the physical nature of the transmission of the data over the CD disc, not more than 1 out of any 5 consecutive user channel symbols may be in error. The error may cause a change in data value, which is not detected by this interface, or it may cause a data symbol to be seen as a sync symbol, or a sync symbol to be seen as a data symbol. However, not more than 1 out of any 5 consecutive user channel symbols should be affected in this way.

The SPDIF U channel circuitry recognizes the 98-symbol packet structure, and sends the 96 symbol payload to the processor application. The 96 symbol payload is transmitted to the processor via 2 registers:

- The SPDIFRxUChannel register. In this register, data is presented 3 symbols at the time to the processor. Every time 3 new valid symbols, received on the SPDIF U Channel are present, the UChannelRxFull interrupt is asserted. For one 98-symbol packet, 96 symbols are carried across SPDIFRxUChannel. To transfer all this data, 32 UChannelRxFull interrupts are generated.
- The QChannelReceive register. In this register, only the Q bit of the packet is accumulated.Operation is similar to UChannelReceive. Because only Q-bit is transferred, only 96 Q-bits are transferred for any 98-symbol packet. To transfer this data, 4 QChannelRxFull interrupts are generated. When QChannelRxFull occurs, it is coincident with UChannelRxFull. There is only one QChannelRxFull for every 8 UChannelRxFull. The convention is that most significant data is transmitted first, and is left-aligned in the registers.
- Timing regarding packet boundary is extracted by hardware. The last UChannelRxFull corresponding to a given packet should be coincident with the last QChannelRxFull. In this last U, Q channel interrupt, symbols 95-98 are received, Q channel bits 67-98. The interrupts are coincident with UQSyncFound, flagging last symbols of the current frame.
- When the start of the new packet is found before the current packet is complete (less than 98 symbols in the packet), the UQFrameError interrupt is set. The application software should read out UChannelReceive and QchannelReceive registers, discard the value, and assume the start of a new packet.
- As already said, packet sync extraction is tolerant for single-symbol errors. Packet sync detection is based on the recognition of the sequence data-sync-sync-data in the symbol stream, because this is the only syncing sequence that is not affected by single errors. If the sync symbols are not found 98 symbols after the previous

occurrence, it is assumed to be destroyed by channel error, and a new sync symbols is interpolated.
- Normally, only data bytes are passed to the application software. Every databyte will have its most significant bit set. If sync symbols are passed to the application software, they are seen as all-zero symbols. Sync symbols can only end up in the data stream due to channel error.

40.3.1.3.2 Behavior of U Channel receive interface on incoming non-CD data

This mode is selected if UsyncMode, bit 1 in register CD Text control is set'0'.

In non-CD mode, the SPDIF U channel stream is recognized as a sequence of "data symbols". No packet recognition is done.

Any sequence found in the SPDIF U channel stream starting with a leading one, followed by 7 information bits, is recognized as a "data symbol". Subsequent data symbols are separated by "pauses". During the "pause", "zero bits" are seen on the SPDIF U channel.

3 consecutive data symbols seen in the SPDIF U Channel stream are grouped together into the SPDIFRxUChannel register. First symbol is left, last symbol is right aligned. When SPDIFRxUChannel contains 3 new data symbols, UChannelRxFull is asserted.

In this mode, the operation of QchannelRx and associated interrupt QchannelRxFull is reserved, undefined. And the operation of UQFrameError and UQSyncFound is also reserved, undefined.

The U channel is extracted, and output by the SPDIF Rx on SPDIFRxUChannel-Stream.

When incoming SPDIF data parity error or bit error is detected, and if the next SPDIF word for that channel is error-free, the SPDIF word in error is replaced with the average of the previous word and next word. When incoming SPDIF data parity error or bit error is detected, and the next SPDIF word is in error, the previous SPDIF word is repeated.

40.3.1.4 Validity Flag Reception

An interrupt is associated with the Validity flag. (interrupt 16 - SPDIFValNoGood). This interrupt is set every time a frame is seen on the SPDIF interface with the validity bit set to "invalid".

40.3.1.5 SPDIF Receiver Interrupt Exception Definition

Several SPDIF exceptions can trigger an interrupt. They are:

- Control Status channel change. Set when SPDIFRxCChannel_1 register is updated. The register is updated for every new C-Channel received. The exception is reset on write to InterruptClear register.
- SPDIF Illegal Symbol. Set on reception of illegal symbol during SPDIF receive. Reset by writing register InterruptClear.[1]
- SPDIF bit error. Set on reception of bit error. (Parity bit does not match). Reset on write to InterruptClear register.
- Receive data FIFO full. Set when SPDIF receive data FIFO is full.
- Receive data FIFO underrun/overrun. Set when there is a underrun/overrun on the SPDIF receive data FIFO.
- Receive data FIFO resynchronization. Set when a resynchronization event occurs on the SPDIF receive data FIFO.
- Receive U Channel buffer full. Set when next 24 bits of U channel code are available.
- Receive Q Channel buffer overrun. Set when Q channel buffer overrun.
- Receive U Channel buffer overrun. Set on U channel buffer overrun.
- Receive Q Channel buffer full. Set when next 24 bits of Q channel code are available.
- Receive UQ sync found. Set when UQ channel sync found.
- Receive UQ frame error. Set when UQ frame error found.

40.3.1.6 Standards Compliance

The SPDIF interface is compatible with the Tech 3250-E standard of the European Broadcasting Union, except clause 6.3.3 and the IEC60958-3 Ed2 for relevant topics.

Supported input frequency range is 12 Khz up to 96 Khz. (fully compliant) and 96 Khz up to 176 Khz (Can interface with compliant SPDIF transmitter within same cabinet, making reasonable assumptions on jitter added due to interconnecting wire.)

Tolerated jitter on SPDIF input signals are 0.25 bit peak-peak for high frequencies. There is no jitter limit for low frequencies. The user channel extraction in CD mode is capable of coping with single-symbol errors, and still retrieve U channel frames on correct boundaries. This capability is required for reliable reception of CD-Text from some Philips CD channel decoders. This capability was deemed more important than compliance with the IEC60958 annex A.3 standard, and for this reason user channel

1. The SPDIF input is a biphase/mark modulated signal. The time between any two successive transitions of the SPDIF signal is always 1, 2 or 3 SPDIF symbol periods long. The SPDIF receiver will parse the stream, and split it in so-called symbols. It recognizes s1, s2 and s3 symbols, depending on the length of the symbols. Not all sequences of these symbols are allowed. To give an example, a sequence s2-s1-s1-s1-s2 cannot occur in a no-error SPDIF signal. If the receiver finds such an illegal sequence, the illegal symbol interrupt is set. No corrective action is undertaken. When the interrupt occurs, this means that(a) The SPDIF signal is destroyed by noise (b) The SPDIF frequency changed.

reception is not compliant with IEC60958 annex A.3. However, the interface is capable to receive U channel inserted by a typical CD channel decoder. Also, in this case, it is more robust and tolerant for channel error than what is required by IEC60958 annex A.3.

40.3.1.7 SPDIF PLOCK Detection and Rxclk Output

Using the high speed system clock, the internal DPLL can extract the bit clock (advanced pulse) from the input bitstream. When this internal DPLL is locked, the LOCK bit of PhaseConfig Register will be set, and the SPDIF Lock output pin SPDIF_LOCK will be asserted.

After DPLL has locked, the pulses are generated, and the average pulse rate is 128 x the sampling frequency.(For a 44.1 Khz input sampling frequency, the average pulse rate = 128 x 44.1 Khz.) The pulse signal is used in the FreqMeas circuit to generate the frequency measurement result.

40.3.1.8 Measuring Frequency of SPDIF_RxClk

The internal DPLL can extract the bit clock (advanced plus) from the input bitstream. To do that, it is necessary to measure the frequency of the incoming signal in relationship with the system clock (BUS_CLK).

Associated with it are two registers, PhaseConfig and FreqMeas. The circuit will measure the frequency of the incoming clock as a function of the BUS_CLK. The circuit is a second-order filter. The output is a value represented by an unsigned number stored in the 24-bit FreqMeas register, giving the frequency of the source as a function of the BUS_CLK.

FreqMeas[23:0] = FreqMeas_CLK / BUS_CLK * 2^{10} * GAIN.

For example, if the GAIN is selected as 8*(2**10) (PhaseConfig[5:3] = 3'b011), the actual result

FreqMeas_CLK / BUS_CLK is equal to FreqMeas[23:0] / 2^{23}.

40.3.2 SPDIF Transmitter

Audio data for the SPDIF transmitter is provided by processor via the SPDIFTxLeft and SPDIFTxRight registers.

Clocking for SPDIF transmitter is selected through a multiplexer from several clock sources (see register bitfield description STC[TxClk_Source] for transmit clock source inputs). The SPDIF transmitter clock source can be divided down as needed using Txclk_DF. The SPDIF transmitter output can be chosen from either the SPDIF transmitter block, directly from the SPDIF receiver (via the output multiplexer), or disabled.

The SPDIF transmitter generates a SPDIF output bitstream in IEC60958 biphase mark format, consisting of audio data, channel status.

40.3.2.1 Audio Data Transmission

Audio data for the SPDIF transmitter is provided by the processor via SPDIFTxLeft and SPDIFTxRight registers. They send audio data to Tx left and right FIFOs. The Tx left and right FIFOs are also 16-deep and 24-width (equal to the audio data width).

40.3.2.1.1 SPDIF transmitter data registers - Behavior on overrun, underrun

The SPDIF Data Transmit registers (SPDIFTxLeft and SPDIFTxRight) have individual FIFOs for left and right channel. As a result, there is always the possibility that left and right FIFOs may go out of sync due to FIFO underruns and FIFO overruns that affect only one part (left or right) of any FIFO. To prevent this from happening, hardware has been added on the device. Two mechanisms to prevent mismatch between the FIFOs are available.

If SPDIF Tx FIFO underruns on the right half of the FIFO, no sample leaves that FIFO (because it was already empty). Special hardware will make sure that the next sample read from the left FIFO will not leave the FIFO (no read strobe is generated). If the underrun occurs on the left half of the FIFO, next read strobe to the right FIFO is blocked.

40.3.2.1.2 SPDIF transmitter data registers - Automatic resynchronization of FIFOs

See Audio Data Reception and Application Information.

40.3.2.2 Channel Status Transmission

40.3.2.2.1 48bits Mode

A total of 48 Consumer channel status bits are transmitted from two registers. Channel Status Bits are ordered first bit left.

CS-channel MSB bit "0" is located in bit position 23 in the memory-mapped register SPDIFTxCChannelCons_h. CS-channel bit "23" is considered bit 0 in the register. C-channel bits 24-47 are seen as MSB-LSB bits of register SPDIFTxCChannelCons_l.

40.3.2.3 Validity Flag Transmission

The validity bit setting is performed via the ValCtrl bit of the SPDIF_SCR register.

40.3.3 Clocks

The table found here describes the clock sources for SPDIF.

Please see clock control block for clock setting, configuration and gating information.

Table 40-3. SPDIF Clocks

Clock name	Description
gclkw_t0	Global clock
ipg_clk_s	Peripheral access clock
tx_clk	Module Tx clock

40.4 External Signals

The following table describes the external signals of SPDIF:

Table 40-4. SPDIF External Signals

Signal	Description	Direction
SPDIF_EXT_CLK	External clock signal	I
SPDIF_IN	Input line	I
SPDIF_LOCK	Lock signal	O
SPDIF_OUT	Output line signal	O
SPDIF_SR_CLK	RX Clock	O
SPDIF_OUT_CLK	TX Clock	O

40.5 Application Information

The automatic FIFO resynchronization can be switched on, and will avoid all mismatches between left and right FIFOs, if the software obeys the following rules:

1. When the left data is read or written to the left FIFO, in the same place of the program, data must be read or written to the right FIFO. Maximum time difference between left and right is 1/2 sample clock. (E.g. if sample frequency is 44 Khz, approximately 10 micro-seconds. For 88 Khz, approximately 5 micro-seconds.)
2. Write/read data to FIFOs at least 2 samples at the time. If there is a mismatch Left-Right, the resync logic may go on only 1 sample clock after last data is read/written to the FIFO. Also acceptable is polling the FIFO, if at least part of the time 2 samples will be read/written to it.
3. The first 192 frames (after DPLL lock) must be discarded.

40.5.1 SPDIF receiver details

There are three exceptions associated with the SPDIF Receiver FIFOs

- full
- under/overrun
- resync

When the "full" condition is set for processor data input registers, the processor should read data from the FIFO, before overrun occurs. When "full" is set, and the FIFO contains e.g. 6 samples, it is acceptable for the software to read first 6 samples from the LEFT address, followed by 6 samples from the RIGHT address, or 6 samples from the RIGHT address, followed by 6 samples from the LEFT address, or 1 sample LEFT, followed by 1 sample RIGHT repeated 6 times. There is no order specified.

The implementation for SPDIF Rx is a double FIFO, one for left and one for right. "full" is set when both FIFOs are full. "underrun, overrun" are set when one of the FIFOs do underrun or do overrun. The resync interrupt means hardware took special action to resynchronize left and right FIFOs.

The FIFO level at which the "full" interrupt is generated, is programmable via the Full Select field in the SPDIFConfigReg register.

40.5.2 Rx FIFO on and Rx FIFO reset

Two additional control fields of the SPDIF Rx FIFO are the on/off select and FIFO reset fields.

If on/off select is set to off, all-zero will be read from the FIFO, irrespective of the data received over the SPDIF interface.

If FIFO reset is set, the FIFO is blocked at "1 sample in FIFO". In this, the full interrupt will be on if FullSelect is set to "00". If FullSelect is set to any other value, interrupt will be off. The other interrupts are always off.

40.5.3 SPDIFTxLeft, SPDIFTxRight details

With SPDIF Tx FIFOs three exceptions are associated:

- empty
- under/overrun
- resync

When the empty condition is set for processor data output registers, the processor should write data to the FIFO, before underrun occurs. When empty is set and, for instance, 6 samples need to be written, it is acceptable for the software to write first 6 samples from the LEFT address, followed by 6 samples from the RIGHT address, or 1 sample LEFT, followed by 1 sample RIGHT repeated 6 times. Left should be written before right. The implementation of all data out FIFOs is a double FIFO, one for left and one for right. Empty is set when both FIFOs are empty. Underrun, overrun are set when one of the FIFOs do underrun or do overrun. Resync is set when the hardware resynchronizes left and right FIFOs.

On receiving underrun, overrun interrupt, synchronization between Left and Right words in the FIFOs may be lost. Synchronization will not be lost when the underrun or overrun comes from the IEC60958 side of the FIFO. If the processor reads or writes more data from, for example, left than from right, synchronization will be lost. If automatic resynchronization is enabled, and if the software obeys the rules to let this work, resynchronization will be automatic.

40.6 SPDIF Memory Map/Register Definition

40.6.1 SPDIF register descriptions

40.6.1.1 SPDIF memory map

SPDIF base address: 4038_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	SPDIF Configuration Register (SCR)	32	RW	0000_0400h
4h	CDText Control Register (SRCD)	32	RW	0000_0000h
8h	PhaseConfig Register (SRPC)	32	RW	0000_0000h
Ch	InterruptEn Register (SIE)	32	RW	0000_0000h
10h	InterruptClear Register (SIC)	32	WO	0000_0000h
10h	InterruptStat Register (SIS)	32	RO	0000_0002h
14h	SPDIFRxLeft Register (SRL)	32	RO	0000_0000h
18h	SPDIFRxRight Register (SRR)	32	RO	0000_0000h
1Ch	SPDIFRxCChannel_h Register (SRCSH)	32	RO	0000_0000h
20h	SPDIFRxCChannel_l Register (SRCSL)	32	RO	0000_0000h
24h	UchannelRx Register (SRU)	32	RO	0000_0000h
28h	QchannelRx Register (SRQ)	32	RO	0000_0000h
2Ch	SPDIFTxLeft Register (STL)	32	WO	0000_0000h
30h	SPDIFTxRight Register (STR)	32	WO	0000_0000h
34h	SPDIFTxCChannelCons_h Register (STCSCH)	32	RW	0000_0000h
38h	SPDIFTxCChannelCons_l Register (STCSCL)	32	RW	0000_0000h
44h	FreqMeas Register (SRFM)	32	RO	0000_0000h
50h	SPDIFTxClk Register (STC)	32	RW	0002_0F00h

40.6.1.2 SPDIF Configuration Register (SCR)

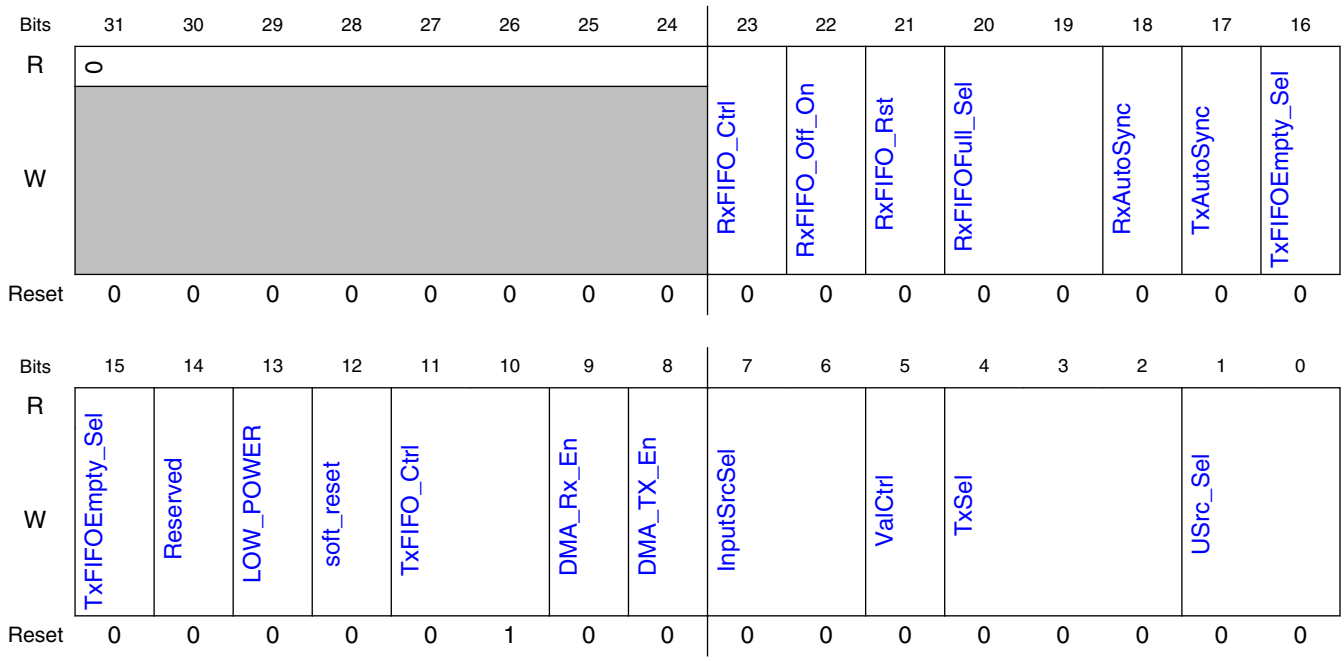
40.6.1.2.1 Offset

Register	Offset
SCR	0h

40.6.1.2.2 Function

Contains SPDIF configuration selections.

40.6.1.2.3 Diagram



40.6.1.2.4 Fields

Field	Function
31-24 —	Reserved
23 RxFIFO_Ctrl	RxFIFO_Ctrl 0b - Normal operation 1b - Always read zero from Rx data register
22 RxFIFO_Off_On	RxFIFO_Off_On 0b - SPDIF Rx FIFO is on 1b - SPDIF Rx FIFO is off. Does not accept data from interface
21 RxFIFO_Rst	RxFIFO_Rst 0b - Normal operation 1b - Reset register to 1 sample remaining
20-19 RxFIFOFull_Sel	RxFIFOFull_Sel 00b - Full interrupt if at least 1 sample in Rx left and right FIFOs 01b - Full interrupt if at least 4 sample in Rx left and right FIFOs 10b - Full interrupt if at least 8 sample in Rx left and right FIFOs 11b - Full interrupt if at least 16 sample in Rx left and right FIFO
18 RxAutoSync	RxAutoSync 0b - Rx FIFO auto sync off 1b - RxFIFO auto sync on
17 TxAutoSync	TxAutoSync 0b - Tx FIFO auto sync off 1b - Tx FIFO auto sync on

Table continues on the next page...

Field	Function
16-15 TxFIFOEmpty_Sel	TxFIFOEmpty_Sel 00b - Empty interrupt if 0 sample in Tx left and right FIFOs 01b - Empty interrupt if at most 4 sample in Tx left and right FIFOs 10b - Empty interrupt if at most 8 sample in Tx left and right FIFOs 11b - Empty interrupt if at most 12 sample in Tx left and right FIFOs
14 —	- Reserved
13 LOW_POWER	LOW_POWER When write 1 to this bit, it will cause SPDIF enter low-power mode. return 1 when SPDIF in Low-Power mode.
12 soft_reset	soft_reset When write 1 to this bit, it will cause SPDIF software reset. The software reset will last 8 cycles. When in the reset process, return 1 when read. else return 0 when read.
11-10 TxFIFO_Ctrl	TxFIFO_Ctrl 00b - Send out digital zero on SPDIF Tx 01b - Tx Normal operation 10b - Reset to 1 sample remaining 11b - Reserved
9 DMA_Rx_En	DMA_Rx_En DMA Receive Request Enable (RX FIFO full)
8 DMA_TX_En	DMA_TX_En DMA Transmit Request Enable (Tx FIFO empty)
7-6 InputSrcSel	InputSrcSel Input Source Selector 00b - SPDIF_IN 01-11b - None
5 ValCtrl	ValCtrl 0b - Outgoing Validity always set 1b - Outgoing Validity always clear
4-2 TxSel	TxSel 000b - Off and output 0 001b - Feed-through SPDIFIN 101b - Tx Normal operation
1-0 USrc_Sel	USrc_Sel U channel source select 00b - No embedded U channel 01b - U channel from SPDIF receive block (CD mode) 10b - Reserved 11b - U channel from on chip transmitter

40.6.1.3 CDText Control Register (SRCD)

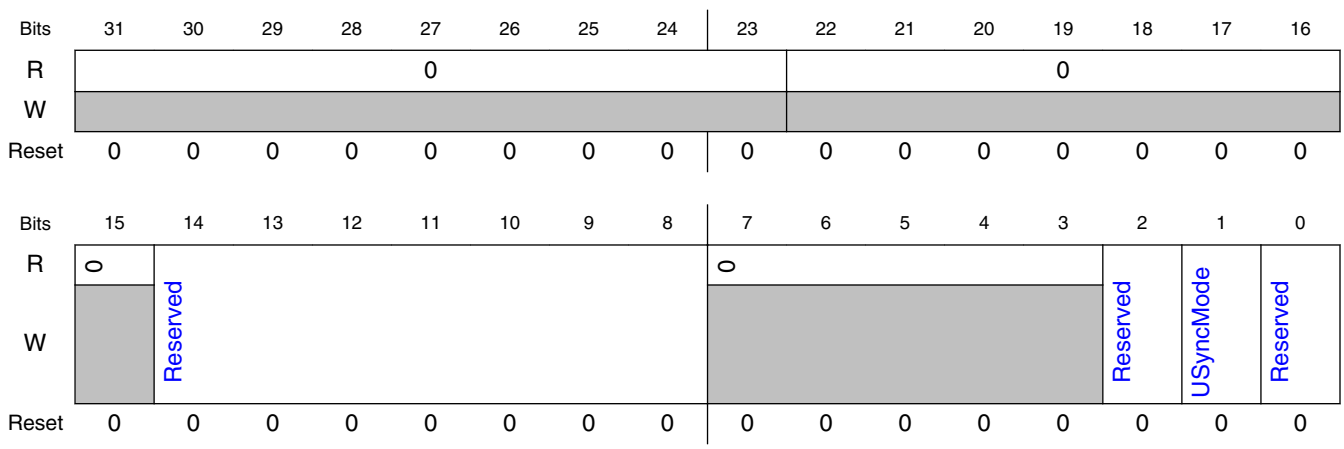
40.6.1.3.1 Offset

Register	Offset
SRCD	4h

40.6.1.3.2 Function

Contains CD text controls

40.6.1.3.3 Diagram



40.6.1.3.4 Fields

Field	Function
31-23 —	This is a 24-bit register the upper byte is unimplemented.
22-15 —	- Return zeros when read
14-8 —	- Reserved. Set to zero.
7-3 —	- Return zeros when read
2 —	- Reserved
1	USyncMode

Table continues on the next page...

Field	Function
USyncMode	0b - Non-CD data 1b - CD user channel subcode
0 —	- Reserved

40.6.1.4 PhaseConfig Register (SRPC)

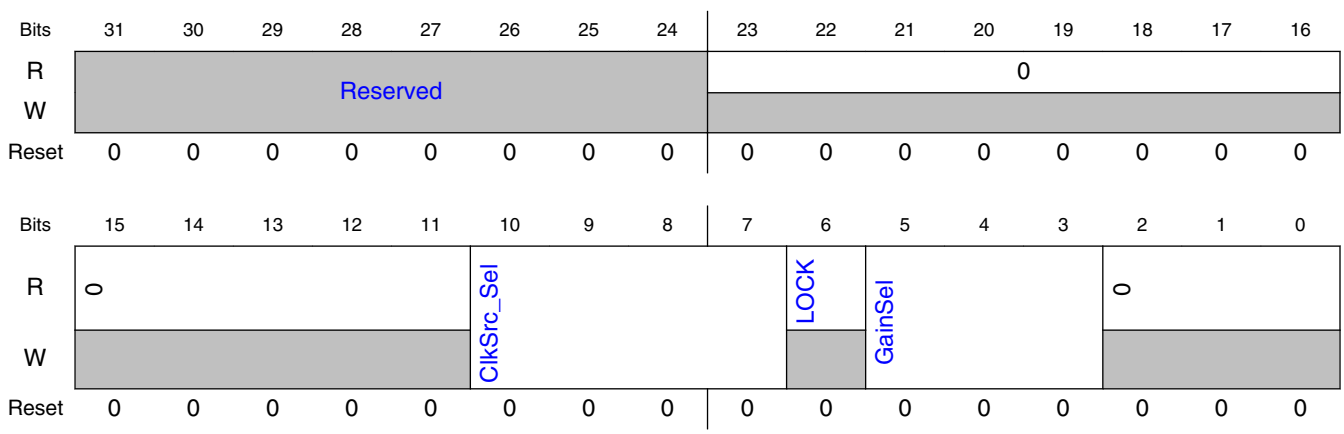
40.6.1.4.1 Offset

Register	Offset
SRPC	8h

40.6.1.4.2 Function

This register contains clocking and DPLL configuration and status details.

40.6.1.4.3 Diagram



40.6.1.4.4 Fields

Field	Function
31-24 —	This is a 24-bit register the upper byte is unimplemented.
23-11	-

Table continues on the next page...

Field	Function
—	Reserved, return zeros when read
10-7 ClkSrc_Sel	ClkSrc_Sel Clock source selection, all other settings not shown are reserved: 0000b - if (DPLL Locked) SPDIF_RxClk else REF_CLK_32K (XTALOSC) 0001b - if (DPLL Locked) SPDIF_RxClk else tx_clk (SPDIF0_CLK_ROOT) 0011b - if (DPLL Locked) SPDIF_RxClk else SPDIF_EXT_CLK 0101b - REF_CLK_32K (XTALOSC) 0110b - tx_clk (SPDIF0_CLK_ROOT) 1000b - SPDIF_EXT_CLK
6 LOCK	LOCK LOCK bit to show that the internal DPLL is locked, read only
5-3 GainSel	GainSel Gain selection: 000b - 24*(2**10) 001b - 16*(2**10) 010b - 12*(2**10) 011b - 8*(2**10) 100b - 6*(2**10) 101b - 4*(2**10) 110b - 3*(2**10)
2-0 —	- Reserved, return zeros when read

40.6.1.5 InterruptEn Register (SIE)

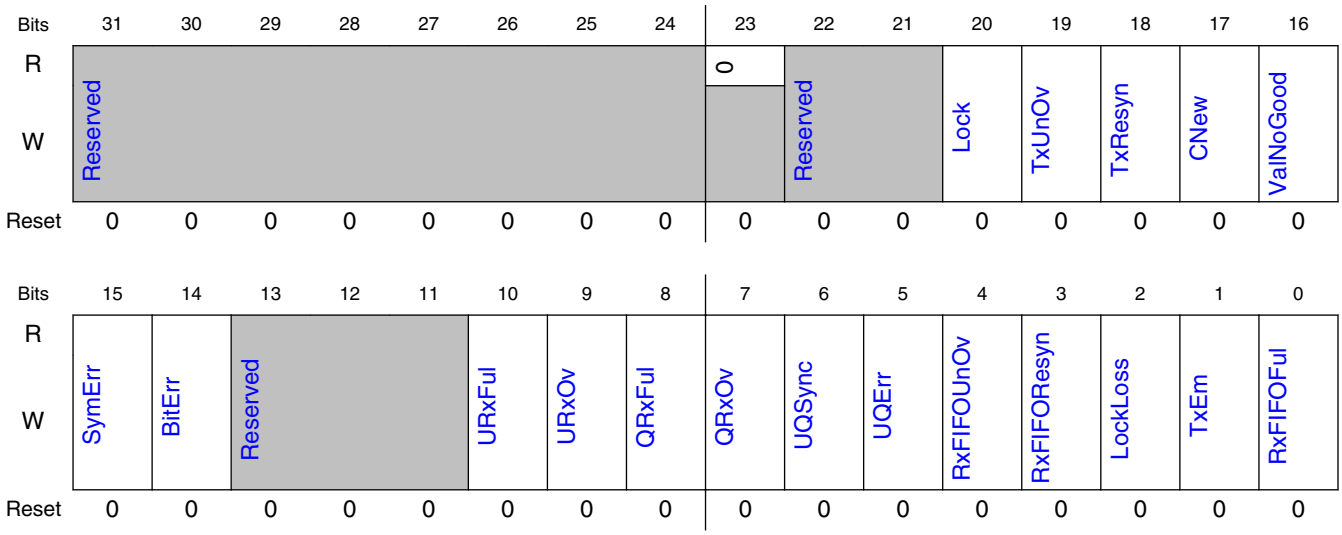
40.6.1.5.1 Offset

Register	Offset
SIE	Ch

40.6.1.5.2 Function

The InterruptEn register (SPDIF_SIE) provides control over the enabling of interrupts.

40.6.1.5.3 Diagram



40.6.1.5.4 Fields

Field	Function
31-24 —	This is a 24-bit register the upper byte is unimplemented.
23 —	- Reserved
22-21 —	- Reserved. Set to zero.
20 Lock	Lock SPDIF receiver's DPLL is locked
19 TxUnOv	TxUnOv SPDIF Tx FIFO under/overrun
18 TxResyn	TxResyn SPDIF Tx FIFO resync
17 CNew	CNew SPDIF receive change in value of control channel
16 ValNoGood	ValNoGood SPDIF validity flag no good
15 SymErr	SymErr SPDIF receiver found illegal symbol
14 BitErr	BitErr SPDIF receiver found parity bit error
13-11	-

Table continues on the next page...

Field	Function
—	Reserved. Set to zero.
10 URxFul	URxFul U Channel receive register full, can't be cleared with reg. IntClear. To clear it, read from U Rx reg.
9 URxOv	URxOv U Channel receive register overrun
8 QRxFul	QRxFul Q Channel receive register full, can't be cleared with reg. IntClear. To clear it, read from Q Rx reg.
7 QRxOv	QRxOv Q Channel receive register overrun
6 UQSync	UQSync U/Q Channel sync found
5 UQErr	UQErr U/Q Channel framing error
4 RxFIFOUnOv	RxFIFOUnOv Rx FIFO underrun/overrun
3 RxFIFOResyn	RxFIFOResyn Rx FIFO resync
2 LockLoss	LockLoss SPDIF receiver loss of lock
1 TxEm	TxEm SPDIF Tx FIFO empty, can't be cleared with reg. IntClear. To clear it, write toTx FIFO.
0 RxFIFOFul	RxFIFOFul SPDIF Rx FIFO full, can't be cleared with reg. IntClear. To clear it, read from Rx FIFO.

40.6.1.6 InterruptClear Register (SIC)

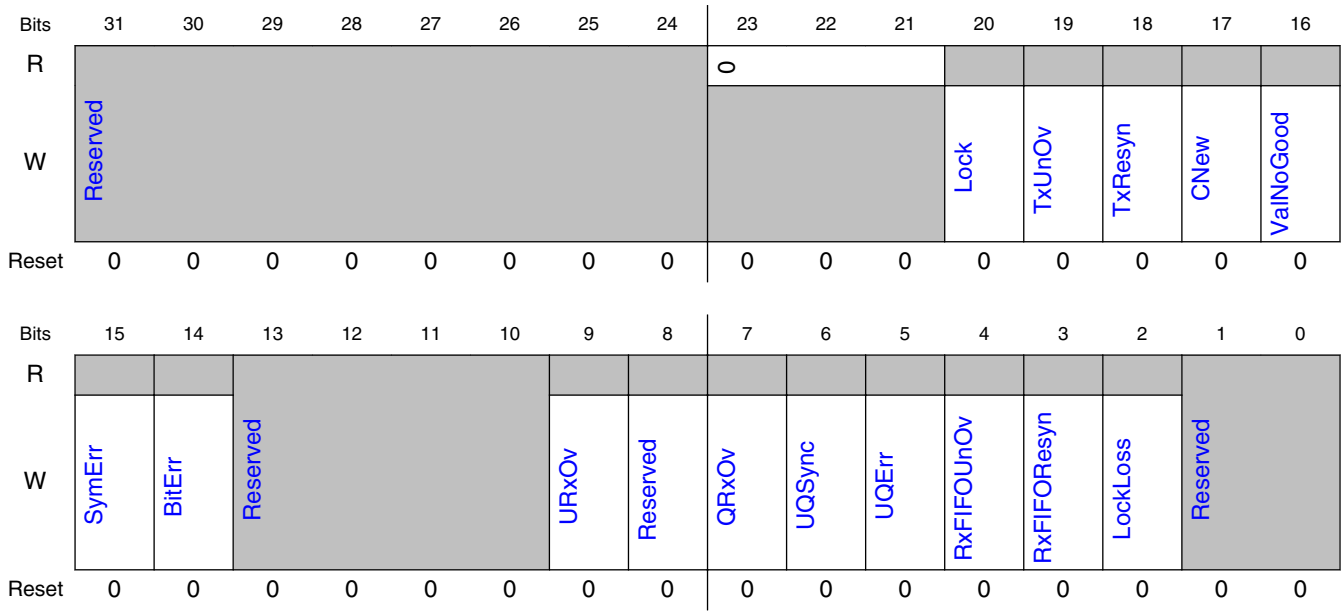
40.6.1.6.1 Offset

Register	Offset
SIC	10h

40.6.1.6.2 Function

The InterruptClear (SPDIF_SIC) register is a write only register and is used to clear interrupts.

40.6.1.6.3 Diagram



40.6.1.6.4 Fields

Field	Function
31-24 —	This is a 24-bit register the upper byte is unimplemented.
23-21 —	- Reserved, for InterruptStat/Clear return zeros when read.
20 Lock	Lock SPDIF receiver's DPLL is locked
19 TxUnOv	TxUnOv SPDIF Tx FIFO under/overrun
18 TxResyn	TxResyn SPDIF Tx FIFO resync
17 CNew	CNew SPDIF receive change in value of control channel
16 ValNoGood	ValNoGood SPDIF validity flag no good
15 SymErr	SymErr SPDIF receiver found illegal symbol
14 BitErr	BitErr SPDIF receiver found parity bit error
13-10	-

Table continues on the next page...

Field	Function
—	Reserved
9 URxOv	URxOv U Channel receive register overrun
8 —	- Reserved
7 QRxOv	QRxOv Q Channel receive register overrun
6 UQSync	UQSync U/Q Channel sync found
5 UQErr	UQErr U/Q Channel framing error
4 RxFIFOUnOv	RxFIFOUnOv Rx FIFO underrun/overrun
3 RxFIFOResyn	RxFIFOResyn Rx FIFO resync
2 LockLoss	LockLoss SPDIF receiver loss of lock
1-0 —	- Reserved

40.6.1.7 InterruptStat Register (SIS)

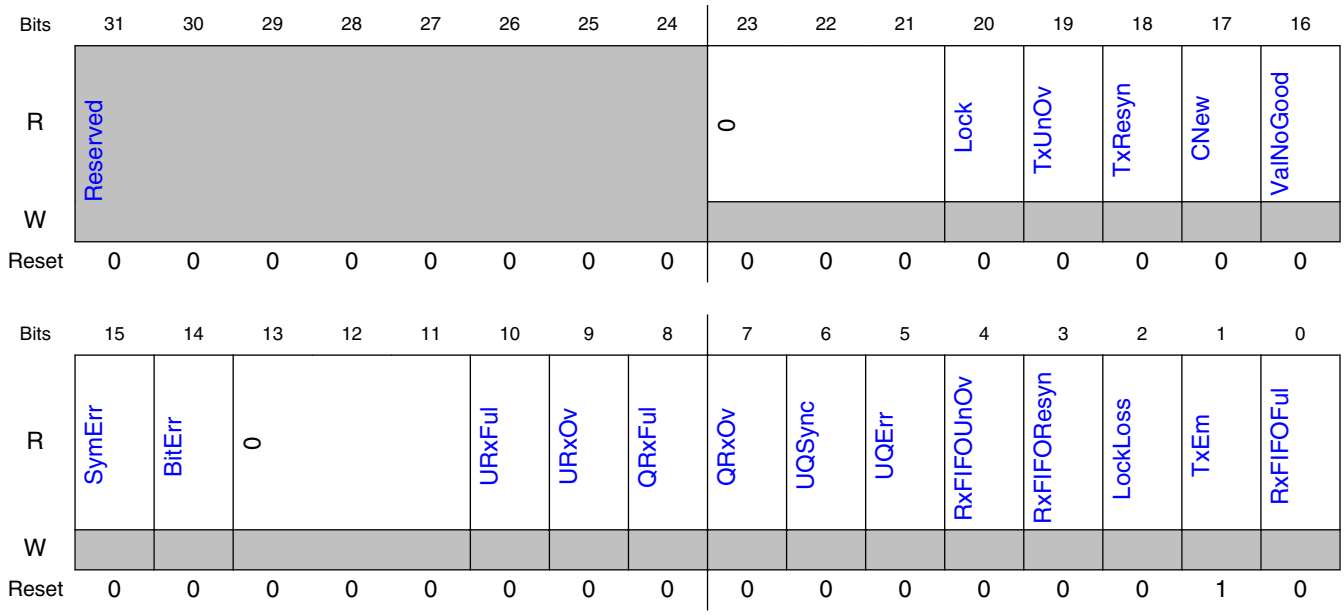
40.6.1.7.1 Offset

Register	Offset
SIS	10h

40.6.1.7.2 Function

The InterruptStat (SPDIF_SIS) register is a read only register that provides the status on interrupt operations.

40.6.1.7.3 Diagram



40.6.1.7.4 Fields

Field	Function
31-24 —	This is a 24-bit register the upper byte is unimplemented.
23-21 —	- Reserved, for InterruptStat/Clear return zeros when read.
20 Lock	Lock SPDIF receiver's DPLL is locked
19 TxUnOv	TxUnOv SPDIF Tx FIFO under/overrun
18 TxResyn	TxResyn SPDIF Tx FIFO resync
17 CNew	CNew SPDIF receive change in value of control channel
16 ValNoGood	ValNoGood SPDIF validity flag no good
15 SymErr	SymErr SPDIF receiver found illegal symbol
14 BitErr	BitErr SPDIF receiver found parity bit error
13-11	-

Table continues on the next page...

Field	Function
—	Reserved. Return zeros when read
10 URxFul	URxFul U Channel receive register full, can't be cleared with reg. IntClear. To clear it, read from U Rx reg.
9 URxOv	URxOv U Channel receive register overrun
8 QRxFul	QRxFul Q Channel receive register full, can't be cleared with reg. IntClear. To clear it, read from Q Rx reg.
7 QRxOv	QRxOv Q Channel receive register overrun
6 UQSync	UQSync U/Q Channel sync found
5 UQErr	UQErr U/Q Channel framing error
4 RxFIFOUnOv	RxFIFOUnOv Rx FIFO underrun/overrun
3 RxFIFOResyn	RxFIFOResyn Rx FIFO resync
2 LockLoss	LockLoss SPDIF receiver loss of lock
1 TxEm	TxEm SPDIF Tx FIFO empty, can't be cleared with reg. IntClear. To clear it, write toTx FIFO.
0 RxFIFOFul	RxFIFOFul SPDIF Rx FIFO full, can't be cleared with reg. IntClear. To clear it, read from Rx FIFO.

40.6.1.8 SPDIFRxLeft Register (SRL)

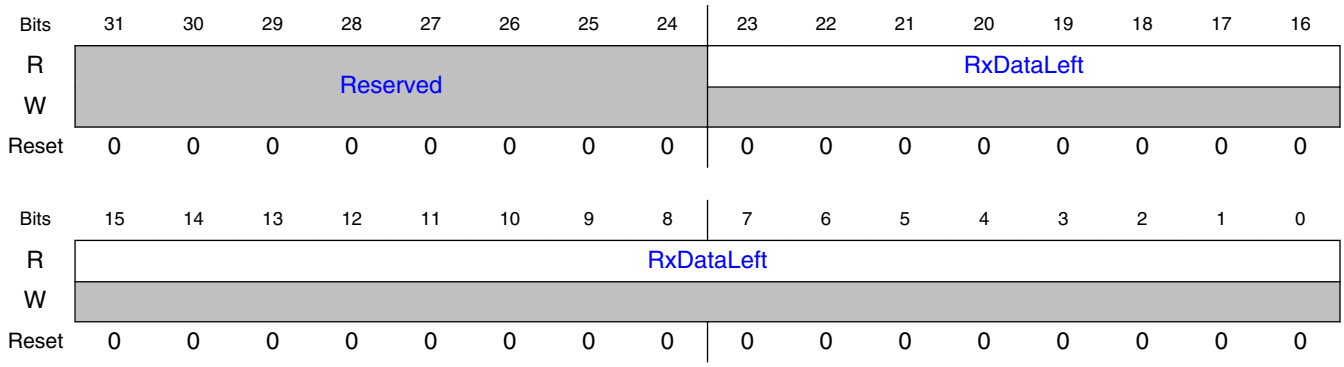
40.6.1.8.1 Offset

Register	Offset
SRL	14h

40.6.1.8.2 Function

SPDIFRxLeft register is an audio data reception register.

40.6.1.8.3 Diagram



40.6.1.8.4 Fields

Field	Function
31-24 —	This is a 24-bit register the upper byte is unimplemented.
23-0 RxDataLeft	RxDataLeft Processor receive SPDIF data left

40.6.1.9 SPDIFRxRight Register (SRR)

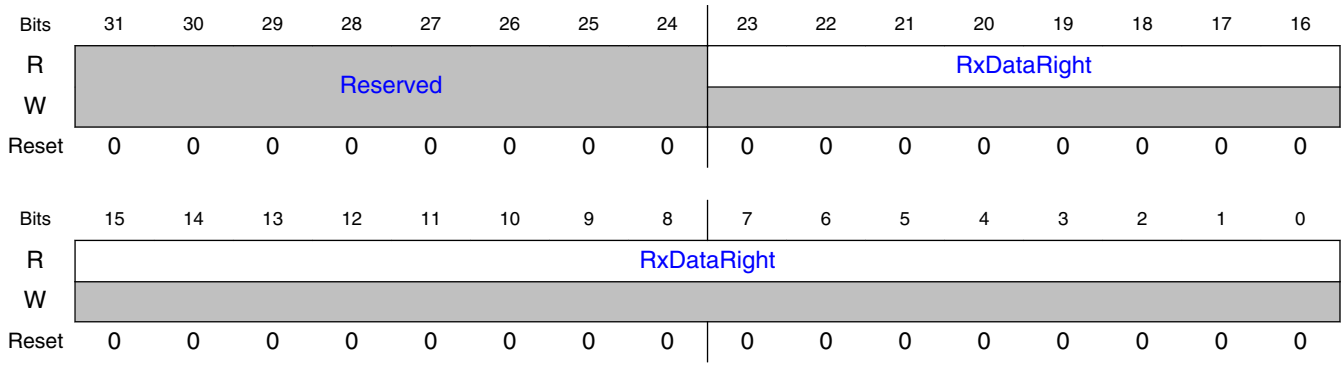
40.6.1.9.1 Offset

Register	Offset
SRR	18h

40.6.1.9.2 Function

SPDIFRxRight register is an audio data reception register.

40.6.1.9.3 Diagram



40.6.1.9.4 Fields

Field	Function
31-24 —	This is a 24-bit register the upper byte is unimplemented.
23-0 RxDataRight	RxDataRight Processor receive SPDIF data right

40.6.1.10 SPDIFRxCChannel_h Register (SRCSH)

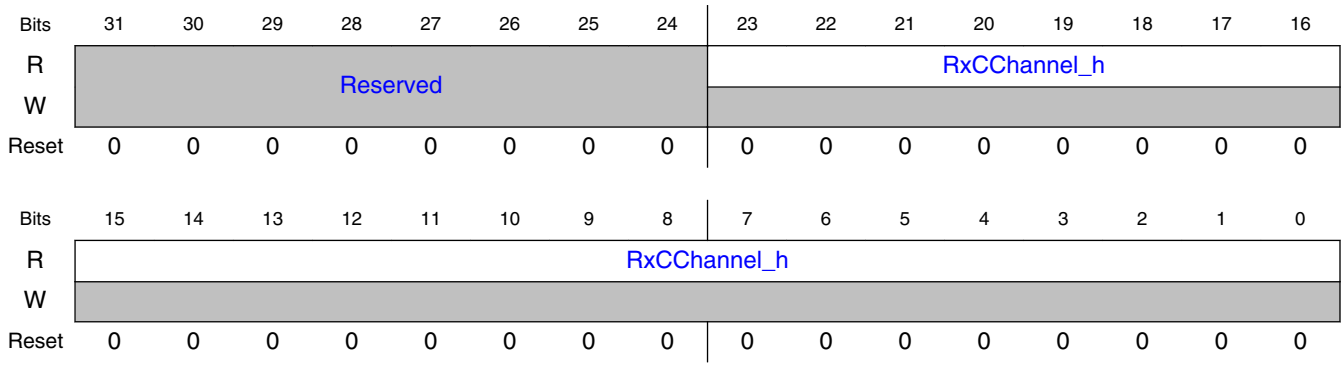
40.6.1.10.1 Offset

Register	Offset
SRCSH	1Ch

40.6.1.10.2 Function

SPDIFRxCChannel_h register is a channel status reception register.

40.6.1.10.3 Diagram



40.6.1.10.4 Fields

Field	Function
31-24 —	This is a 24-bit register the upper byte is unimplemented.
23-0 RxCChannel_h	RxCChannel_h SPDIF receive C channel register, contains first 24 bits of C channel without interpretation

40.6.1.11 SPDIFRxCChannel_l Register (SRCSL)

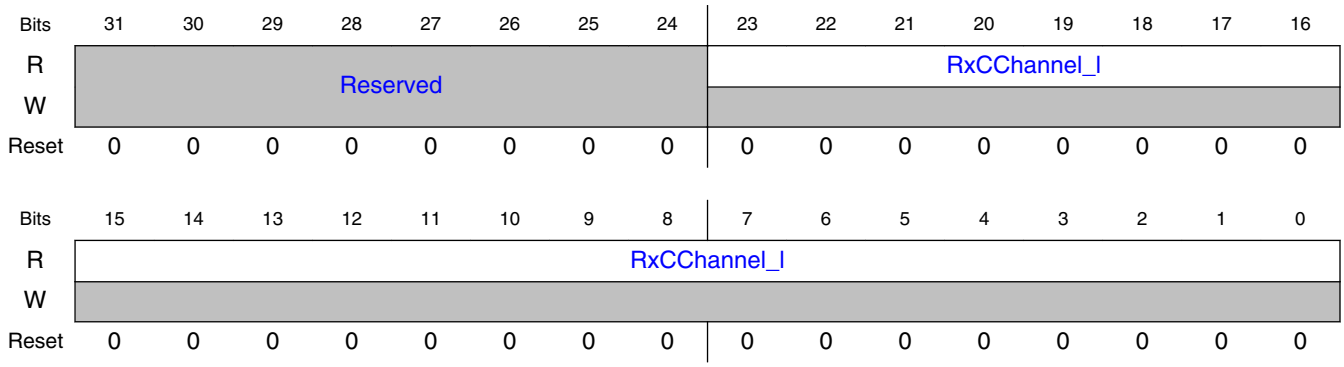
40.6.1.11.1 Offset

Register	Offset
SRCSL	20h

40.6.1.11.2 Function

SPDIFRxCChannel_l register is a channel status reception register.

40.6.1.11.3 Diagram



40.6.1.11.4 Fields

Field	Function
31-24 —	This is a 24-bit register the upper byte is unimplemented.
23-0 RxCChannel_l	RxCChannel_l SPDIF receive C channel register, contains next 24 bits of C channel without interpretation

40.6.1.12 UchannelRx Register (SRU)

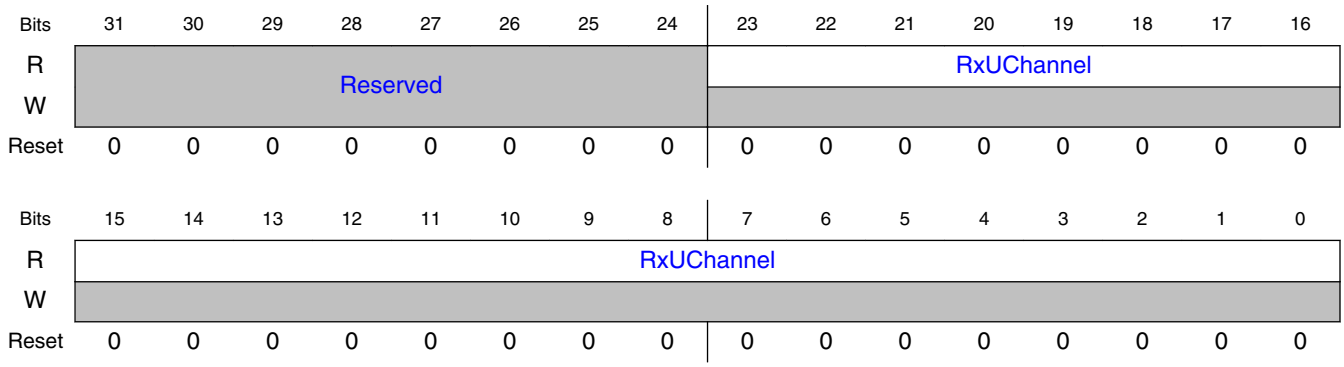
40.6.1.12.1 Offset

Register	Offset
SRU	24h

40.6.1.12.2 Function

UChannelRx register is a user bits reception register.

40.6.1.12.3 Diagram



40.6.1.12.4 Fields

Field	Function
31-24 —	[unimplemented] This is a 24-bit register the upper byte is unimplemented.
23-0 RxUChannel	RxUChannel SPDIF receive U channel register, contains next 3 U channel bytes

40.6.1.13 QchannelRx Register (SRQ)

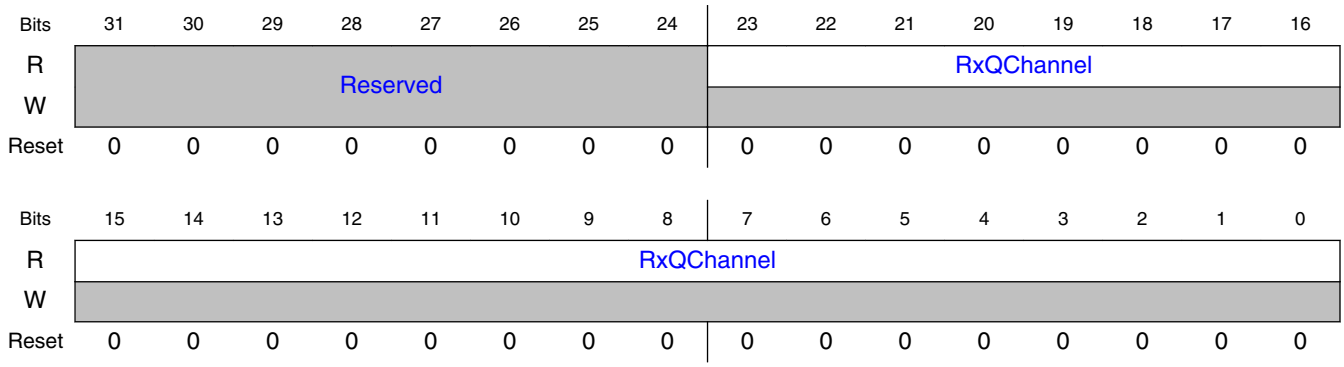
40.6.1.13.1 Offset

Register	Offset
SRQ	28h

40.6.1.13.2 Function

QChannelRx register is a user bits reception register.

40.6.1.13.3 Diagram



40.6.1.13.4 Fields

Field	Function
31-24 —	This is a 24-bit register the upper byte is unimplemented.
23-0 RxQChannel	RxQChannel SPDIF receive Q channel register, contains next 3 Q channel bytes

40.6.1.14 SPDIFTxLeft Register (STL)

40.6.1.14.1 Offset

Register	Offset
STL	2Ch

40.6.1.14.2 Function

SPDIFTxLeft register is an audio data transmission register.

40.6.1.14.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W									TxDataLeft							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	TxDataLeft															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

40.6.1.14.4 Fields

Field	Function
31-24 —	[unimplemented] This is a 24-bit register the upper byte is unimplemented.
23-0 TxDataLeft	TxDataLeft SPDIF transmit left channel data. It is write-only, and always returns zeros when read

40.6.1.15 SPDIFTxRight Register (STR)

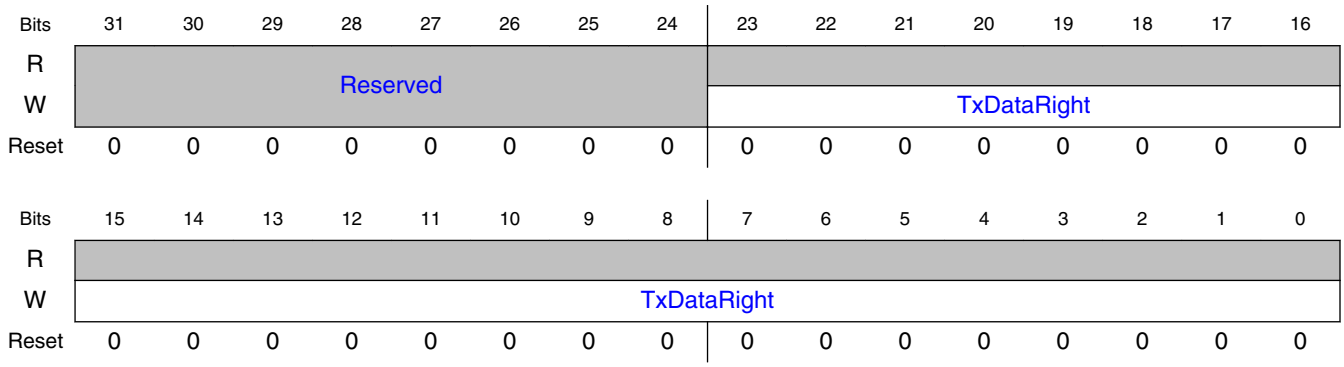
40.6.1.15.1 Offset

Register	Offset
STR	30h

40.6.1.15.2 Function

SPDIFTxRight register is an audio data transmission register.

40.6.1.15.3 Diagram



40.6.1.15.4 Fields

Field	Function
31-24 —	[unimplemented] This is a 24-bit register the upper byte is unimplemented.
23-0 TxDataRight	TxDataRight SPDIF transmit right channel data. It is write-only, and always returns zeros when read

40.6.1.16 SPDIFTxCChannelCons_h Register (STCSCH)

40.6.1.16.1 Offset

Register	Offset
STCSCH	34h

40.6.1.16.2 Function

SPDIFTxCChannelCons_h register is a channel status transmission register.

40.6.1.16.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved								TxCChannelCons_h							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TxCChannelCons_h															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

40.6.1.16.4 Fields

Field	Function
31-24 —	This is a 24-bit register the upper byte is unimplemented.
23-0 TxCChannelCons_h	TxCChannelCons_h SPDIF transmit Cons. C channel data, contains first 24 bits without interpretation. When read, it returns the latest data written by the processor

40.6.1.17 SPDIFTxCChannelCons_l Register (STCSCL)

40.6.1.17.1 Offset

Register	Offset
STCSCL	38h

40.6.1.17.2 Function

SPDIFTxCChannelCons_l register is a channel status transmission register.

40.6.1.17.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved								TxCChannelCons_l							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TxCChannelCons_l															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

40.6.1.17.4 Fields

Field	Function
31-24 —	This is a 24-bit register the upper byte is unimplemented.
23-0 TxCChannelCons_l	TxCChannelCons_l SPDIF transmit Cons. C channel data, contains next 24 bits without interpretation. When read, it returns the latest data written by the processor

40.6.1.18 FreqMeas Register (SRFM)

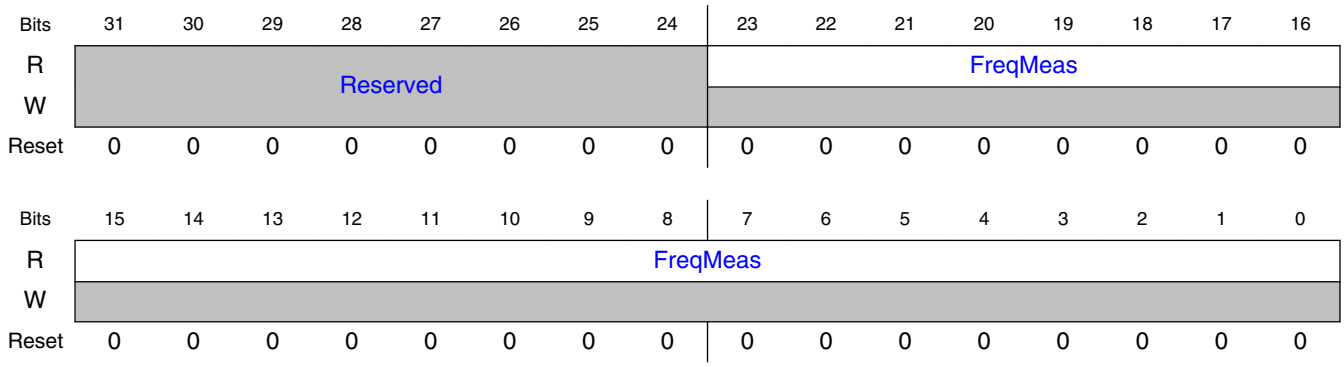
40.6.1.18.1 Offset

Register	Offset
SRFM	44h

40.6.1.18.2 Function

Contains Frequency measurement data.

40.6.1.18.3 Diagram



40.6.1.18.4 Fields

Field	Function
31-24 —	This is a 24-bit register the upper byte is unimplemented.
23-0 FreqMeas	FreqMeas Frequency measurement data

40.6.1.19 SPDIFTxClk Register (STC)

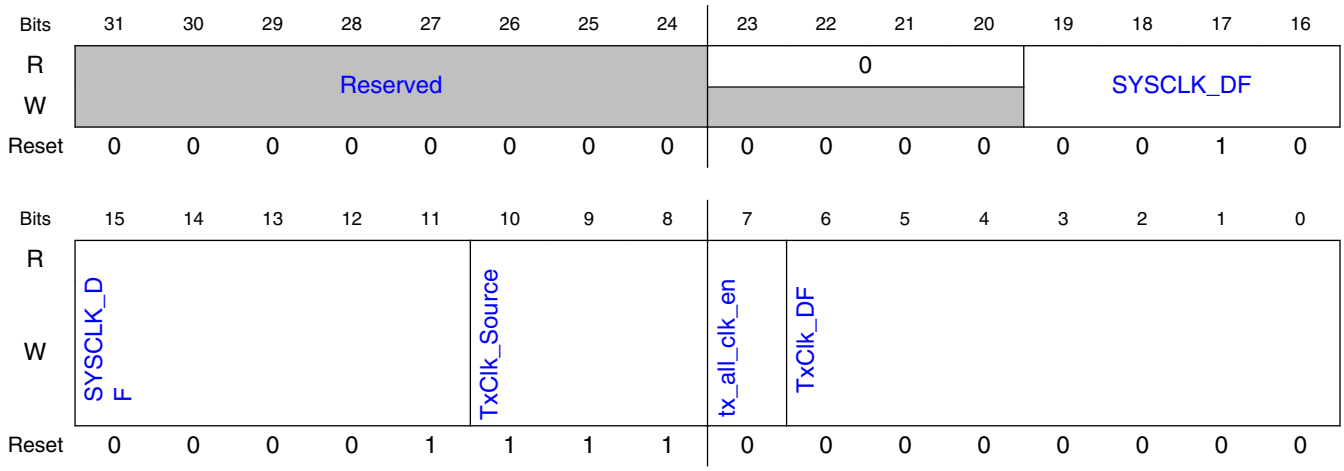
40.6.1.19.1 Offset

Register	Offset
STC	50h

40.6.1.19.2 Function

The SPDIFTxClk Control register includes the means to select the transmit clock and frequency division.

40.6.1.19.3 Diagram



40.6.1.19.4 Fields

Field	Function
31-24 —	This is a 24-bit register the upper byte is unimplemented.
23-20 —	- Reserved, return zeros when read
19-11 SYSCLK_DF	SYSCLK_DF system clock divider factor, 2~512. 000000000b - no clock signal 000000001b - divider factor is 2 111111111b - divider factor is 512
10-8 TxClk_Source	TxClk_Source 000b - REF_CLK_32K input (XTALOSC 32 kHz clock) 001b - tx_clk input (from SPDIF0_CLK_ROOT. See clock control block for more information.) 011b - SPDIF_EXT_CLK, from pads 101b - ipg_clk input (frequency divided)
7 tx_all_clk_en	tx_all_clk_en Spdif transfer clock enable.When data is going to be transfered, this bit should be set to1. 0b - disable transfer clock. 1b - enable transfer clock.
6-0 TxClk_DF	TxClk_DF Divider factor (1-128) 0000000b - divider factor is 1 0000001b - divider factor is 2 1111111b - divider factor is 128

Chapter 41
10/100-Mbps Ethernet MAC (ENET)

41.1 Chip-specific ENET information

Table 41-1. Reference links to related information

Topic	Related module(s)	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
Interrupts, DMA Events and XBAR Assignments	-	Interrupts, DMA Events and XBAR Assignments

41.2 Overview

The core implements a dual-speed 10/100-Mbit/s Ethernet MAC compliant with the IEEE802.3-2002 standard. The MAC layer provides compatibility with half- or full-duplex 10/100-Mbit/s Ethernet LANs.

The MAC operation is fully programmable and can be used in Network Interface Card (NIC), bridging, or switching applications. The core implements the remote network monitoring (RMON) counters according to IETF RFC 2819.

The core also implements a hardware acceleration block to optimize the performance of network controllers providing TCP/IP, UDP, and ICMP protocol services. The acceleration block performs critical functions in hardware, which are typically implemented with large software overhead.

The core implements programmable embedded FIFOs that can provide buffering on the receive path for lossless flow control.

Advanced power management features are available with magic packet detection and programmable power-down modes.

A unified DMA (uDMA), internal to the ENET module, optimizes data transfer between the ENET core and the SoC, and supports an enhanced buffer descriptor programming model to support IEEE 1588 functionality.

The programmable Ethernet MAC with IEEE 1588 integrates a standard IEEE 802.3 Ethernet MAC with a time-stamping module. The IEEE 1588 standard provides accurate clock synchronization for distributed control nodes for industrial automation applications.

41.2.1 Block diagram

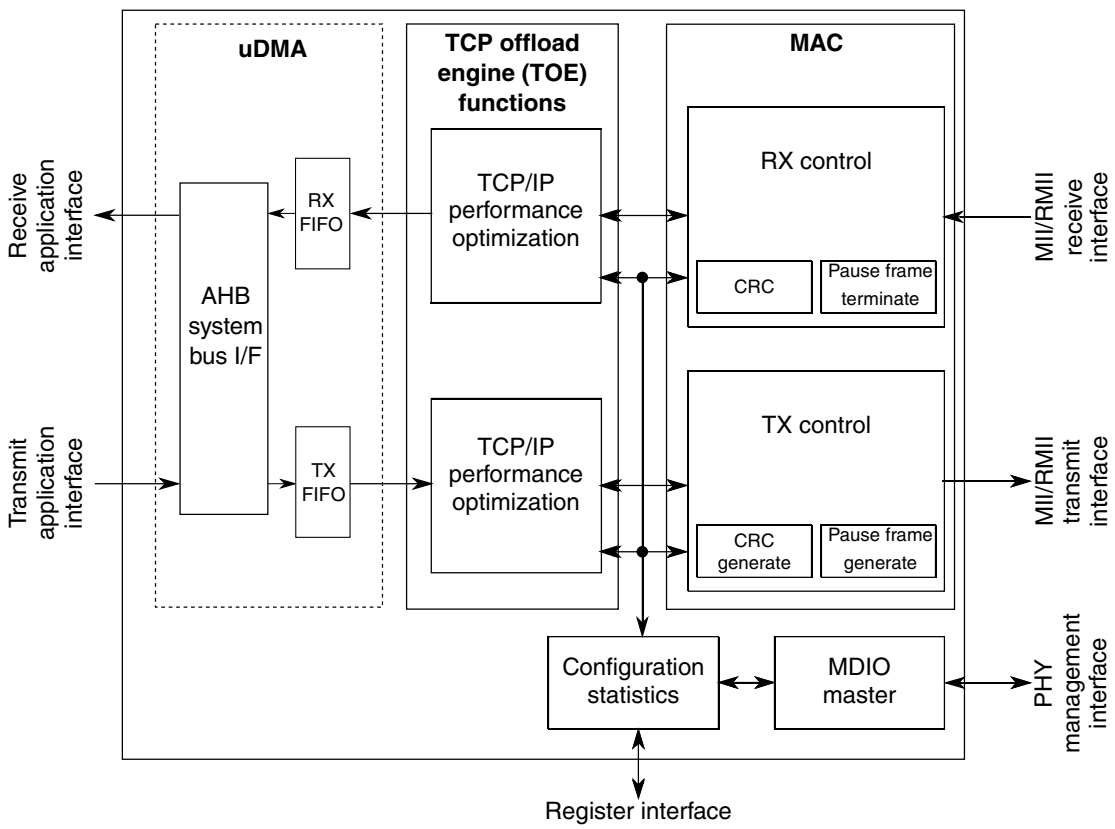


Figure 41-1. Ethernet MAC-NET core block diagram

41.2.2 Features

The MAC-NET core includes the following features.

41.2.2.1 Ethernet MAC features

- Implements the full 802.3 specification with preamble/SFD generation, frame padding generation, CRC generation and checking
- Supports zero-length preamble
- Dynamically configurable to support 10/100-Mbit/s operation
- Supports 10/100 Mbit/s full-duplex and configurable half-duplex operation
- Compliant with the AMD magic packet detection with interrupt for node remote power management
- Seamless interface to commercial ethernet PHY devices via one of the following:
  - a 4-bit Media Independent Interface (MII) operating at 2.5/25 MHz.
  - a 4-bit non-standard MII-Lite (MII without the CRS and COL signals) operating at 2.5/25 MHz.
  - a 2-bit Reduced MII (RMII) operating at 50 MHz.
- CRC-32 checking at full speed with optional forwarding of the frame check sequence (FCS) field to the client
- CRC-32 generation and append on transmit or forwarding of user application provided FCS selectable on a per-frame basis
- In full-duplex mode:
  - Supports Ethernet pause frame (802.3 Annex 31A) providing fully automated flow control without any user application overhead
  - Pause quanta used to form pause frames, dynamically programmable
  - Pause frame generation additionally controllable by user application offering flexible traffic flow control
  - Optional forwarding of received pause frames to the user application
  - Implements standard flow-control mechanism
- In half-duplex mode: provides full collision support, including jamming, backoff, and automatic retransmission
- Supports VLAN-tagged frames according to IEEE 802.1Q
- Programmable MAC address: Insertion on transmit; discards frames with mismatching destination address on receive (except broadcast and pause frames)
- Programmable promiscuous mode support to omit MAC destination address checking on receive
- Multicast and unicast address filtering on receive based on 64-entry hash table, reducing higher layer processing load
- Programmable frame maximum length providing support for any standard or proprietary frame length
- Statistics indicators for frame traffic and errors (alignment, CRC, length) and pause frames providing for IEEE 802.3 basic and mandatory management information database (MIB) package and remote network monitoring (RFC 2819)

- Simple handshake user application FIFO interface with fully programmable depth and threshold levels
- Provides separate status word for each received frame on the user interface providing information such as frame length, frame type, VLAN tag, and error information
- Multiple internal loopback options
- Programmable Clause 22 and Clause 45 MDIO Master interface for PHY device configuration and management
- Supports legacy FEC buffer descriptors
- Supports interupt coalescing

41.2.2.2 IP protocol performance optimization features

- Operates on TCP/IP and UDP/IP and ICMP/IP protocol data or IP header only
- Enables wire-speed processing
- Supports IPv4 and IPv6
- Transparent passing of frames of other types and protocols
- Supports VLAN tagged frames according to IEEE 802.1q with transparent forwarding of VLAN tag and control field
- Automatic IP-header and payload (protocol specific) checksum calculation and verification on receive
- Automatic IP-header and payload (protocol specific) checksum generation and insertion on transmit. Configurable on a per-frame basis
- Supports IP and TCP, UDP, ICMP data for checksum generation and checking
- Supports full header options for IPv4 and TCP protocol headers
- Provides IPv6 support to datagrams with base header only — datagrams with extension headers are passed transparently unmodifed/unchecked
- Provides statistics information for received IP and protocol errors
- Configurable automatic discard of erroneous frames
- Configurable automatic host-to-network (RX) and network-to-host (TX) byte order conversion for IP and TCP/UDP/ICMP headers within the frame
- Configurable padding remove for short IP datagrams on receive

- Configurable Ethernet payload alignment to allow for 32-bit word-aligned header and payload processing
- Programmable store-and-forward operation with clock and rate decoupling FIFOs

41.2.2.3 IEEE 1588 features

- Supports all IEEE 1588 frames.
- Allows reference clock to be chosen independently of network speed.
- Software-programmable precise time-stamping of ingress and egress frames
- Timer monitoring capabilities for system calibration and timing accuracy management
- Precise time-stamping of external events with programmable interrupt generation
- Programmable event and interrupt generation for external system control
- Supports hardware- and software-controllable timer synchronization.
- Provides a 4-channel IEEE 1588 timer. Each channel supports input capture and output compare using the 1588 counter.

41.3 Functional description

This section provides a complete functional description of the MAC-NET core.

41.3.1 Ethernet MAC frame formats

- Minimum length of 64 bytes
- Maximum length of 1518 bytes excluding the preamble and the start frame delimiter (SFD) bytes

An Ethernet frame consists of the following fields:

- Seven bytes preamble
- Start frame delimiter (SFD)
- Two address fields

- Length or type field
- Data field
- Frame check sequence (CRC value)

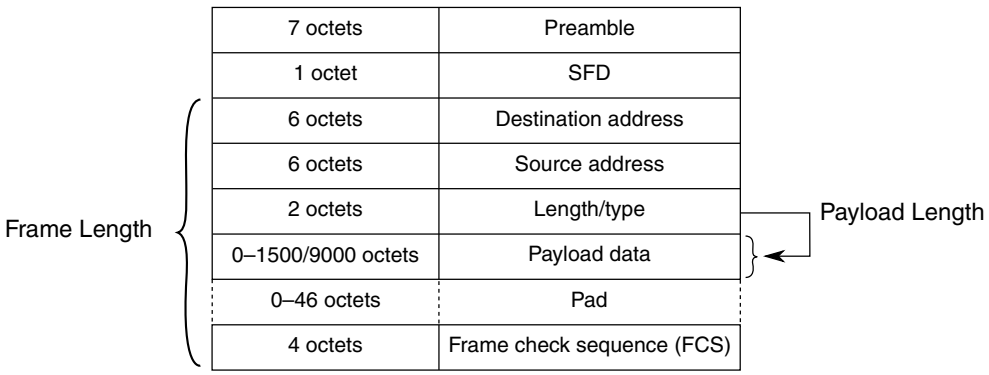


Figure 41-2. MAC frame format overview

Optionally, MAC frames can be VLAN-tagged with an additional four-byte field inserted between the MAC source address and the type/length field. VLAN tagging is defined by the IEEE P802.1q specification. VLAN-tagged frames have a maximum length of 1522 bytes, excluding the preamble and the SFD bytes.

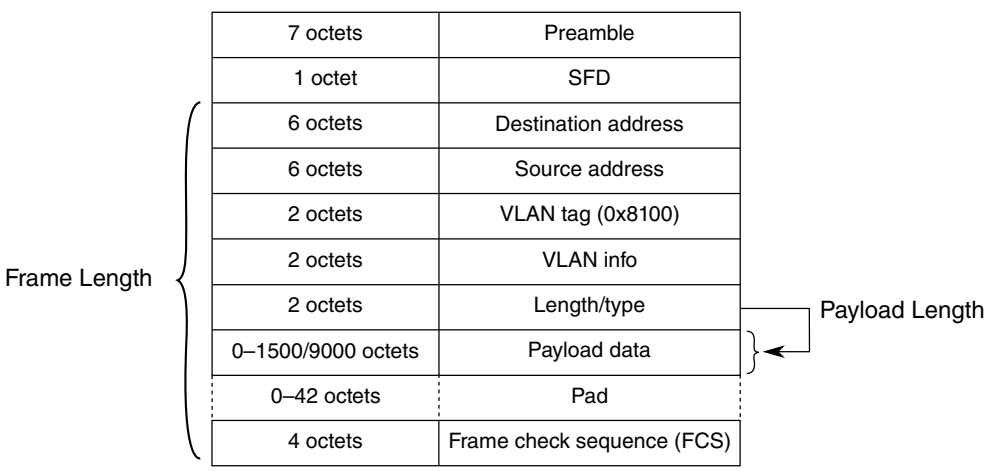


Figure 41-3. VLAN-tagged MAC frame format overview

Table 41-2. MAC frame definition

Term	Description
Frame length	Defines the length, in octets, of the complete frame without preamble and SFD. A frame has a valid length if it contains at least 64 octets and does not exceed the programmed maximum length.
Payload length	The length/type field indicates the length of the frame's payload section. The most significant byte is sent/received first.

Table continues on the next page...

Table 41-2. MAC frame definition (continued)

Term	Description
	• If the length/type field is set to a value less than 46, the payload is padded so that the minimum frame length requirement (64 bytes) is met. For VLAN-tagged frames, a value less than 42 indicates a padded frame. • If the length/type field is set to a value larger than the programmed frame maximum length (e.g. 1518) it is interpreted as a type field.
Destination and source address	48-bit MAC addresses. The least significant byte is sent/received first and the first two least significant bits of the MAC address distinguish MAC frames, as detailed in MAC address check.

Note

Although the IEEE specification defines a maximum frame length, the MAC core provides the flexibility to program any value for the frame maximum length.

41.3.1.1 Pause Frames

The receiving device generates a pause frame to indicate a congestion to the emitting device, which should stop sending data.

Pause frames are indicated by the length/type set to 0x8808. The two first bytes of a pause frame following the type, defines a 16-bit opcode field set to 0x0001 always. A 16-bit pause quanta is defined in the frame payload bytes 2 (P1) and 3 (P2) as defined in the following table. The P1 pause quanta byte is the most significant.

Table 41-3. Pause Frame Format (Values in Hex)

1	2	3	4	5	6	7	8	9	10	11	12	13	14
55	55	55	55	55	55	55	D5	01	80	C2	00	00	01
Preamble							SFD	Multicast Destination Address					
15	16	17	18	19	20	21	22	23	24	25	26	27 –68	
00	00	00	00	00	00	88	08	00	01	hi	lo	00	
Source Address						Type		Opcode		P1	P2	pad (42)	
69	70	71	72										
26	6B	AE	0A										
CRC-32													

There is no payload length field found within a pause frame and a pause frame is always padded with 42 bytes (0x00).

If a pause frame with a pause value greater than zero (XOFF condition) is received, the MAC stops transmitting data as soon the current frame transfer is completed. The MAC stops transmitting data for the value defined in pause quanta. One pause quanta fraction refers to 512 bit times.

If a pause frame with a pause value of zero (XON condition) is received, the transmitter is allowed to send data immediately (see Full-duplex flow control operation for details).

41.3.1.2 Magic packets

A magic packet is a unicast, multicast, or broadcast packet, which carries a defined sequence in the payload section.

Magic packets are received and inspected only under specific conditions as described in Magic packet detection.

The defined sequence to decode a magic packet is formed with a synchronization stream which consists of six consecutive 0xFF bytes, and is followed by sequence of sixteen consecutive unicast MAC addresses of the node to be awakened.

This sequence can be located anywhere in the magic packet payload. The magic packet is formed with a standard Ethernet header, optional padding, and CRC.

41.3.2 IP and higher layers frame format

The following sections use the term datagram to describe the protocol specific data unit that is found within the payload section of its container entity.

For example, an IP datagram specifies the payload section of an Ethernet frame. A TCP datagram specifies the payload section within an IP datagram.

41.3.2.1 Ethernet types

IP datagrams are carried in the payload section of an Ethernet frame. The Ethernet frame type/length field discriminates several datagram types.

The following table lists the types of interest:

Table 41-4. Ethernet type value examples

Type	Description
0x8100	VLAN-tagged frame. The actual type is found 4 octets later in the frame.

Table continues on the next page...

Table 41-4. Ethernet type value examples (continued)

Type	Description
0x0800	IPv4
0x0806	ARP
0x86DD	IPv6

41.3.2.2 IPv4 datagram format

The following figure shows the IP Version 4 (IPv4) header, which is located at the beginning of an IP datagram. It is organized in 32-bit words. The first byte sent/received is the leftmost byte of the first word (in other words, version/IHL field).

The IP header can contain further options, which are always padded if necessary to guarantee the payload following the header is aligned to a 32-bit boundary.

The IP header is immediately followed by the payload, which can contain further protocol headers (for example, TCP or UDP, as indicated by the protocol field value). The complete IP datagram is transported in the payload section of an Ethernet frame.

Table 41-5. IPv4 header format

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
Version	IHL	TOS		Length			
Fragment ID				Flags	Fragment offset		
TTL		Protocol		Header checksum			
Source address							
Destination address							
Options							

Table 41-6. IPv4 header fields

Field name	Description
Version	4-bit IP version information. 0x4 for IPv4 frames.
IHL	4-bit Internet header length information. Determines number of 32-bit words found within the IP header. If no options are present, the default value is 0x5.
TOS	Type of service/DiffServ field.
Length	Total length of the datagram in bytes, including all octets of header and payload.
Fragment ID, flags, fragment offset	Fields used for IP fragmentation.
TTL	Time-to-live. In effect, is decremented at each router arrival. If zero, datagram must be discarded.
Protocol	Identifier of protocol that follows in the datagram.

Table continues on the next page...

Table 41-6. IPv4 header fields (continued)

Field name	Description
Header checksum	Checksum of IP header. For computational purposes, this field's value is zero.
Source address	Source IP address.
Destination address	Destination IP address.

41.3.2.3 IPv6 datagram format

The following figure shows the IP version 6 (IPv6) header, which is located at the beginning of an IP datagram. It is organized in 32-bit words and has a fixed length of ten words (40 bytes). The next header field identifies the type of the header that follows the IPv6 header. It is defined similar to the protocol identifier within IPv4, with new definitions for identifying extension headers. These headers can be inserted between the IPv6 header and the protocol header, which will shift the protocol header accordingly.The accelerator currently only supports IPv6 without extension headers (in other words, the next header specifies TCP, UDP, or IMCP).

The first byte sent/received is the leftmost byte of the first word (in other words, version/ traffic class fields).

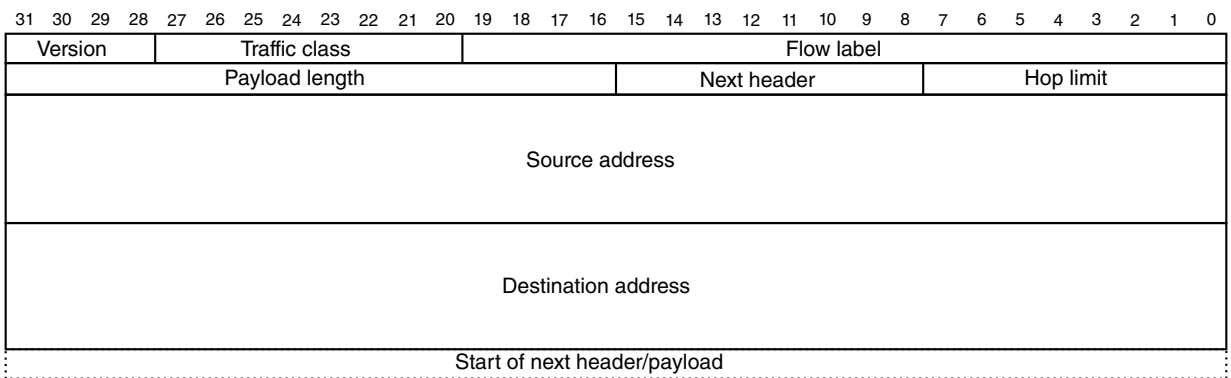


Figure 41-4. IPv6 header format

Table 41-7. IPv6 header fields

Field name	Description
Version	4-bit IP version information. 0x6 for all IPv6 frames.
Traffic class	8-bit field defining the traffic class.
Flow label	20-bit flow label identifying frames of the same flow.
Payload length	16-bit length of the datagram payload in bytes. It includes all octets following the IPv6 header.
Next header	Identifies the header that follows the IPv6 header. This can be the protocol header or any IPv6 defined extension header.

Table continues on the next page...

Table 41-7. IPv6 header fields (continued)

Field name	Description
Hop limit	Hop counter, decremented by one by each station that forwards the frame. If hop limit is 0 the frame must be discarded.
Source address	128-bit IPv6 source address.
Destination address	128-bit IPv6 destination address.

41.3.2.4 Internet Control Message Protocol (ICMP) datagram format

An internet control message protocol (ICMP) is found following the IP header, if the protocol identifier is 1. The ICMP datagram has a four-octet header followed by additional message data.

Table 41-8. ICMP header format

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
Type		Code		Checksum			
ICMP message data							

Table 41-9. IP header fields

Field name	Description
Type	8-bit type information
Code	8-bit code that is related to the message type
Checksum	16-bit one's complement checksum over the complete ICMP datagram

41.3.2.5 User Datagram Protocol (UDP) datagram format

A user datagram protocol header is found after the IP header, when the protocol identifier is 17.

The payload of the datagram is after the UDP header. The header byte order follows the conventions given for the IP header above.

Table 41-10. UDP header format

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
Source port				Destination port			
Length				Checksum			

Table 41-11. UDP header fields

Field name	Description
Source port	Source application port
Destination port	Destination application port
Length	Length of user data which immediately follows the header, including the UDP header (that is, minimum value is 8)
Checksum	Checksum over the complete datagram and some IP header information

41.3.2.6 TCP datagram format

A TCP header is found following the IP header, when the protocol identifier has a value of 6.

The TCP payload immediately follows the TCP header.

Table 41-12. TCP header format

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
Source port				Destination port			
Sequence number							
Acknowledgement number							
Offset	Reserved		Flags	Window			
Checksum				Urgent pointer			
Options							

Table 41-13. TCP header fields

Field name	Description
Source port	Source application port
Destination port	Destination application port
Sequence number	Transmit sequence number
Ack. number	Receive sequence number
Offset	Data offset, which is number of 32-bit words within TCP header — if no options selected, defaults to value of 5
Flags	URG, ACK, PSH, RST, SYN, FIN flags
Window	TCP receive window size information
Checksum	Checksum over the complete datagram (TCP header and data) and IP header information
Options	Additional 32-bit words for protocol options

41.3.3 IEEE 1588 message formats

The following sections describe the IEEE 1588 message formats.

41.3.3.1 Transport encapsulation

The precision time protocol (PTP) datagrams are encapsulated in Ethernet frames using the UDP/IP transport mechanism, or optionally, with the newer 1588v2 directly in Ethernet frames (layer 2).

Typically, multicast addresses are used to allow efficient distribution of the synchronization messages.

41.3.3.1.1 UDP/IP

The 1588 messages (v1 and v2) can be transported using UDP/IP multicast messages.

The table below shows IP multicast groups defined for PTP. The table also shows their respective MAC layer multicast address mapping according to RFC 1112 (last three octets of IP follow the fixed value of 01-00-5E).

Table 41-14. UDP/IP multicast domains

Name	IP Address	MAC Address mapping
DefaultPTPdomain	224.0.1.129	01-00-5E-00-01-81
AlternatePTPdomain1	224.0.1.130	01-00-5E-00-01-82
AlternatePTPdomain2	224.0.1.131	01-00-5E-00-01-83
AlternatePTPdomain3	224.0.1.132	01-00-5E-00-01-84

Table 41-15. UDP port numbers

Message type	UDP port	Note
Event	319	Used for SYNC and DELAY_REQUEST messages
General	320	All other messages (for example, follow-up, delay-response)

41.3.3.1.2 Native Ethernet (PTPv2)

In addition to using UDP/IP frames, IEEE 1588v2 defines a native Ethernet frame format that uses ethertype = 0x88F7. The payload of the Ethernet frame immediately contains the PTP datagram, starting with the PTPv2 header.

Besides others, version 2 adds a peer delay mechanism to allow delay measurements between individual point-to-point links along a path over multiple nodes. The following multicast domains are also defined in PTPv2.

Table 41-16. PTPv2 multicast domains

Name	MAC address
Normal messages	01-1B-19-00-00-00
Peer delay messages	01-80-C2-00-00-0E

41.3.3.2 PTP header

All PTP frames contain a common header that determines the protocol version and the type of message, which defines the remaining content of the message.

All multi-octet fields are transmitted in big-endian order (the most significant byte is transmitted/received first).

The last four bits of versionPTP are at the same position (second byte) for PTPv1 and PTPv2 headers. This allows accurate identification by inspecting the first two bytes of the message.

41.3.3.2.1 PTPv1 header

Table 41-17. Common PTPv1 message header

Offset	Octets	Bits							
		7	6	5	4	3	2	1	0
0	2	versionPTP = 0x0001							
2	2	versionNetwork							
4	16	subdomain							
20	1	messageType							
21	1	sourceCommunicationTechnology							
22	6	sourceUuid							
28	2	sourcePortId							
30	2	sequenceId							
32	1	control							
33	1	0x00							
34	2	flags							
36	4	reserved							

The type of message is encoded in the messageType and control fields as shown in Table 41-18 :

Table 41-18. PTPv1 message type identification

messageType	control	Message Name	Message
0x01	0x0	SYNC	Event message
0x01	0x1	DELAY_REQ	Event message
0x02	0x2	FOLLOW_UP	General message
0x02	0x3	DELAY_RESP	General message
0x02	0x4	MANAGEMENT	General message
other	other	—	Reserved

The field sequenceId is used to non-ambiguously identify a message.

41.3.3.2.2 PTPv2 header

Table 41-19. Common PTPv2 message header

Offset	Octets	Bits							
		7	6	5	4	3	2	1	0
0	1	transportSpecific				messageId			
1	1	reserved				versionPTP = 0x2			
2	2	messageLength							
4	1	domainNumber							
5	1	reserved							
6	2	flags							
8	8	correctionField							
16	4	reserved							
20	10	sourcePortIdentity							
30	2	sequenceId							
32	1	control							
33	1	logMeanMessageInterval							

The type of message is encoded in the field messageId as follows:

Table 41-20. PTPv2 message type identification

messageId	Message name	Message
0x0	SYNC	Event message
0x1	DELAY_REQ	Event message
0x2	PATH_DELAY_REQ	Event message
0x3	PATH_DELAY_RESP	Event message
0x4–0x7	—	Reserved
0x8	FOLLOW_UP	General message
0x9	DELAY_RESP	General message

Table continues on the next page...

Table 41-20. PTPv2 message type identification (continued)

messageId	Message name	Message
0xa	PATH_DELAY_FOLLOW_UP	General message
0xb	ANNOUNCE	General message
0xc	SIGNALING	General message
0xd	MANAGEMENT	General message

The PTPv2 flags field contains further details on the type of message, especially if one-step or two-step implementations are used. The one- or two-step implementation is controlled by the TWO_STEP bit in the first octet of the flags field as shown below. Reserved bits are cleared.

Table 41-21. PTPv2 message flags field definitions

Bit	Name	Description
0	ALTERNATE_MASTER	See IEEE 1588 Clause 17.4
1	TWO_STEP	1 Two-step clock 0 One-step clock
2	UNICAST	1 Transport layer address uses a unicast destination address 0 Multicast is used
3	—	Reserved
4	—	Reserved
5	Profile specific	
6	Profile specific	
7	—	Reserved

41.3.4 MAC receive

The MAC receive engine performs the following tasks:

- Check frame framing
- Remove frame preamble and frame SFD field
- Discard frame based on frame destination address field
- Terminate pause frames
- Check frame length
- Remove payload padding if it exists

- Calculate and verify CRC-32
- Write received frames in the core receive FIFO

If the MAC is programmed to operate in half-duplex mode, it will also check if the frame is received with a collision.

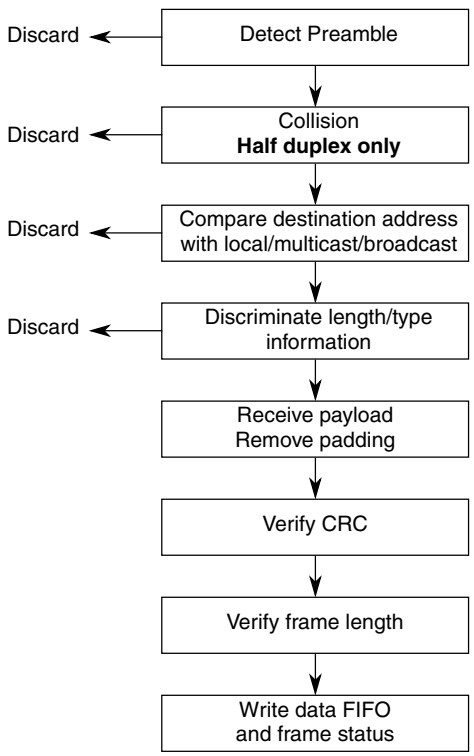


Figure 41-5. MAC receive flow

41.3.4.1 Collision detection in half-duplex mode

If the packet is received with a collision detected during reception of the first 64 bytes, the packet is discarded (if frame size was less than ~14 octets) or transmitted to the user application with an error and RxBD[CE] set.

41.3.4.2 Preamble processing

The IEEE 802.3 standard allows a maximum size of 56 bits (seven bytes) for the preamble, while the MAC core allows any preamble length, including zero length preamble.

The MAC core checks for the start frame delimiter (SFD) byte. If the next byte of the preamble, which is different from 0x55, is not 0xD5, the frame is discarded.

Although the IEEE specification dictates that the inner-packet gap should be at least 96 bits, the MAC core is designed to accept frames separated by only 64 10/100-Mbit/s operation (MII) bits.

The MAC core removes the preamble and SFD bytes.

41.3.4.3 MAC address check

The destination address bit 0 differentiates between multicast and unicast addresses.

- If bit 0 is 0, the MAC address is an individual (unicast) address.
- If bit 0 is 1, the MAC address defines a group (multicast) address.
- If all 48 bits of the MAC address are set, it indicates a broadcast address.

41.3.4.3.1 Unicast address check

If a unicast address is received, the destination MAC address is compared to the node MAC address programmed by the host in the PADDR1/2 registers.

If the destination address matches any of the programmed MAC addresses, the frame is accepted.

If Promiscuous mode is enabled (RCR[PROM] = 1) no address checking is performed and all unicast frames are accepted.

41.3.4.3.2 Multicast and unicast address resolution

The hash table algorithm used in the group and individual hash filtering operates as follows.

- The 48-bit destination address is mapped into one of 64 bits, represented by 64 bits in ENET*n*_GAUR/GALR (group address hash match) or ENET*n*_IAUR/IALR (individual address hash match).
- This mapping is performed by passing the 48-bit address through the on-chip 32-bit CRC generator and selecting the six most significant bits of the CRC-encoded result to generate a number between 0 and 63.
- The msb of the CRC result selects ENET*n*_GAUR (msb = 1) or ENET*n*_GALR (msb = 0).
- The five lsbs of the hash result select the bit within the selected register.
- If the CRC generator selects a bit set in the hash table, the frame is accepted; else, it is rejected.

For example, if eight group addresses are stored in the hash table and random group addresses are received, the hash table prevents roughly 56/64 (or 87.5%) of the group address frames from reaching memory. Those that do reach memory must be further filtered by the processor to determine if they truly contain one of the eight desired addresses.

The effectiveness of the hash table declines as the number of addresses increases.

The user must initialize the hash table registers. Use this CRC32 polynomial to compute the hash:

- $FCS(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x^{1} + 1$

If Promiscuous mode is enabled (ENET*n*_RCR[PROM] = 1) all unicast and multicast frames are accepted regardless of ENET*n*_GAUR/GALR and ENET*n*_IAUR/IALR settings.

41.3.4.3.3 Broadcast address reject

All broadcast frames are accepted if BC_REJ is cleared or ENET*n*_RCR[PROM] is set. If PROM is cleared when ENET*n*_RCR[BC_REJ] is set, all broadcast frames are rejected.

Table 41-22. Broadcast address reject programming

PROM	BC_REJ	Broadcast frames
0	0	Accepted
0	1	Rejected
1	0	Accepted
1	1	Accepted

41.3.4.3.4 Miss-bit implementation

For higher layer filtering purposes, RxBD[M] indicates an address miss when the MAC operates in promiscuous mode and accepts a frame that would otherwise be rejected.

If a group/individual hash or exact match does not occur and Promiscuous mode is enabled (RCR[PROM] = 1), the frame is accepted and the M bit is set in the buffer descriptor; otherwise, the frame is rejected.

This means the status bit is set in any of the following conditions during Promiscuous mode:

- A broadcast frame is received when BC_REJ is set

- A unicast is received that does not match either:
  - Node address (PALR[PADDR1] and PAUR[PADDR2])
  - Hash table for unicast (IAUR[IADDR1] and IALR[IADDR2])
- A multicast is received that does not match the GAUR[GADDR1] and GALR[GADDR2] hash table entries

41.3.4.4 Frame length/type verification: payload length check

If the length/type is less than 0x600 and NLC is set, the MAC checks the payload length and reports any error in the frame status word RCR[NLC] and interrupt bit EIR[PLR].

If the length/type is greater than or equal to 0x600, the MAC interprets the field as a type and no payload length check is performed.

The length check is performed on VLAN and stacked VLAN frames. If a padded frame is received, no length check can be performed due to the extended frame payload because padded frames can never have a payload length error.

41.3.4.5 Frame length/type verification: frame length check

When the receive frame length exceeds MAX_FL bytes, the BABR interrupt is generated and the RxBD[LG] bit is set.

The frame is not truncated unless the frame length exceeds the value programmed in ENET*n*_FTRL[TRUNC_FL]. If the frame is truncated, RxBD[TR] is set. In addition, a truncated frame always has the CRC error indication set (RxBD[CR]).

41.3.4.6 VLAN frames processing

VLAN frames have a length/type field set to 0x8100 immediately followed by a 16-Bit VLAN control information field.

VLAN-tagged frames are received as normal frames because the VLAN tag is not interpreted by the MAC function, and are pushed complete with the VLAN tag to the user application. If the length/type field of the VLAN-tagged frame, which is found four octets later in the frame, is less than 42, the padding is removed. In addition, the frame status word (RxBD[VLAN]) indicates that the current frame is VLAN tagged.

41.3.4.7 Pause frame termination

The receive engine terminates pause frames and does not transfer them to the receive FIFO. The quanta is extracted and sent to the MAC transmit path via a small internal clock rate decoupling asynchronous FIFO.

The quanta is written only if a correct CRC and frame length are detected by the control state machine. If not, the quanta is discarded and the MAC transmit path is not paused.

Valid pause frames are ignored if ENET*n*_RCR[FCE] is cleared and are forwarded to the client interface when ENET*n*_RCR[PAUFWD] is set.

41.3.4.8 CRC check

The CRC-32 field is checked and forwarded to the core FIFO interface if ENET*n*_RCR[CRCFWD] is cleared and ENET*n*_RCR[PADEN] is cleared. When CRCFWD is set (regardless of PADEN), the CRC-32 field is checked and terminated (not transmitted to the FIFO).

The CRC polynomial, as specified in the 802.3 standard, is:

- $FCS(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x^{1} + 1$

The 32 bits of the CRC value are placed in the frame check sequence (FCS) field with the x^{31} term as right-most bit of the first octet. The CRC bits are thus received in the following order: x^{31}, x^{30},..., x^{1}, x^{0}.

If a CRC error is detected, the frame is marked invalid and RxBD[CR] is set.

41.3.4.9 Frame padding removal

When a frame is received with a payload length field set to less than 46 (42 for VLAN-tagged frames and 38 for frames with stacked VLANs), the zero padding can be removed before the frame is written into the data FIFO depending on the setting of ENET*n*_RCR[PADEN].

Note

If a frame is received with excess padding (in other words, the length field is set as mentioned above, but the frame has more than 64 octets) and padding removal is enabled, then the

padding is removed as normal and no error is reported if the frame is otherwise correct (for example: good CRC, less than maximum length, and no other error).

41.3.5 MAC transmit

Frame transmission starts when the transmit FIFO holds enough data.

After a transfer starts, the MAC transmit function performs the following tasks:

- Generates preamble and SFD field before frame transmission
- Generates XOFF pause frames if the receive FIFO reports a congestion or if ENET*n*_TCR[TFC_PAUSE] is set with ENET*n*_OPD[PAUSE_DUR] set to a non-zero value
- Generates XON pause frames if the receive FIFO congestion condition is cleared or if TFC_PAUSE is set with PAUSE_DUR cleared
- Suspends Ethernet frame transfer (XOFF) if a non-zero pause quanta is received from the MAC receive path
- Adds padding to the frame if required
- Calculates and appends CRC-32 to the transmitted frame
- Sends the frame with correct inter-packet gap (IPG) (deferring)

When the MAC is configured to operate in half-duplex mode, the following additional tasks are performed:

- Collision detection
- Frame retransmit after back-off timer expires

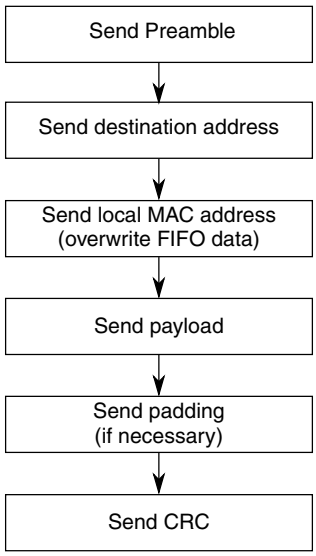


Figure 41-6. Frame transmit overview

41.3.5.1 Frame payload padding

The IEEE specification defines a minimum frame length of 64 bytes.

If the frame sent to the MAC from the user application has a size smaller than 60 bytes, the MAC automatically adds padding bytes (0x00) to comply with the Ethernet minimum frame length specification. Transmit padding is always performed and cannot be disabled.

If the MAC is not allowed to append a CRC (TxBD[TC] = 1), the user application is responsible for providing frames with a minimum length of 64 octets.

41.3.5.2 MAC address insertion

On each frame received from the core transmit FIFO interface, the source MAC address is either:

- Replaced by the address programmed in the PADDR1/2 fields (ENET*n*_TCR[ADDINS] = 1)
- Transparently forwarded to the Ethernet line (ENET*n*_TCR[ADDINS] = 0)

41.3.5.3 CRC-32 generation

The CRC-32 field is optionally generated and appended at the end of a frame.

The CRC polynomial, as specified in the 802.3 standard, is:

- $FCS(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^{8} + x^{7} + x^{5} + x^{4} + x^{2} + x^{1} + 1$

The 32 bits of the CRC value are placed in the FCS field so that the x^{31} term is the right-most bit of the first octet. The CRC bits are thus transmitted in the following order: x^{31}, x^{30},..., x^{1}, x^{0}.

41.3.5.4 Inter-packet gap (IPG)

In full-duplex mode, after frame transmission and before transmission of a new frame, an IPG (programmed in ENET*n*_TIPG) is maintained. The minimum IPG can be programmed between 8 and 26 byte-times (64 and 208 bit-times).

In half-duplex mode, the core constantly monitors the line. Actual transmission of the data onto the network occurs only if it has been idle for a 96-bit time period, and any back-off time requirements have been satisfied. In accordance with the standard, the core begins to measure the IPG from CRS de-assertion.

41.3.5.5 Collision detection and handling — half-duplex operation only

A collision occurs on a half-duplex network when concurrent transmissions from two or more nodes take place. During transmission, the core monitors the line condition and detects a collision when the PHY device asserts COL.

When the core detects a collision while transmitting, it stops transmission of the data and transmits a 32-bit jam pattern. If the collision is detected during the preamble or the SFD transmission, the jam pattern is transmitted after completing the SFD, which results in a minimum 96-bit fragment. The jam pattern is a fixed pattern that is not compared to the actual frame CRC, and has a very low probability (0.532) of having a jam pattern identical to the CRC.

If a collision occurs before transmission of 64 bytes (including preamble and SFD), the MAC core waits for the backoff period and retransmits the packet data (stored in a 64-byte re-transmit buffer) that has already been sent on the line. The backoff period is generated from a pseudo-random process (truncated binary exponential backoff).

If a collision occurs after transmission of 64 bytes (including preamble and SFD), the MAC discards the remainder of the frame, optionally sets the LC interrupt bit, and sets TxBD[LCE].

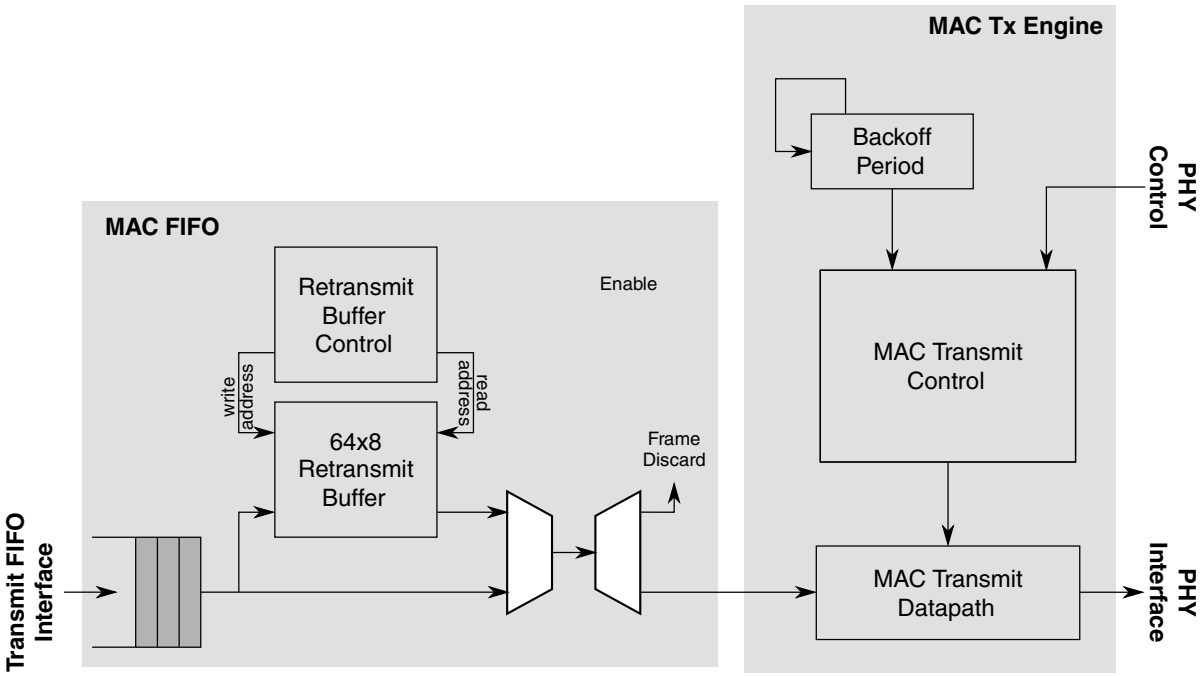


Figure 41-7. Packet re-transmit overview

The backoff time is represented by an integer multiple of slot times. One slot is equal to a 512-bit time period. The number of the delay slot times, before the n^{th} re-transmission attempt, is chosen as a uniformly-distributed random integer in the range:

- $0 < r < 2^k$
- $k = \min(n, N)$; where n is the number of retransmissions and $N = 10$

For example, after the first collision, the backoff period is 0 or 1 slot time. If a collision occurs on the first retransmission, the backoff period is 0, 1, 2, or 3, and so on.

The maximum backoff time (in 512-bit time slots) is limited by N = 10 as specified in the IEEE 802.3 standard.

If a collision occurs after 16 consecutive retransmissions, the core reports an excessive collision condition (ENET*n*_EIR[RL] interrupt field and TxBD[EE]) and discards the current packet from the FIFO.

In networks violating the standard requirements, a collision may occur after transmission of the first 64 bytes. In this case, the core stops the current packet transmission and discards the rest of the packet from the transmit FIFO. The core resumes transmission with the next packet available in the core transmit FIFO.

warning

Ethernet PHYs that support the SQE Test, or "heartbeat,"
feature must disable this feature. When this feature is enabled,

the PHY asserts the collision signal after a frame is transmitted to indicate to the ENET that the PHY's collision logic is working. This may cause data corruption in the next frame from the ENET. This corrupted frame contains up to 21 zero bytes which start somewhere within the MAC destination address field. The ENET, however, will still generate a good FCS (CRC-32) but with corrupted data.

41.3.6 Full-duplex flow control operation

Three conditions are handled by the core's flow control engine:

- Remote device congestion — The remote device connected to the same Ethernet segment as the core reports congestion and requests that the core stop sending data.
- Core FIFO congestion — When the core's receive FIFO reaches a user-programmable threshold (RX section empty), the core sends a pause frame back to the remote device requesting the data transfer to stop.
- Local device congestion — Any device connected to the core can request (typically, via the host processor) the remote device to stop transmitting data.

41.3.6.1 Remote device congestion

When the MAC transmit control gets a valid pause quanta from the receive path and if ENET*n*_RCR[FCE] is set, the MAC transmit logic:

- Completes the transfer of the current frame.
- Stops sending data for the amount of time specified by the pause quanta in 512 bit time increments.
- Sets ENET*n*_TCR[RFC_PAUSE].

Frame transfer resumes when the time specified by the quanta expires and if no new quanta value is received, or if a new pause frame with a quanta value set to 0x0000 is received. The MAC also resets RFC_PAUSE to zero.

If ENET*n*_RCR[FCE] cleared, the MAC ignores received pause frames.

Optionally and independent of ENET*n*_RCR[FCE], pause frames are forwarded to the client interface if PAUFWD is set.

41.3.6.2 Local device/FIFO congestion

The MAC transmit engine generates pause frames when the local receive FIFO is not able to receive more than a pre-defined number of words (FIFO programmable threshold) or when pause frame generation is requested by the local host processor.

- To generate a pause frame, the host processor sets ENET*n*_TCR[TFC_PAUSE]. A single pause frame is generated when the current frame transfer is completed and TFC_PAUSE is automatically cleared. Optionally, an interrupt is generated.
- An XOFF pause frame is generated when the receive FIFO asserts its section empty flag (internal). An XOFF pause frame is generated automatically, when the current frame transfer completes.
- An XON pause frame is generated when the receive FIFO deasserts its section empty flag (internal). An XON pause frame is generated automatically, when the current frame transfer completes.

When an XOFF pause frame is generated, the pause quanta (payload byte P1 and P2) is filled with the value programmed in ENET*n*_OPD[PAUSE_DUR].

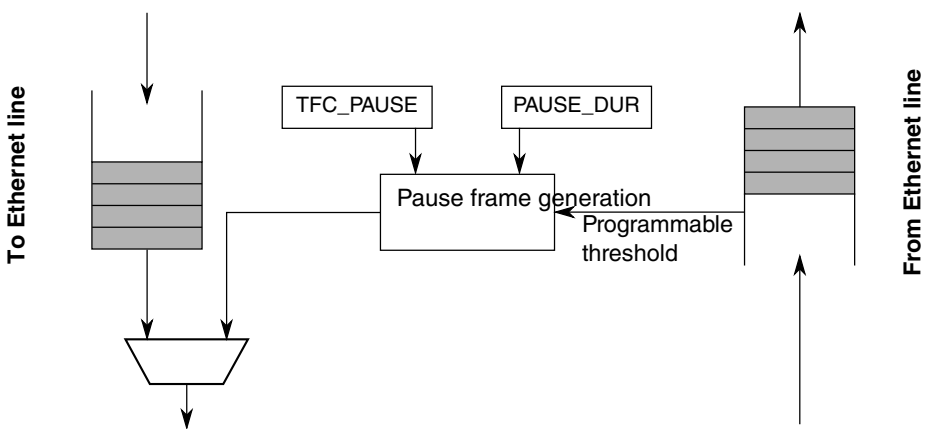


Figure 41-8. Pause frame generation overview

Note

Although the flow control mechanism should prevent any FIFO overflow on the MAC core receive path, the core receive FIFO is protected. When an overflow is detected on the receive FIFO, the current frame is truncated with an error indication set in the frame status word. The frame should subsequently be discarded by the user application.

41.3.7 Magic packet detection

Magic packet detection wakes a node that is put in power-down mode by the node management agent. Magic packet detection is supported only if the MAC is configured in sleep mode.

41.3.7.1 Sleep mode

To put the MAC in Sleep mode, set ENET*n*_ECR[SLEEP]. At the same time ENET*n*_ECR[MAGICEN] should also be set to enable magic packet detection.

In addition, if ENET is enabled, write 1 to ENET*n*_ECR[SLEEP] before entering into low power mode.

When the MAC is in Sleep mode:

- The transmit logic is disabled.
- The FIFO receive/transmit functions are disabled.
- The receive logic is kept in Normal mode, but it ignores all traffic from the line except magic packets. They are detected so that a remote agent can wake the node.

41.3.7.2 Magic packet detection

The core is designed to detect magic packets (See "Magic Packets" topics) with the destination address set to:

- Any multicast address
- The broadcast address
- The unicast address programmed in PADDR1/2

When a magic packet is detected, EIR[WAKEUP] is set and none of the statistic registers are incremented.

41.3.7.3 Wakeup

When a magic packet is detected, indicated by ENET*n*_EIR[WAKEUP], ENET*n*_ECR[SLEEP] should be cleared to resume normal operation of the MAC. Clearing the SLEEP bit automatically masks ENET*n*_ECR[MAGICEN], disabling magic packet detection.

41.3.8 IP accelerator functions

The following sections describe the IP accelerator functions.

41.3.8.1 Checksum calculation

The IP and ICMP, TCP, UDP checksums are calculated with one's complement arithmetic summing up 16-bit values.

- For ICMP, the checksum is calculated over the complete ICMP datagram, in other words without IP header.
- For TCP and UDP, the checksums contain the header and data sections and values from the IP header, which can be seen as a pseudo-header that is not actually present in the data stream.

Table 41-23. IPv4 pseudo-header for checksum calculation

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
Source address							
Destination address							
Zero		Protocol		TCP/UDP length			

Table 41-24. IPv6 pseudo-header for checksum calculation

31 30 29 28	27 26 25 24	23 22 21 20	19 18 17 16	15 14 13 12	11 10 9 8	7 6 5 4	3 2 1 0
Source address							
Destination address							
TCP/UDP length							
Zero						Next header	

The TCP/UDP length value is the length of the TCP or UDP datagram, which is equal to the payload of an IP datagram. It is derived by subtracting the IP header length from the complete IP datagram length that is given in the IP header (IPv4), or directly taken from the IP header (IPv6). The protocol field is the corresponding value from the IP header. The Zero fields are all zeroes.

For IPv6, the complete 128-bit addresses are considered. The next header value identifies the upper layer protocol as either TCP or UDP. It may differ from the next header value of the IPv6 header if extension headers are inserted before the protocol header.

The checksum calculation uses 16-bit words in network byte order: The first byte sent/received is the MSB, and the second byte sent/received is the LSB of the 16-bit value to add to the checksum. If the frame ends on an odd number of bytes, a zero byte is appended for checksum calculation only, and is not actually transmitted.

41.3.8.2 Additional padding processing

According to IEEE 802.3, any Ethernet frame must have a minimum length of 64 octets.

The MAC usually removes padding on receive when a frame with length information is received. Because IP frames have a type value instead of length, the MAC does not remove padding for short IP frames, as it is not aware of the frame contents.

The IP accelerator function can be configured to remove the Ethernet padding bytes that might follow the IP datagram.

On transmit, the MAC automatically adds padding as necessary to fill any frame to a 64-byte length.

41.3.8.3 32-bit Ethernet payload alignment

The data FIFOs allow inserting two additional arbitrary bytes in front of a frame. This extends the 14-byte Ethernet header to a 16-byte header, which leads to alignment of the Ethernet payload, following the Ethernet header, on a 32-bit boundary.

This function can be enabled for transmit and receive independently with the corresponding SHIFT16 bits in the ENET*n*_TACC and ENET*n*_RACC registers.

When enabled, the valid frame data is arranged as shown in Table 41-25.

Table 41-25. 64-bit interface data structure with SHIFT16 enabled

63 56	55 48	47 40	39 32	31 24	23 16	15 8	7 0

Table continues on the next page...

Table 41-25. 64-bit interface data structure with SHIFT16 enabled (continued)

Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0	Any value	Any value
Byte 13	Byte 12	Byte 11	Byte 10	Byte 9	Byte 8	Byte 7	Byte 6
...							

41.3.8.3.1 Receive processing

When ENET*n*_RACC[SHIFT16] is set, each frame is received with two additional bytes in front of the frame.

The user application must ignore these first two bytes and find the first byte of the frame in bits 23–16 of the first word from the RX FIFO.

Note

SHIFT16 must be set during initialization and kept set during the complete operation, because it influences the FIFO write behavior.

41.3.8.3.2 Transmit processing

When ENET*n*_TACC[SHIFT16] is set, the first two bytes of the first word written (bits 15–0) are discarded immediately by the FIFO write logic.

The SHIFT16 bit can be enabled/disabled for each frame individually if required, but can be changed only between frames.

41.3.8.4 Received frame discard

Because the receive FIFO must be operated in store and forward mode (ENET*n*_RSFL cleared), received frames can be discarded based on the following errors:

- The MAC function receives the frame with an error:
  - The frame has an invalid payload length
  - Frame length is greater than MAX_FL
  - Frame received with a CRC-32 error
  - Frame truncated due to receive FIFO overflow
  - Frame is corrupted as PHY signaled an error (RX_ERR asserted during reception)

- An IP frame is detected and the IP header checksum is wrong
- An IP frame with a valid IP header and a valid IP header checksum is detected, the protocol is known but the protocol-specific checksum is wrong

If one of the errors occurs and the IP accelerator function is configured to discard frames (ENET*n*_RACC), the frame is automatically discarded. Statistics are maintained normally and are not affected by this discard function.

41.3.8.5 IPv4 fragments

When an IPv4 IP fragment frame is received, only the IP header is inspected and its checksum verified. 32-bit alignment operates the same way on fragments as it does on normal IP frames, as specified above.

The IP fragment frame payload is not inspected for any protocol headers. As such, a protocol header would only exist in the very first fragment. To assist in protocol-specific checksum verification, the one's-complement sum is calculated on the IP payload (all bytes following the IP header) and provided with the frame status word.

The frame fragment status field (RxBD[FRAG]) is set to indicate a fragment reception, and the one's-complement sum of the IP payload is available in RxBD[Payload checksum].

Note

After all fragments have been received and reassembled, the application software can take advantage of the payload checksum delivered with the frame's status word to calculate the protocol-specific checksum of the datagram.

For example, if a TCP payload is delivered by multiple IP fragments, the application software can calculate the pseudo-header checksum value from the first fragment, and add the payload checksums delivered with the status for all fragments to verify the TCP datagram checksum.

41.3.8.6 IPv6 support

The following sections describe the IPv6 support.

41.3.8.6.1 Receive processing

An Ethernet frame of type 0x86DD identifies an IP Version 6 frame (IPv6) frame. If an IPv6 frame is received, the first IP header is inspected (first ten words), which is available in every IPv6 frame.

If the receive SHIFT16 function is enabled, the IP header is aligned on a 32-bit boundary allowing more efficient processing (see 32-bit Ethernet payload alignment).

For TCP and UDP datagrams, the pseudo-header checksum calculation is performed and verified.

To assist in protocol-specific checksum verification, the one's-complement sum is always calculated on the IP payload (all bytes following the IP header) and provided with the frame status word. For example, if extension headers were present, their sums can be subtracted in software from the checksum to isolate the TCP/UDP datagram checksum, if required.

41.3.8.6.2 Transmit processing

For IPv6 transmission, the SHIFT16 function is supported to process 32-bit aligned datagrams.

IPv6 has no IP header checksum; therefore, the IP checksum insertion configuration is ignored.

The protocol checksum is inserted only if the next header of the IP header is a known protocol (TCP, UDP, or ICMP). If a known protocol is detected, the checksum over all bytes following the IP header is calculated and inserted in the correct position.

The pseudo-header checksum calculation is performed for TCP and UDP datagrams accordingly.

41.3.9 Resets and stop controls

The following sections describe the resets and stop controls.

41.3.9.1 Hardware reset

To reset the Ethernet module, set ENET*n*_ECR[RESET].

41.3.9.2 Soft reset

When ENET*n*_ECR[ETHER_EN] is cleared during operation, the following occurs:

- uDMA, buffer descriptor, and FIFO control logic are reset, including the buffer descriptor and FIFO pointers.
- A currently ongoing transmit is terminated by asserting TXER to the PHY.
- A currently ongoing transmit FIFO write from the application is terminated by stopping the write to the FIFO, and all further data from the application is ignored. All subsequent writes are ignored until re-enabled.
- A currently ongoing receive FIFO read is terminated. The RxBD has arbitrary values in this case.

41.3.9.3 Debug mode

When the processor enters debug mode and ECR[DBGEN] is set, the MAC enters a freeze state where it stops all transmit and receive activities gracefully.

The following happens when the MAC enters hardware freeze:

- A currently ongoing receive transaction on the receive application interface is completed as normal. No further frames are read from the FIFO.
- A currently ongoing transmit transaction on the transmit application interface is completed as normal (in other words, until writing end-of-packet (EOP)).
- A currently ongoing frame receive is completed normally, after which no further frames are accepted from the MII.
- A currently ongoing frame transmit is completed normally, after which no further frames are transmitted.

41.3.9.4 Graceful stop

During a graceful stop, any currently ongoing transactions are completed normally and no further frames are accepted. The MAC can resume from a graceful stop without the need for a reset (for example, clearing ETHER_EN is not required).

The following conditions lead to a graceful stop of the MAC transmit or receive datapaths.

41.3.9.4.1 Graceful transmit stop (GTS)

When gracefully stopped, the MAC is no longer reading frame data from the transmit FIFO and has completed any ongoing transmission.

In any of the following conditions, the transmit datapath stops after an ongoing frame transmission has been completed normally.

- ENET*n*_TCR[GTS] is set by software.
- ENET*n*_TCR[TFC_PAUSE] is set by software requesting a pause frame transmission. The status (and register bit) is cleared after the pause frame has been sent.
- A pause frame was received stopping the transmitter. The stopped situation is terminated when the pause timer expires or a pause frame with zero quanta is received.
- MAC is placed in Sleep mode by software or the processor entering Stop mode (see Sleep mode).
- The MAC is in Hardware Freeze mode.

When the transmitter has reached its stopped state, the following events occur:

- The GRA interrupt is asserted, when transitioned into stopped state.
- In Hardware Freeze mode, the GRA interrupt does not wait for the application write completion and asserts when the transmit state machine (in other words, line side of TX FIFO) reaches its stopped state.

41.3.9.4.2 Graceful receive stop (GRS)

When gracefully stopped, the MAC is no longer writing frames into the receive FIFO.

The receive datapath stops after any ongoing frame reception has been completed normally, if any of the following conditions occur:

- MAC is placed in Sleep mode either by the software or the processor is in Stop mode). The MAC continues to receive frames and search for magic packets if enabled (see Magic packet detection). However, no frames are written into the receive FIFO, and therefore are not forwarded to the application.
- The MAC is in Hardware Freeze mode. The MAC does not accept any frames from the MII.

When the receive datapath is stopped, the following events occur:

- If the RX is in the stopped state, RCR[GRS] is set
- The GRA interrupt is asserted when the transmitter and receiver are stopped
- Any ongoing receive transaction to the application (RX FIFO read) continues normally until the frame end of package (EOP) is reached. After this, the following occurs:
  - When Sleep mode is active, all further frames are discarded, flushing the RX FIFO
  - In Hardware Freeze mode, no further frames are delivered to the application and they stay in the receive FIFO.

Note

The assertion of GRS does not wait for an ongoing FIFO read transaction on the application side of the FIFO (FIFO read).

41.3.9.4.3 Graceful stop interrupt (GRA)

The graceful stop interrupt (GRA) is asserted for the following conditions:

- In Sleep mode, the interrupt asserts only after both TX and RX datapaths are stopped.
- In Hardware Freeze mode, the interrupt asserts only after both TX and RX datapaths are stopped.
- The MAC transmit datapath is stopped for any other condition (GTS, TFC_PAUSE, pause received).

The GRA interrupt is triggered only once when the stopped state is entered. If the interrupt is cleared while the stop condition persists, no further interrupt is triggered.

41.3.10 IEEE 1588 functions

To allow for IEEE 1588 or similar time synchronization protocol implementations, the MAC is combined with a time-stamping module to support precise time-stamping of incoming and outgoing frames. Set ENET*n*_ECR[EN1588] to enable 1588 support.

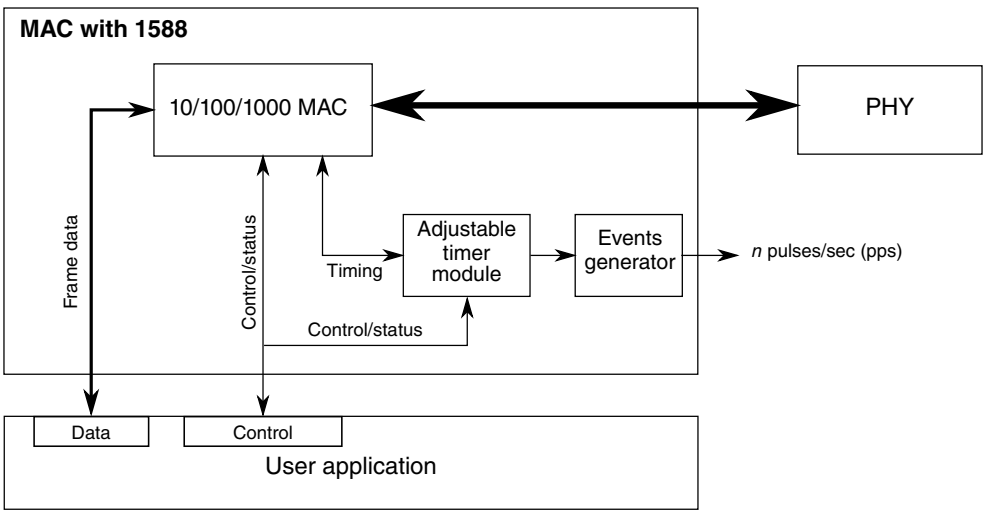


Figure 41-9. IEEE 1588 functions overview

41.3.10.1 Adjustable timer module

The adjustable timer module (TSM) implements the free-running counter (FRC), which generates the timestamps. The FRC operates with the time-stamping clock, which can be set to any value depending on your system requirements.

Through dedicated correction logic, the timer can be adjusted to allow synchronization to a remote master and provide a synchronized timing reference to the local system. The timer can be configured to cause an interrupt after a fixed time period, to allow synchronization of software timers or perform other synchronized system functions.

The timer is typically used to implement a period of one second; hence, its value ranges from 0 to (1×10^9)-1. The period event can trigger an interrupt, and software can maintain the seconds and hours time values as necessary.

41.3.10.1.1 Adjustable timer implementation

The adjustable timer consists of a programmable counter/accumulator and a correction counter. The periods of both counters and their increment rates are freely configurable, allowing very fine tuning of the timer.

See Timer Synchronization for Multi-Port Implementations, for external clock input options.

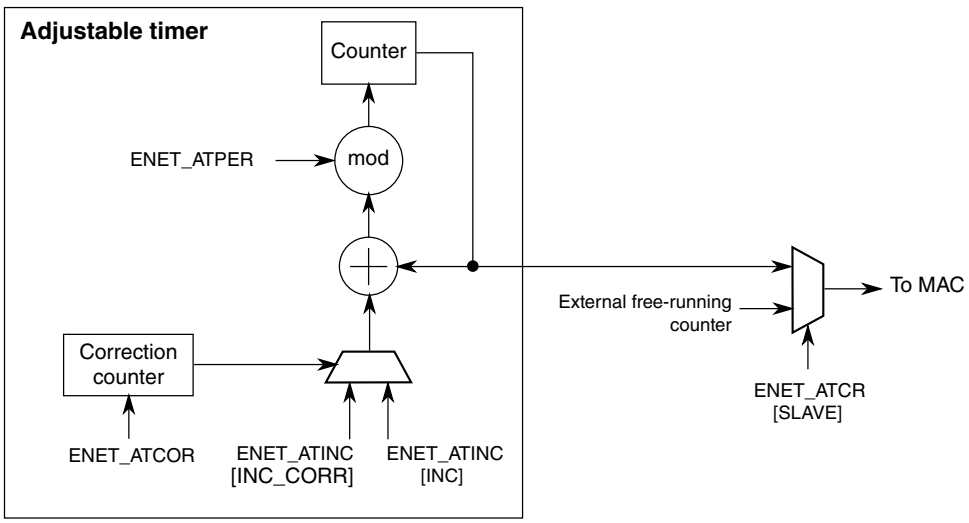


Figure 41-10. Adjustable timer implementation detail

The counter produces the current time. During each time-stamping clock cycle, a constant value is added to the current time as programmed in ENET*n*_ATINC. The value depends on the chosen time-stamping clock frequency. For example, if it operates at 125 MHz, setting the increment to eight represents 8 ns.

The period, configured in ENET*n*_ATPER, defines the modulo when the counter wraps. In a typical implementation, the period is set to 1×10^9 so that the counter wraps every second, and hence all timestamps represent the absolute nanoseconds within the one second period. When the period is reached, the counter wraps to start again respecting the period modulo. This means it does not necessarily start from zero, but instead the counter is loaded with the value (Current + Inc $-(1 \times 10^9)$), assuming the period is set to 1×10^9.

The correction counter operates fully independently, and increments by one with each time-stamping clock cycle. When it reaches the value configured in ENET*n*_ATCOR, it restarts and instructs the timer once to increment by the correction value, instead of the normal value.

The normal and correction increments are configured in ENET*n*_ATINC. To speed up the timer, set the correction increment more than the normal increment value. To slow down the timer, set the correction increment less than the normal increment value.

The correction counter only defines the distance of the corrective actions, not the amount. This allows very fine corrections and low jitter (in the range of 1 ns) independent of the chosen clock frequency.

By enabling slave mode (ENET*n*_ATCR[SLAVE] = 1), the timer is ignored and the current time is externally provided from one of the external modules. See the Chip Configuration details for which clock source is used. This is useful if multiple modules within the system must operate from a single timer (see Timer Synchronization for Multi-Port Implementations). When slave mode is enabled, you still must set ENET*n*_ATINC[INC] to the value of the master, since it is used for internal comparisons.

41.3.10.2 Timer Synchronization for Multi-Port Implementations

Additional inputs are available to provide a timer value for all time-stamping functions. This is necessary to synchronize the two MACs to a single reference timer. See the Chip Configuration details for available clock inputs to the time-stampimg functions. To operate the MAC in slave mode, ENET*n*_ATCR[SLAVE] disables the internal adjustable timer and uses the externally provided timer.

41.3.10.3 Transmit timestamping

Only 1588 event frames need to be time-stamped on transmit. The client application (for example, the MAC driver) should detect 1588 event frames and set TxBD[TS] together with the frame.

If TxBD[TS] is set, the MAC records the timestamp for the frame in ENET*n*_ATSTMP. ENET*n*_EIR[TS_AVAIL] is set to indicate that a new timestamp is available.

Software implements a handshaking procedure by setting TxBD[TS] when it transmits the frame for which a timestamp is needed, and then waits for ENET*n*_EIR[TS_AVAIL] to determine when the timestamp is available. The timestamp is then read from ENET*n*_ATSTMP. This is done for all event frames. Other frames do not use TxBD[TS] and, therefore, do not interfere with the timestamp capture.

41.3.10.4 Receive timestamping

When a frame is received, the MAC latches the value of the timer when the frame's start of frame delimiter (SFD) field is detected, and provides the captured timestamp on RxBD[1588 timestamp]. This is done for all received frames.

41.3.10.5 Time synchronization

The adjustable timer module is available to synchronize the local clock of a node to a remote master. It implements a free running 32-bit counter, and also contains an additional correction counter.

The correction counter increases or decreases the rate of the free running counter, enabling very fine granular changes of the timer for synchronization, yet adding only very low jitter when performing corrections.

The application software implements, in a slave scenario, the required control algorithm, setting the correction to compensate for local oscillator drifts and locking the timer to the remote master clock on the network.

The timer and all timestamp-related information should be configured to show the true nanoseconds value of a second (in other words, the timer is configured to have a period of one second). Hence, the values range from 0 to (1×10^9)–1. In this application, the seconds counter is implemented in software using an interrupt function that is executed when the nanoseconds counter wraps at 1×10^9.

41.3.10.6 Input Capture and Output Compare

The Input Capture Output Compare block can be used to provide precise hardware timing for input and output events.

41.3.10.6.1 Input capture

The TCCR*n* capture registers latch the time value when the corresponding external event occurs. An event can be a rising-, falling-, or either-edge of one of the 1588_TMR*n* signals. An event will cause the corresponding TCSR*n*[TF] timer flag to be set, indicating that an input capture has occurred. If the corresponding interrupt is enabled with the TCSR*n*[TIE] field, an interrupt can be generated.

41.3.10.6.2 Output compare

The TCCR*n* compare registers are loaded with the time at which the corresponding event should occur. When the ENET free-running counter value matches the output compare reference value in the TCCR*n* register, the corresponding flag, TCSR*n*[TF], is set, indicating that an output compare has occurred. The corresponding interrupt, if enabled by TCSR*n*[TIE], will be generated.The corresponding 1588_TMR*n* output signal will be asserted according to TCSR*n*[TMODE].

NOTE

If TSCR*n*[TMODE] is set to 10X1b or 1010b then the timer output pin toggle-on-overflow will occur only when PINPER, PEREN, and EN bits of the ATCR register are one.

41.3.10.6.3 DMA requests

A DMA request can be enabled by setting TCSR*n*[TDRE]. The corresponding DMA request is generated when the TCSR*n*[TF] timer flag is set. When the DMA has completed, the corresponding TCSRn[TF] flag is cleared.

41.3.11 FIFO thresholds

The core FIFO thresholds are fully programmable to dynamically change the FIFO operation.

For example, store and forward transfer can be enabled by a simple change in the FIFO threshold registers.

The thresholds are defined in 64-bit words.

The receive and transmit FIFOs both have a depth of 256 words.

41.3.11.1 Receive FIFO

Four programmable thresholds are available, which can be set to any value to control the core operation as follows.

Table 41-26. Receive FIFO thresholds definition

Register	Description
ENET*n*_RSFL [RX_SECTION_FULL]	When the FIFO level reaches the ENET*n*_RSFL value, the MAC status signal is asserted to indicate that data is available in the receive FIFO (cut-through operation). Once asserted, if the FIFO empties below the threshold set with ENET*n*_RAEM and if the end-of-frame is not yet stored in the FIFO, the status signal is deasserted again. If a frame has a size smaller than the threshold (in other words, an end-of-frame is available for the frame), the status is also asserted. To enable store and forward on the receive path, clear ENET*n*_RSFL. The MAC status signal is asserted only when a complete frame is stored in the receive FIFO. When programming a non-zero value to ENET*n*_RSFL (cut-through operation) it should be greater than ENET*n*_RAEM.
ENET*n*_RAEM	When the FIFO level reaches the ENET*n*_RAEM value and the end-of-frame has not been received, the core receive read control stops the FIFO read (and subsequently stops transferring data to the MAC client application).

Table continues on the next page...

Table 41-26. Receive FIFO thresholds definition (continued)

Register	Description
[RX_ALMOST_EMPTY]	It continues to deliver the frame, if again more data than the threshold or the end-of-frame is available in the FIFO. Set ENET*n*_RAEM to a minimum of six.
ENET*n*_RAFL [RX_ALMOST_FULL]	When the FIFO level approaches the maximum and there is no more space remaining for at least ENETn_RAFL number of words, the MAC control logic stops writing data in the FIFO and truncates the receive frame to avoid FIFO overflow. The corresponding error status is set when the frame is delivered to the application. Set ENET*n*_RAFL to a minimum of 4.
ENET*n*_RSEM [RX_SECTION_EMPTY]	When the FIFO level reaches the ENET*n*_RSEM value, an indication is sent to the MAC transmit logic, which generates an XOFF pause frame. This indicates FIFO congestion to the remote Ethernet client. When the FIFO level goes below the value programmed in ENET*n*_RSEM, an indication is sent to the MAC transmit logic, which generates an XON pause frame. This indicates the FIFO congestion is cleared to the remote Ethernet client. Clearing ENET*n*_RSEM disables any pause frame generation.

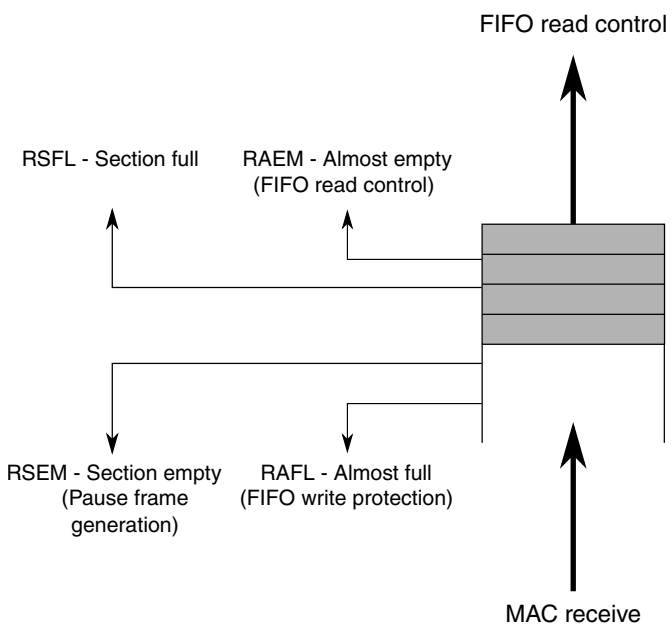


Figure 41-11. Receive FIFO overview

41.3.11.2 Transmit FIFO

Four programmable thresholds are available which control the core operation as described below.

Table 41-27. Transmit FIFO thresholds definition

Register	Description
ENET*n*_TAEM [TX_ALMOST _EMPTY]	When the FIFO level reaches the ENET*n*_TAEM value and no end-of-frame is available for the frame, the MAC transmit logic avoids a FIFO underflow by stopping FIFO reads and transmitting the Ethernet frame with an MII error indication. Set ENET*n*_TAEM to a minimum of 4.
ENET*n*_TAFL [TX_ALMOST _FULL]	When the FIFO level approaches the maximum, so that there is no more space for at least ENET*n*_TAFL number of words, the MAC deasserts its control signal to the application. If the application does not react on this signal, the FIFO write control logic avoids FIFO overflow by truncating the current frame and setting the error status. As a result, the frame is transmitted with an MII error indication. Set ENET*n*_TAFL to a minimum of 4. Larger values allow more latency for the application to react on the MAC control signal deassertion, before the frame is truncated. A typical setting is 8, which offers 3–4 clock cycles of latency to the application to react on the MAC control signal deassertion.
ENET*n*_TSEM [TX_SECTION _EMPTY]	When the FIFO level reaches the ENET*n*_TSEM value, a MAC status signal is deasserted to indicate that the transmit FIFO is getting full. This gives the ENET module an indication to slow or stop its write transaction to avoid a buffer overflow. This is a pure indication function to the application. It has no effect within the MAC. When ENET*n*_TSEM is 0, the signal is never deasserted.
ENET*n*_TFWR	When the FIFO level reaches the ENET*n*_TFWR value and when STRFWD is cleared, the MAC transmit control logic starts frame transmission before the end-of-frame is available in the FIFO (cut-through operation). If a complete frame has a size smaller than the ENET*n*_TFWR threshold, the MAC also transmits the frame to the line. To enable store and forward on the transmit path, set STRFWD. In this case, the MAC starts to transmit data only when a complete frame is stored in the transmit FIFO.

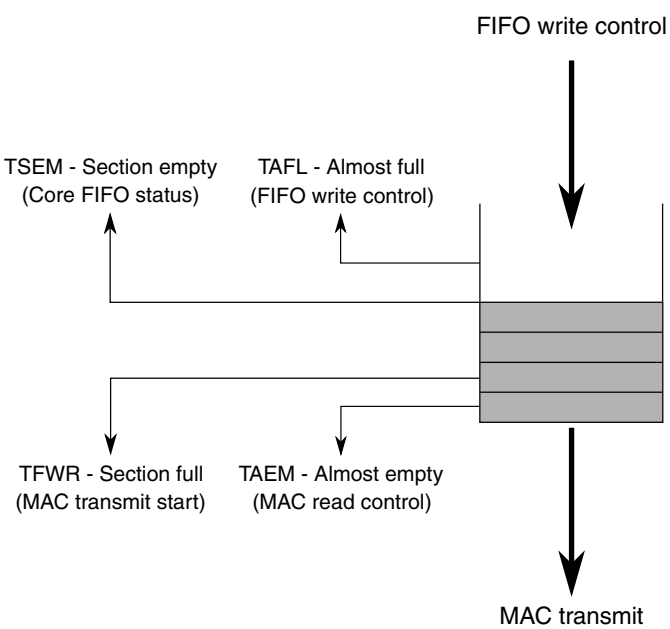


Figure 41-12. Transmit FIFO overview

41.3.12 Loopback options

The core implements internal loopback options, which are controlled by the following ENET*n*_RCR register fields:

Table 41-28. Loopback options

Register field	Description
LOOP	Internal MII loopback. The MAC transmit is returned to the MAC receive. No data is transmitted to the external interfaces. In MII internal loopback, MII_TXCLK and MII_RXCLK must be provided with a clock signal (2.5 MHz for 10 Mbit/s, and 25 MHz for 100 Mbit/s))

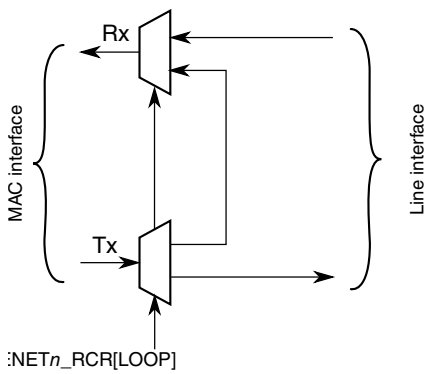


Figure 41-13. Loopback options

41.3.13 Legacy buffer descriptors

To support the Ethernet controller on previous chips, legacy FEC buffer descriptors are available. To enable legacy support, write 0 to ENET*n*_ECR[1588EN].

NOTE

- The legacy buffer descriptor tables show the byte order for little-endian chips. ECR[DBSWP] must be set to 1 after reset to enable little-endian mode.

41.3.13.1 Legacy receive buffer descriptor

The following table shows the legacy FEC receive buffer descriptor. Table 41-32 contains the descriptions for each field.

Table 41-29. Legacy FEC receive buffer descriptor (RxBD)

	Byte 1								Byte 0							
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Offset + 0	Data length															
Offset + 2	E	RO1	W	RO2	L	—	—	M	BC	MC	LG	NO	—	CR	OV	TR
Offset + 4	Rx data buffer pointer — low halfword															
Offset + 6	Rx data buffer pointer — high halfword															

41.3.13.2 Legacy transmit buffer descriptor

The following table shows the legacy FEC transmit buffer descriptor. Table 41-34 contains the descriptions for each field.

Table 41-30. Legacy FEC transmit buffer descriptor (TxBD)

	Byte 1								Byte 0							
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Offset + 0	Data Length															
Offset + 2	R	TO1	W	TO2	L	TC	ABC[1]	—	—	—	—	—	—	—	—	—
Offset + 4	Tx Data Buffer Pointer — low halfword															
Offset + 6	Tx Data Buffer Pointer — high halfword															

1. This field is not supported by the uDMA.

41.3.14 Enhanced buffer descriptors

This section provides a description of the enhanced operation of the driver/uDMA via the buffer descriptors.

It is followed by a detailed description of the receive and transmit descriptor fields. To enable the enhanced features, set ENET*n*_ECR[1588EN].

NOTE

The enhanced buffer descriptor tables show the byte order for little-endian chips. ECR[DBSWP] must be set to 1 after reset to enable little-endian mode.

41.3.14.1 Enhanced receive buffer descriptor

The following table shows the enhanced uDMA receive buffer descriptor. Table 41-32 contains the descriptions for each field.

Table 41-31. Enhanced uDMA receive buffer descriptor (RxBD)

	Byte 1								Byte 0							
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Offset + 0	Data length															
Offset + 2	E	RO1	W	RO2	L	—	—	M	BC	MC	LG	NO	—	CR	OV	TR
Offset + 4	Rx data buffer pointer – low halfword															
Offset + 6	Rx data buffer pointer – high halfword															
Offset + 8	VPCP			—	—	—	—	—	—	—	ICE	PCR	—	VLAN	IPV6	FRAG
Offset + A	ME	—	—	—	—	PE	CE	UC	INT	—	—	—	—	—	—	—
Offset + C	Payload checksum															
Offset + E	Header length					—	—	—	Protocol type							
Offset + 10	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 12	BDU	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 14	1588 timestamp – low halfword															
Offset + 16	1588 timestamp – high halfword															
Offset + 18	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 1A	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 1C	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 1E	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Table 41-32. Receive buffer descriptor field definitions

Word	Field	Description
Offset + 0	15–0 Data Length	Data length. Written by the MAC. Data length is the number of octets written by the MAC into this BD's data buffer if L is cleared (the value is equal to EMRBR), or the length of the frame including CRC if L is set. It is written by the MAC once as the BD is closed.
Offset + 2	15 E	Empty. Written by the MAC (= 0) and user (= 1). 0 The data buffer associated with this BD is filled with received data, or data reception has aborted due to an error condition. The status and length fields have been updated as required. 1 The data buffer associated with this BD is empty, or reception is currently in progress.
Offset + 2	14 RO1	Receive software ownership. This field is reserved for use by software. This read/write field is not modified by hardware, nor does its value affect hardware.
Offset + 2	13 W	Wrap. Written by user. 0 The next buffer descriptor is found in the consecutive location. 1 The next buffer descriptor is found at the location defined in ENET*n*_RDSR.
Offset + 2	12	Receive software ownership. This field is reserved for use by software. This read/write field is not modified by hardware, nor does its value affect hardware.

Table continues on the next page...

Table 41-32. Receive buffer descriptor field definitions (continued)

Word	Field	Description
	RO2	
Offset + 2	11 L	Last in frame. Written by the uDMA. 0 The buffer is not the last in a frame. 1 The buffer is the last in a frame.
Offset + 2	10–9	Reserved, must be cleared.
Offset + 2	8 M	Miss. Written by the MAC. This field is set by the MAC for frames accepted in promiscuous mode, but flagged as a miss by the internal address recognition. Therefore, while in promiscuous mode, you can use the this field to quickly determine whether the frame was destined to this station. This field is valid only if the L and PROM bits are set. 0 The frame was received because of an address recognition hit. 1 The frame was received because of promiscuous mode. The information needed for this field comes from the promiscuous_miss(ff_rx_err_stat[26]) sideband signal.
Offset + 2	7 BC	Set if the DA is broadcast (FFFF_FFFF_FFFF).
Offset + 2	6 MC	Set if the DA is multicast and not BC.
Offset + 2	5 LG	Receive frame length violation. Written by the MAC. A frame length greater than RCR[MAX_FL] was recognized. This field is valid only if the L field is set. The receive data is not altered in any way unless the length exceeds TRUNC_FL bytes.
Offset + 2	4 NO	Receive non-octet aligned frame. Written by the MAC. A frame that contained a number of bits not divisible by 8 was received, and the CRC check that occurred at the preceding byte boundary generated an error or a PHY error occurred. This field is valid only if the L field is set. If this field is set, the CR field is not set.
Offset + 2	3	Reserved, must be cleared.
Offset + 2	2 CR	Receive CRC or frame error. Written by the MAC. This frame contains a PHY or CRC error and is an integral number of octets in length. This field is valid only if the L field is set.
Offset + 2	1 OV	Overrun. Written by the MAC. A receive FIFO overrun occurred during frame reception. If this field is set, the other status fields, M, LG, NO, and CR, lose their normal meaning and are zero. This field is valid only if the L field is set.
Offset + 2	0 TR	Set if the receive frame is truncated (frame length >TRUNC_FL). If the TR field is set, the frame must be discarded and the other error fields must be ignored because they may be incorrect.
Offset + 4	15–0 Data buffer pointer low	Receive data buffer pointer, low halfword
0ffset + 6	15–0 Data buffer pointer high	Receive data buffer pointer, high halfword[1]
Offset + 8	15–13 VPCP	VLAN priority code point. This field is written by the uDMA to indicate the frame priority level. Valid values are from 0 (best effort) to 7 (highest). This value can be used to prioritize different classes of traffic (e.g., voice, video, data). This field is only valid if the L field is set.
Offset + 8	12–6	Reserved, must be cleared.

Table continues on the next page...

Table 41-32. Receive buffer descriptor field definitions (continued)

Word	Field	Description
Offset + 8	5 ICE	IP header checksum error. This is an accelerator option. This field is written by the uDMA. Set when either a non-IP frame is received or the IP header checksum was invalid. An IP frame with less than 3 bytes of payload is considered to be an invalid IP frame. This field is only valid if the L field is set.
Offset + 8	4 PCR	Protocol checksum error. This is an accelerator option. This field is written by the uDMA. Set when the checksum of the protocol is invalid or an unknown protocol is found and checksumming could not be performed. This field is only valid if the L field is set.
Offset + 8	3	Reserved, must be cleared.
Offset + 8	2 VLAN	VLAN. This is an accelerator option. This field is written by the uDMA. It means that the frame has a VLAN tag. This field is valid only if the L field is set.
Offset + 8	1 IPV6	IPV6 Frame. This field is written by the uDMA. This field indicates that the frame has an IPv6 frame type. If this field is not set it means that an IPv4 or other protocol frame was received. This field is valid only if the L field is set.
Offset + 8	0 FRAG	IPv4 Fragment.This is an accelerator option.This field is written by the uDMA. It indicates that the frame is an IPv4 fragment frame. This field is only valid when the L field is set.
Offset + A	15 ME	MAC error. This field is written by the uDMA. This field means that the frame stored in the system memory was received with an error (typically, a receive FIFO overflow). This field is only valid when the L field is set.
Offset + A	14–11	Reserved, must be cleared.
Offset + A	10 PE	PHY Error. This field is written by the uDMA. Set to "1"when the frame was received with an Error character on the PHY interface. The frame is invalid. This field is valid only when the L field is set.
Offset + A	9 CE	Collision. This field is written by the uDMA. Set when the frame was received with a collision detected during reception. The frame is invalid and sent to the user application. This field is valid only when the L field is set.
Offset + A	8 UC	Unicast. This field is written by the uDMA, and means that the frame is unicast. This field is valid regardless of whether the L field is set.
Offset + A	7 INT	Generate RXB/RXF interrupt. This field is set by the user to indicate that the uDMA is to generate an interrupt on the *dma_int_rxb* / *dma_int_rxfevent*.
Offset + A	6–0	Reserved, must be cleared.
Offset + C	15–0 Payload checksum	Internet payload checksum. This is an accelerator option. It is the one's complement sum of the payload section of the IP frame. The sum is calculated over all data following the IP header until the end of the IP payload. This field is valid only when the L field is set.
Offset + E	15–11 Header length	Header length. This is an accelerator option. This field is written by the uDMA. This field is the sum of 32-bit words found within the IP and its following protocol headers. If an IP datagram with an unknown protocol is found, then the value is the length of the IP header. If no IP frame or an erroneous IP header is found, the value is 0. The following values are minimum values if no header options exist in the respective headers: • ICMP/IP: 6 (5 IP header, 1 ICMP header) • UDP/IP: 7 (5 IP header, 2 UDP header) • TCP/IP: 10 (5 IP header, 5 TCP header) This field is only valid if the L field is set.
Offset + E	10–8	Reserved, must be cleared.

Table continues on the next page...

Table 41-32. Receive buffer descriptor field definitions (continued)

Word	Field	Description
Offset + E	7–0 Protocol type	Protocol type. This is an accelerator option. The 8-bit protocol field found within the IP header of the frame. It is valid only when ICE is cleared. This field is valid only when the L field is set.
Offset + 10	15–0	Reserved, must be cleared.
Offset + 12	15 BDU	Last buffer descriptor update done. Indicates that the last BD data has been updated by uDMA. This field is written by the user (=0) and uDMA (=1).
Offset + 12	14–0	Reserved, must be cleared.
Offset + 14	15–0	This value is written by the uDMA. It is only valid if the L field is set.
Offset + 16	1588 timestamp	
Offset + 18 – Offset + 1E	15–0	Reserved, must be cleared.

1. The receive buffer pointer, containing the address of the associated data buffer, must always be evenly divisible by 64. The buffer must reside in memory external to the MAC. The Ethernet controller never modifies this value.

41.3.14.2 Enhanced transmit buffer descriptor

Table 41-33. Enhanced transmit buffer descriptor (TxBD)

	Byte 1								Byte 0							
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Offset + 0	Data length															
Offset + 2	R	TO1	W	TO2	L	TC	—	—	—	—	—	—	—	—	—	—
Offset + 4	Tx Data Buffer Pointer – low halfword															
Offset + 6	Tx Data Buffer Pointer – high halfword															
Offset + 8	TXE	—	UE	EE	FE	LCE	OE	TSE	—	—	—	—	—	—	—	—
Offset + A	—	INT	TS	PINS	IINS	—	—	—	—	—	—	—	—	—	—	—
Offset + C	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + E	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 10	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 12	BDU	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 14	1588 timestamp – low halfword															
Offset + 16	1588 timestamp – high halfword															
Offset + 18	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 1A	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 1C	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—
Offset + 1E	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—	—

Table 41-34. Enhanced transmit buffer descriptor field definitions

Word	Field	Description
Offset + 0	15–0 Data Length	Data length, written by user. Data length is the number of octets the MAC should transmit from this BD's data buffer. It is never modified by the MAC.
Offset + 2	15 R	Ready. Written by the MAC and you. 0 The data buffer associated with this BD is not ready for transmission. You are free to manipulate this BD or its associated data buffer. The MAC clears this field after the buffer has been transmitted or after an error condition is encountered. 1 The data buffer, prepared for transmission by you, has not been transmitted or currently transmits. You may write no fields of this BD after this field is set.
Offset + 2	14 TO1	Transmit software ownership. This field is reserved for software use. This read/write field is not modified by hardware and its value does not affect hardware.
Offset + 2	13 W	Wrap. Written by user. 0 The next buffer descriptor is found in the consecutive location 1 The next buffer descriptor is found at the location defined in ETDSR.
Offset + 2	12 TO2	Transmit software ownership. This field is reserved for use by software. This read/write field is not modified by hardware and its value does not affect hardware.
Offset + 2	11 L	Last in frame. Written by user. 0 The buffer is not the last in the transmit frame 1 The buffer is the last in the transmit frame
Offset + 2	10 TC	Transmit CRC. Written by user, and valid only when L is set. 0 End transmission immediately after the last data byte 1 Transmit the CRC sequence after the last data byte This field is valid only when the L field is set.
Offset + 2	9 ABC	Append bad CRC. **Note:** This field is not supported by the uDMA and is ignored.
Offset + 2	8–0	Reserved, must be cleared.
Offset + 4	15–0 Data buffer pointer low	Tx data buffer pointer, low halfword
Offset + 6	15–0 Data buffer pointer high	Tx data buffer pointer, high halfword. The buffer must reside in memory external to the MAC. This value is never modified by the Ethernet controller. **NOTE:** For optimal performance, make the transmit buffer pointer evenly divisible by 64.
Offset + 8	15 TXE	Transmit error occurred. This field is written by the uDMA. This field indicates that there was a transmit error of some sort reported with the frame. Effectively this field is an OR of the other error fields including UE, EE, FE, LCE, OE, and TSE. This field is valid only when the L field is set.
Offset + 8	14	Reserved, must be cleared.
Offset + 8	13 UE	Underflow error. This field is written by the uDMA. This field indicates that the MAC reported an underflow error on transmit. This field is valid only when the L field is set.

Table continues on the next page...

Table 41-34. Enhanced transmit buffer descriptor field definitions (continued)

Word	Field	Description
Offset + 8	12 EE	Excess Collision error. This field is written by the uDMA. This field indicates that the MAC reported an excess collision error on transmit. This field is valid only when the L field is set.
Offset + 8	11 FE	Frame with error. This field is written by the uDMA. This field indicates that the MAC reported that the uDMA reported an error when providing the packet. This field is valid only when the L field is set.
Offset + 8	10 LCE	Late collision error. This field is written by the uDMA. This field indicates that the MAC reported that there was a Late Collision on transmit. This field is valid only when the L field is set.
Offset + 8	9 OE	Overflow error. This field is written by the uDMA. This field indicates that the MAC reported that there was a FIFO overflow condition on transmit. This field is only valid when the L field is set.
Offset + 8	8 TSE	Timestamp error. This field is written by the uDMA. This field indicates that the MAC reported a different frame type then a timestamp frame. This field is valid only when the L field is set.
Offset + 8	7–0	Reserved, must be cleared.
Offset + A	15	Reserved, must be cleared.
Offset + A	14 INT	Generate interrupt flags. This field is written by the user. This field is valid regardless of the L field and must be the same for all EBD for a given frame. The uDMA does not update this value.
Offset + A	13 TS	Timestamp. This field is written by the user. This indicates that the uDMA is to generate a timestamp frame to the MAC. This field is valid regardless of the L field and must be the same for all EBD for the given frame. The uDMA does not update this value.
Offset + A	12 PINS	Insert protocol specific checksum. This field is written by the user. If set, the MAC's IP accelerator calculates the protocol checksum and overwrites the corresponding checksum field with the calculated value. The checksum field must be cleared by the application generating the frame. The uDMA does not update this value. This field is valid regardless of the L field and must be the same for all EBD for a given frame.
Offset + A	11 IINS	Insert IP header checksum. This field is written by the user. If set, the MAC's IP accelerator calculates the IP header checksum and overwrites the corresponding header field with the calculated value. The checksum field must be cleared by the application generating the frame. The uDMA does not update this value. This field is valid regardless of the L field and must be the same for all EBD for a given frame.
Offset + A	10–0	Reserved, must be cleared.
Offset + C	15–0	Reserved, must be cleared.
Offset + E	15–0	Reserved, must be cleared.
Offset + 10	15–0	Reserved, must be cleared.
Offset + 12	15 BDU	Last buffer descriptor update done. Indicates that the last BD data has been updated by uDMA. This field is written by the user (=0) and uDMA (=1).
Offset + 12	14–0	Reserved, must be cleared.
Offset + 14	15–0	This value is written by the uDMA . It is valid only when the L field is set.
Offset + 16	1588 timestamp	
Offset + 18–Offset + 1E	15–0	Reserved, must be cleared.

41.3.15 Client FIFO application interface

The FIFO interface is completely asynchronous from the Ethernet line, and the transmit and receive interface can operate at a different clock rate.

All transfers to/from the user application are handled independently of the core operation, and the core provides a simple interface to user applications based on a two-signal handshake.

41.3.15.1 Data structure description

The data structure defined in the following tables for the FIFO interface must be respected to ensure proper data transmission on the Ethernet line. Byte 0 is sent to and received from the line first.

Table 41-35. FIFO interface data structure

	63 56	55 48	47 40	39 32	31 24	23 16	15 8	7 0
Word 0	Byte 7	Byte 6	Byte 5	Byte 4	Byte 3	Byte 2	Byte 1	Byte 0
Word 1	Byte 15	Byte 14	Byte 13	Byte 12	Byte 11	Byte 10	Byte 9	Byte 8
...	...							

The size of a frame on the FIFO interface may not be a modulo of 64-bit.

The user application may not care about the Ethernet frame formats in full detail. It needs to provide and receive an Ethernet frame with the following structure:

- Ethernet MAC destination address
- Ethernet MAC source address
- Optional 802.1q VLAN tag (VLAN type and info field)
- Ethernet length/type field
- Payload

Frames on the FIFO interface do not contain preamble and SFD fields, which are inserted and discarded by the MAC on transmit and receive, respectively.

- On receive, CRC and frame padding can be stripped or passed through transparently.
- On transmit, padding and CRC can be provided by the user application, or appended automatically by the MAC independently for each frame. No size restrictions apply.

Note

On transmit, if ENET*n*_TCR[ADDINS] is set, bytes 6–11 of each frame can be set to any value, since the MAC overwrites the bytes with the MAC address programmed in the ENET*n*_PAUR and ENET*n*_PALR registers.

Table 41-36. FIFO interface frame format

Byte number	Field
0–5	Destination MAC address
6–11	Source MAC address
12–13	Length/type field
14–N	Payload data

VLAN-tagged frames are supported on both transmit and receive, and implement additional information (VLAN type and info).

Table 41-37. FIFO interface VLAN frame format

Byte number	Field
0–5	Destination MAC address
6–11	Source MAC address
12–15	VLAN tag and info
16–17	Length/type field
18–N	Payload data

Note

The standard defines that the LSB of the MAC address is sent/received first, while for all the other header fields — in other words, length/type, VLAN tag, VLAN info, and pause quanta — the MSB is sent/received first.

41.3.15.2 Data structure examples

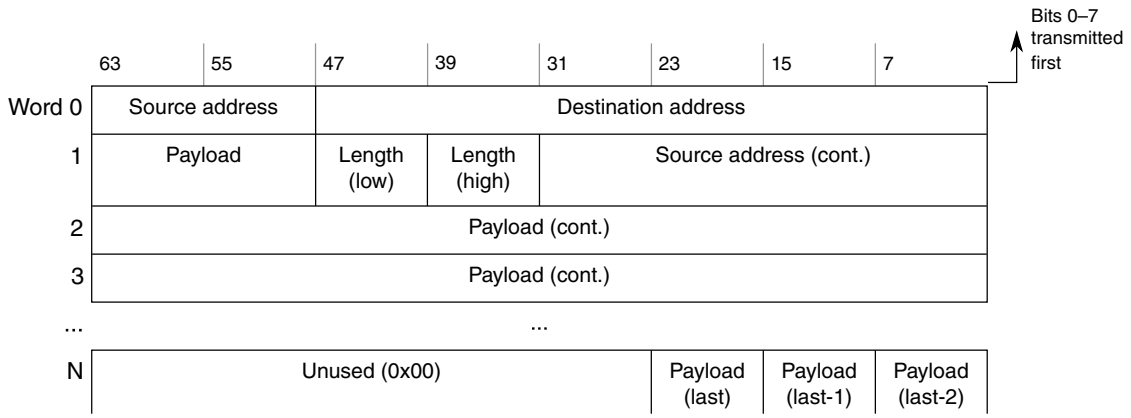


Figure 41-14. Normal Ethernet frame 64-bit mapping example

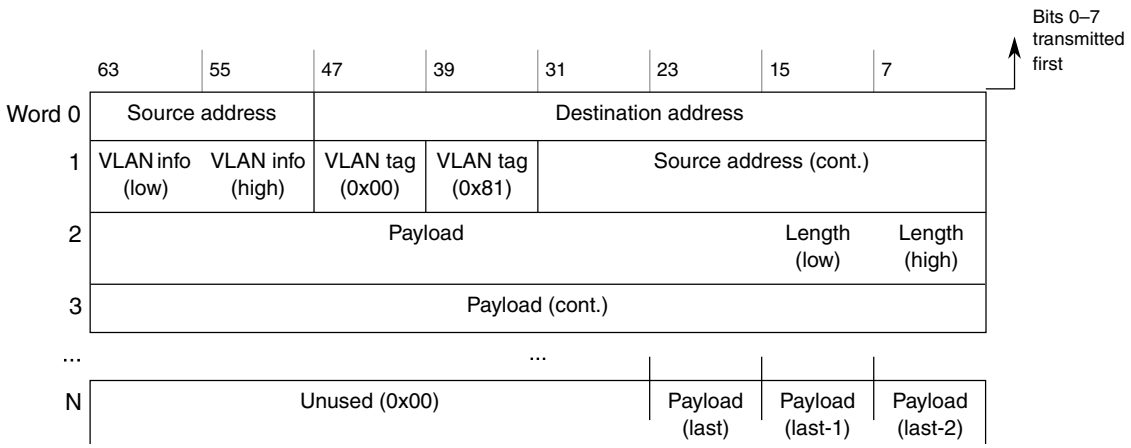


Figure 41-15. VLAN-tagged frame 64-bit mapping example

If CRC forwarding is enabled (CRCFWD = 0), the last four valid octets of the frame contain the FCS field. The non-significant bytes of the last word can have any value.

41.3.15.3 Frame status

A MAC layer status word and an accelerator status word is available in the receive buffer descriptor.

See Enhanced buffer descriptors for details.

The status is available with each frame with the last data of the frame.

If the frame status contains a MAC layer error (for example, CRC or length error), RxBD[ME] is also set with the last data of the frame.

41.3.16 FIFO protection

The following sections describe the FIFO protection mechanisms.

41.3.16.1 Transmit FIFO underflow

During a frame transfer, when the transmit FIFO reaches the almost empty threshold with no end-of-frame indication stored in the FIFO, the MAC logic:

- Stops reading data from the FIFO
- Asserts the MII error signal (MII_TXER) (1 in Figure 41-16) to indicate that the fragment already transferred is not valid
- Deasserts the MII transmit enable signal (MII_TXEN) to terminate the frame transfer (2)

After an underflow, when the application completes the frame transfer (3), the MAC transmit logic discards any new data available in the FIFO until the end of packet is reached (4) and sets the enhanced TxBD[UE] field.

The MAC starts to transfer data on the MII interface when the application sends a new frame with a start of frame indication (5).

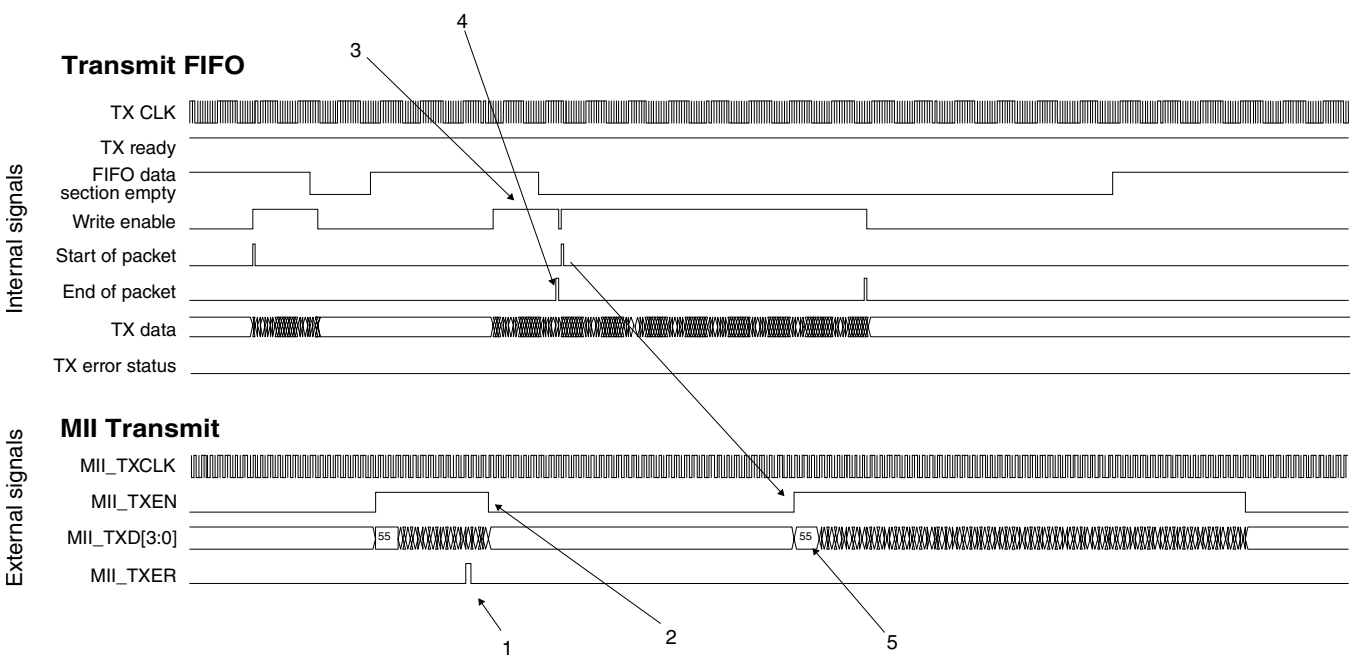


Figure 41-16. Transmit FIFO underflow protection

41.3.16.2 Transmit FIFO overflow

On the transmit path, when the FIFO reaches the programmable almost full threshold, the internal MAC ready signal is deasserted. The application should stop sending new data.

However, if the application keeps sending data, the transmit FIFO overflows, corrupting contents that were previously stored. The core logic sets the enhanced TxBD[OE] field for the next frame transmitted to indicate this overflow occurence.

Note

Overflow is a fatal error and must be addressed by resetting the core or clearing ENET*n*_ECR[ETHER_EN], to clear the FIFOs and prepare for normal operation again.

41.3.16.3 Receive FIFO overflow

During a frame reception, if the client application is not able to receive data (1), the MAC receive control truncates the incoming frame when the FIFO reaches the programmable almost-full threshold to avoid an overflow.

The frame is subsequently received on the FIFO interface with an error indication (enhanced RxBD[ME] field set together with receive end-of-packet) (2) with the truncation error status field set (3).

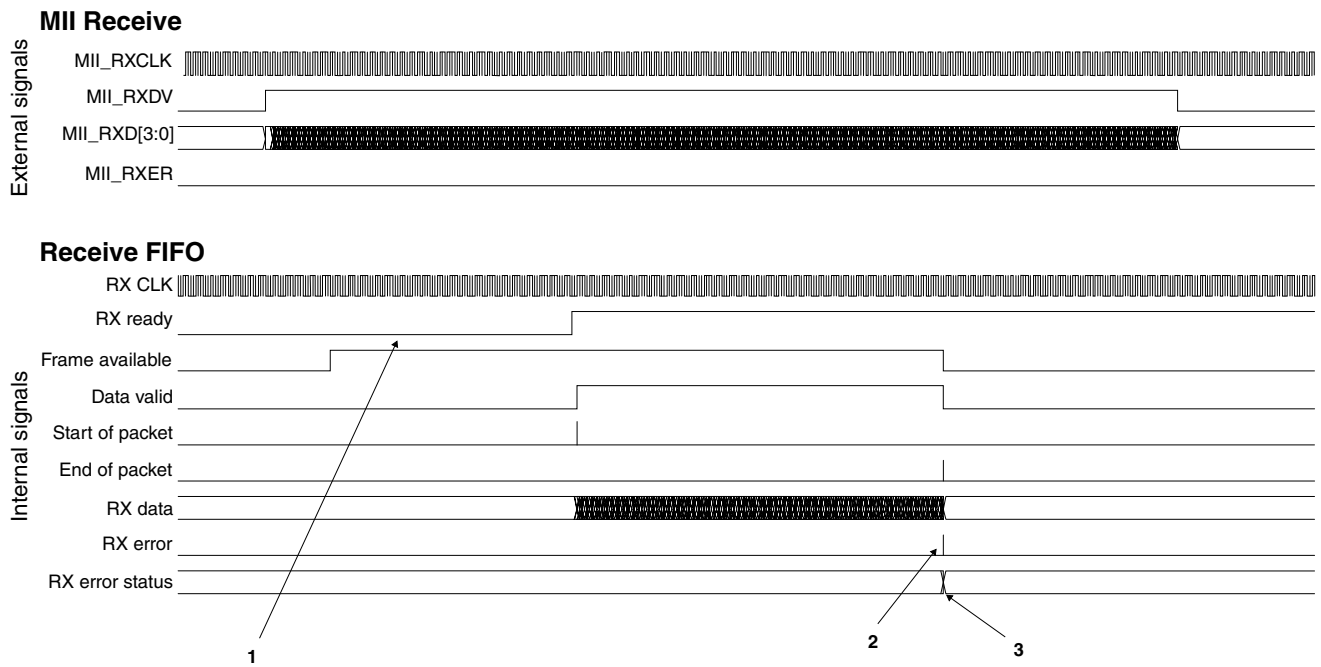


Figure 41-17. Receive FIFO overflow protection

41.3.17 PHY management interface

The MDIO interface is a two-wire management interface. The MDIO management interface implements a standardized method to access the PHY device management registers.

The core implements a master MDIO interface, which can be connected to up to 32 PHY devices.

41.3.17.1 MDIO clause 22 frame format

The core MDIO master controller communicates with the slave (PHY device) using frames that are defined in the following table.

A complete frame has a length of 64 bits made up of an optional 32-bit preamble, 14-bit command, 2-bit bus direction change, and 16-bit data. Each bit is transferred on the rising edge of the MDIO clock (MDC signal). The MDIO data signal is tri-stated between frames.

The core PHY management interface supports the standard MDIO specification (IEEE 802.3 Clause 22).

Table 41-38. MDIO clause 22 frame structure

ST	OP	PHYADR	REGADR	TA	DATA

Table 41-39. MDIO frame field descriptions

Field	Description
ST (2 bits)	Start indication field, programmed with ENET*n*_MMFR[ST] and equal to 01 for Standard MDIO (Clause 22).
OP (2 bits)	Opcode defines type of operation. Programmed with ENET*n*_MMFR[OP]. 01 Write operation 10 Read operation
PHYADR (5 bits)	Five-bit PHY device address, programmed with ENET*n*_MMFR[PA]. Up to 32 devices can be addressed.
REGADR (5 bits)	Five-bit register address, programmed with ENET*n*_MMFR[RA]. Each PHY can implement up to 32 registers.
TA (2 bits)	Turnaround time, programmed with ENET*n*_MMFR[TA]. Two bit-times are reserved for read operations to switch the data bus from write to read. The PHY device presents its register contents in the data phase and drives the bus from the second bit of the turnaround phase.
Data	Data, set by ENET*n*_MMFR[DATA]. Written to or read from the PHY

Table 41-39. MDIO frame field descriptions

Field	Description
(16 bits)	

41.3.17.2 MDIO clause 45 frame format

The extended MDIO frame structure defined in IEEE 802.3 Clause 45 introduces indirect addressing. First, a write transaction to an address register is done, followed by a write or read transaction which will put the 16-bit data in the register or retrieve the register contents respectively. A preamble of 32 bits of logical ones is sent prior to every transaction. The MDIO data signal is tri-stated between frames.

The extended MDIO defines four transactions, which are determined by the two-bit opcode field.

Table 41-40. MDIO clause 45 frame structure

ST	OP	PRTAD	DEVAD	TA	ADDR/DATA

All bits are transmitted from left to right (Preamble bits first) and all fields have their Most-Significant bit sent first (leftmost in above table). The complete frame has a length of 64 bits (32-bit preamble, 14-bit command, 2-bit bus direction change, 16-bit data). Each bit is transferred with the rising edge of the MDIO clock (MDC).

The fields and transactions are summarized in the following tables.

Table 41-41. MDIO clause 45 frame field descriptions

Field	Description
ST	Start indication. Indicates the end of the preamble and start of the frame. This value is 00 for extended MDIO (Clause 45) frames.
OP	Opcode defines if a read or write operation is performed and is programmed with ENET*n*_MMFR[OP]. See Table 41-42 for more information. 00 Address write 01 Write operation 10 Read inc. operation 11 Read operation
PRTAD	The port address specifies a MDIO port. Each Port can have up to 32 devices which each can have a separate set of registers.
DEVAD	Device address. Up to 32 devices can be addressed (within a port).
TA	Turnaround time, programmed with ENET*n*_MMFR[TA]. Two bit-times are reserved for read operations to switch the data bus from write to read. The PHY device presents its register contents in the data phase and drives the bus from the second bit of the turnaround phase.

Table continues on the next page...

Table 41-41. MDIO clause 45 frame field descriptions (continued)

Field	Description
ADDR/DATA	16-bit address (for address write) or data, set by ENET*n*_MMFR[DATA], written to or read from the PHY.

Table 41-42. MDIO Clause 45 Transactions

Transaction Type	Description
Address	A write transaction to the internal address register of the device/port. The data section of the frame contains the value to be stored in the device's internal address "pointer" register for further transactions.
Write	Data write to a register. The 16 bit data will be written to the register identified by the device-internal address.
Read	Data is read from the register identified by the device-internal address.
Read inc.	Read with address postincrement. The register identified by the device-internal address is read. After this, the device-internal address is incremented. If the address register is all '1' (0xFFFF) no increment is done (i.e. increment does not wrap around).

41.3.17.3 MDIO clock generation

The MDC clock is generated from the internal bus clock (i.e., IPS bus clock) divided by the value programmed in ENET*n*_MSCR[MII_SPEED].

41.3.17.4 MDIO operation

To perform an MDIO access, set the MDIO command register (ENET*n*_MMFR) according to the description provided in MII Management Frame Register (ENETn_MMFR).

To check when the programmed access completes, read the ENET*n*_EIR[MII] field.

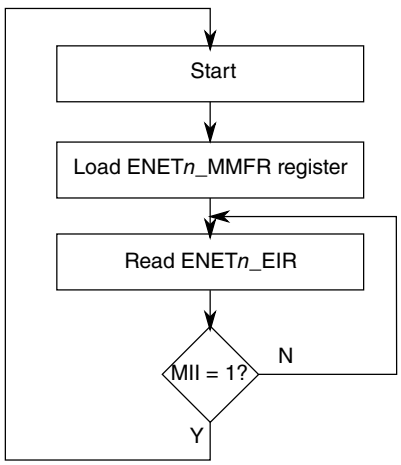


Figure 41-18. MDIO access overview

41.3.18 Ethernet interfaces

The following Ethernet interfaces are implemented:

- Fast Ethernet MII 10/100 (Media Independent Interface)
- RMII 10/100 using interface converters/gaskets

The following table shows how to configure ENET registers to select each interface.

Mode	RCR[RMII_10T]	RCR[RMII_MODE]
MII - 10 Mbit/s	—	0
MII - 100 Mbit/s	—	0
RMII - 10 Mbit/s	1	1
RMII - 100 Mbit/s	0	1

41.3.18.1 RMII interface

In RMII receive mode, for normal reception following assertion of CRS_DV, RXD[1:0] is 00 until the receiver determines that the receive event has a proper start-of-stream delimiter (SSD).

The preamble appears (RXD[1:0]=01) and the MACs begin capturing data following detection of SFD.

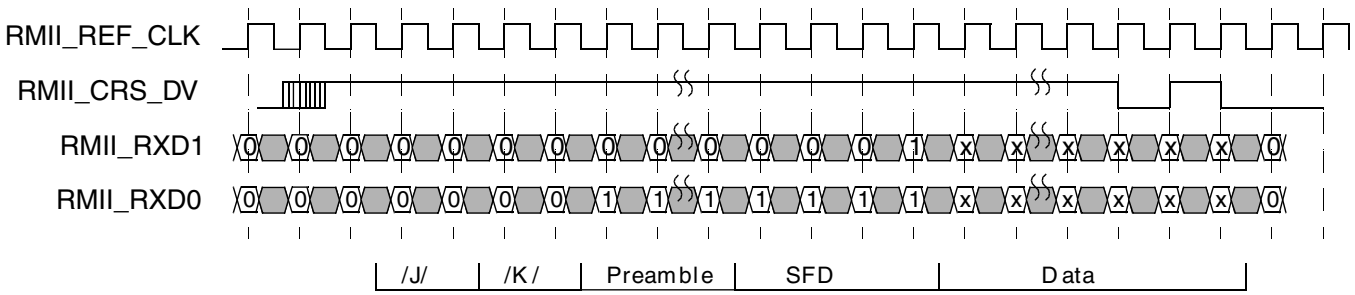


Figure 41-19. RMII receive operation

If a false carrier is detected (bad SSD), then RXD[1:0] is 10 until the end of the receive event. This is a unique pattern since a false carrier can only occur at the beginning of a packet where the preamble is decoded (RXD[1:0] = 01).

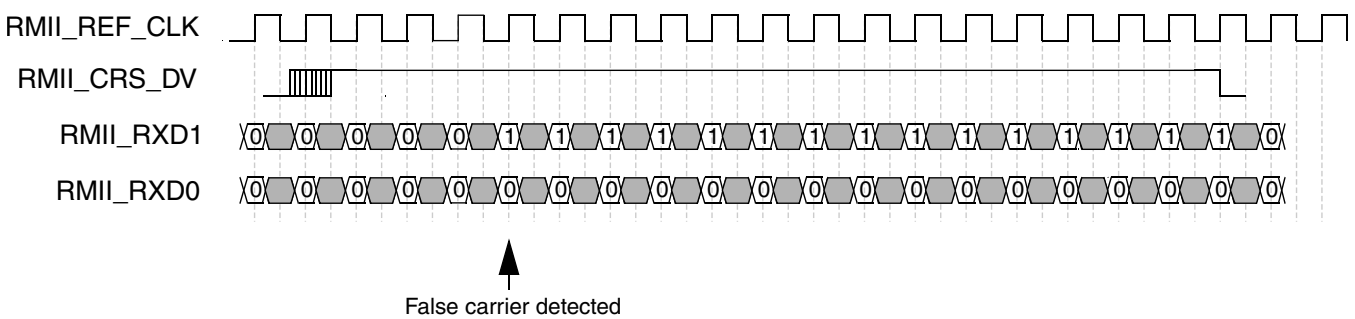


Figure 41-20. RMII receive operation with false carrier

In RMII transmit mode, TXD[1:0] provides valid data for each REF_CLK period while TXEN is asserted.

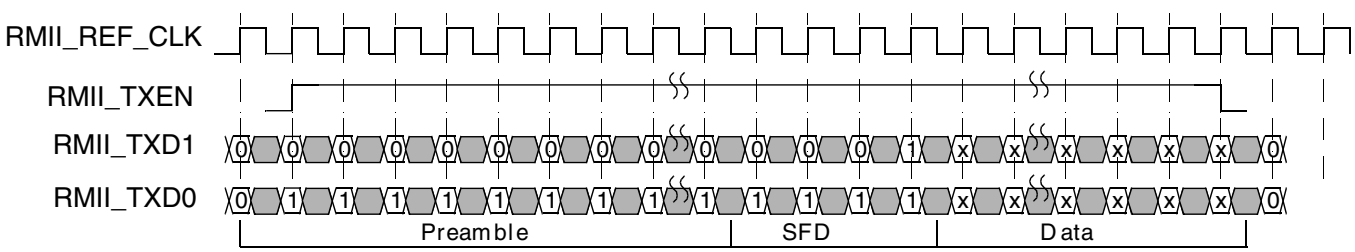


Figure 41-21. RMII transmit operation

41.3.18.2 MII Interface — transmit

On transmit, all data transfers are synchronous to MII_TXCLK rising edge. The MII data enable signal MII_TXEN is asserted to indicate the start of a new frame, and remains asserted until the last byte of the frame is present on the MII_TXD[3:0] bus.

Between frames, MII_TXEN remains deasserted.

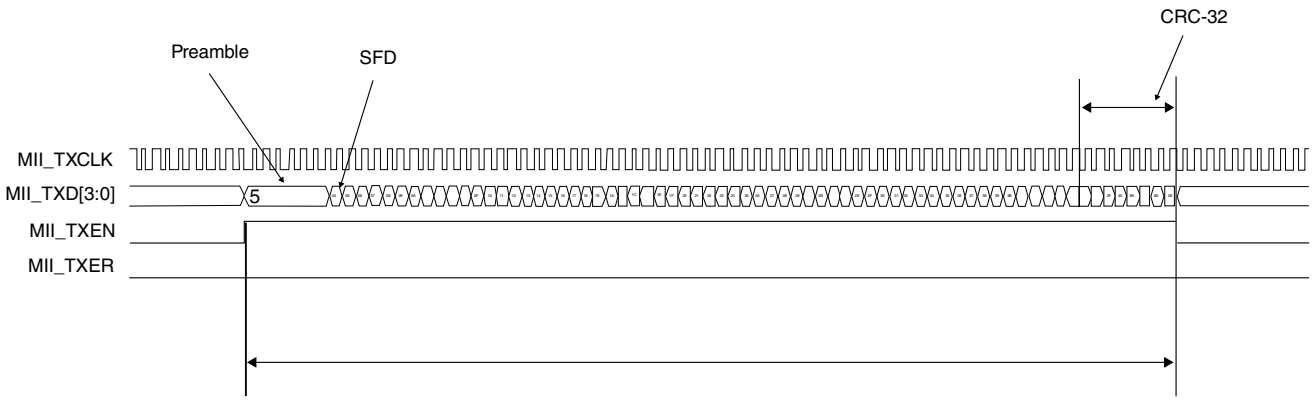


Figure 41-22. MII transmit operation

If a frame is received on the FIFO interface with an error (for example, RxBD[ME] set) the frame is subsequently transmitted with the MII_TXER error signal for one clock cycle at any time during the packet transfer.

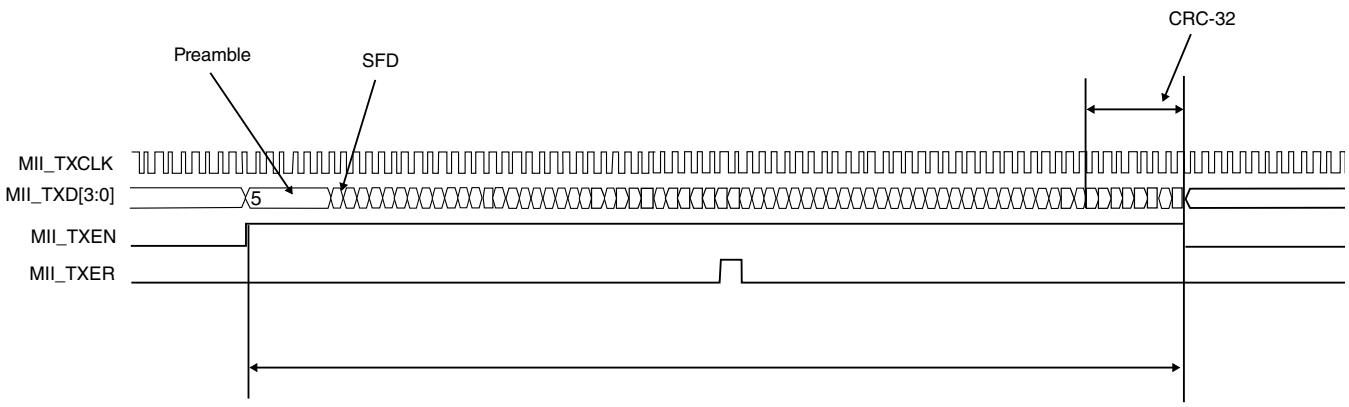


Figure 41-23. MII transmit operation — errored frame

41.3.18.2.1 Transmit with collision — half-duplex

When a collision is detected during a frame transmission (MII_COL asserted), the MAC stops the current transmission, sends a 32-bit jam pattern, and re-transmits the current frame.

(See Collision detection in half-duplex mode for details)

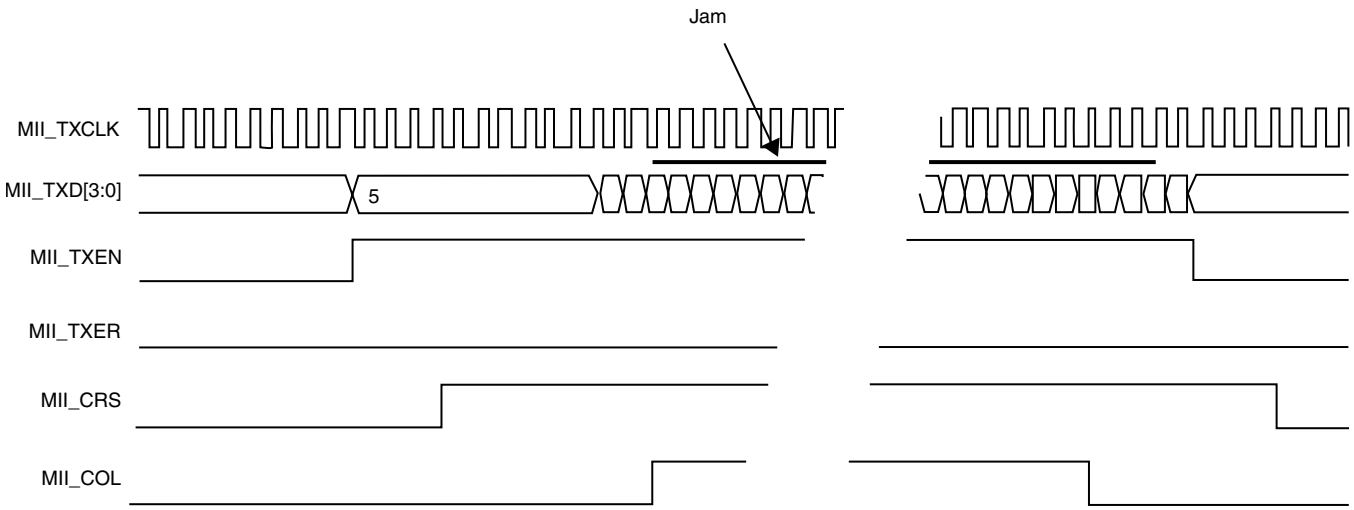


Figure 41-24. MII transmit operation — transmission with collision

41.3.18.3 MII interface — receive

On receive, all signals are sampled on the MII_RXCLK rising edge. The MII data enable signal, MII_RXDV, is asserted by the PHY to indicate the start of a new frame and remains asserted until the last byte of the frame is present on MII_RXD[3:0] bus.

Between frames, MII_RXDV remains deasserted.

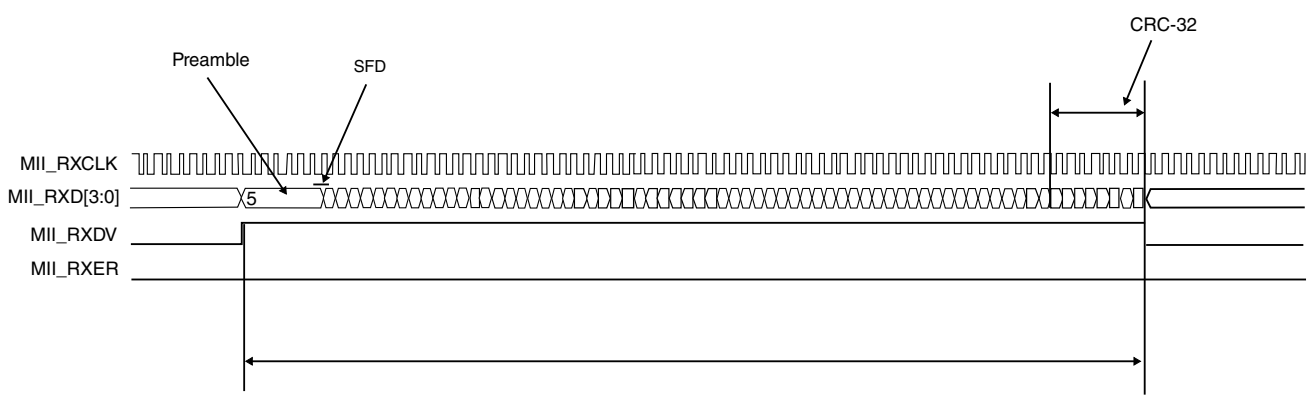


Figure 41-25. MII receive operation

If the PHY detects an error on the frame received from the line, the PHY asserts the MII error signal, MII_RXER, for at least one clock cycle at any time during the packet transfer.

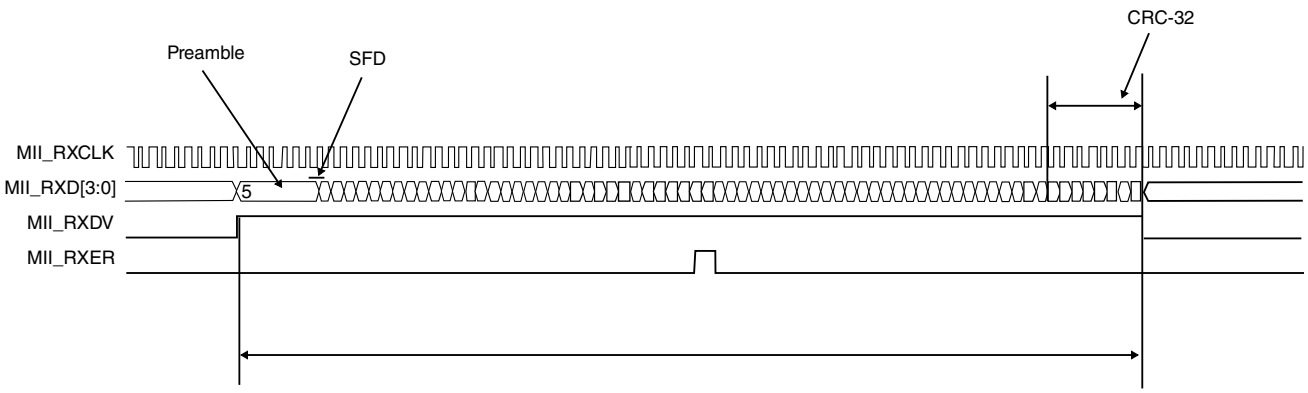


Figure 41-26. MII receive operation — errored frame

A frame received on the MII interface with a PHY error indication is subsequently transferred on the FIFO interface with RxBD[ME] set.

41.3.19 Interrupt coalescence

The purpose of the interrupt coalescing is to reduce the number of interrupts generated by the MAC so as to reduce the CPU loading.

To facilitate this interrupt coalescing, these registers are available with the same control and configuration fields.

- Transmit Interrupt Coalescing Register (TXICn)
- Receive Interrupt Coalescing Register (RXICn)

When coalescing is enabled by asserting the corresponding ICEN field and such interrupt is also enabled by the corresponding interrupt mask of the EIMR register, the MAC generates an interrupt when the threshold number of frames is reached (defined by ICFT) or when the threshold timer expires (defined by ICTT).

When coalescing is disabled by de-asserting ICEN, but interrupt is enabled by the corresponding interrupt mask of the EIMR register, the MAC generates an interrupt as they are received without using coalescing. Interrupt coalescing is done for each transmit and receive queue/class independently.

41.3.19.1 Interrupt coalescence setup

Interrupt coalescence supports both legacy and enhanced BDs. The following guidelines are recommended when setting up interrupt coalescence.

- When the MAC is configured for enhanced (IEEE 1588) mode, that is, enhanced BDs:
  - Set the INT bit in the enhanced received buffer descriptor to one.
  - Set the INT bit in the enhanced transmit buffer descriptor(s) to one.
- Clear the TXB and RXB fields in the EIMR register.

41.3.19.2 Updating the frame count threshold on-the-fly

To update the ICFT field in the RXIC and TXIC registers:

1. Disable interrupt coalescence by clearing the appropriate ICEN field. This will allow the internal interrupt coalescence counter to reset to zero.

NOTE

When disabling interrupt coalescence, if an interrupt event is pending, that is, the interrupt counter is not zero, then an interrupt will occur.

2. Write the new threshold value to the ICFT field.
3. Set ICEN to one.

NOTE

The ICFT field can be updated on-the-fly without disabling the ICEN field. The hardware interrupt will continue and there is a possibility that an interrupt will occur depending on the state of the hardware counter and the previous ICFT value.

41.3.19.3 Updating the timer threshold on-the-fly

To update the ICTT field in the RXIC and TXIC registers:

1. Disable interrupt coalescence by clearing the appropriate ICEN field. This will allow the internal interrupt coalescence counter to reset to zero.

NOTE

When disabling interrupt coalescence, if an interrupt event is pending, that is, the interrupt counter is not zero, then an interrupt will occur.

2. Write the new timer value to the ICTT field.
3. Set ICEN to one.

41.3.20 Clocks

The table found here describes the clock sources for ENET. Please see the chip-specific clocking section for clock setting, configuration and gating information.

Table 41-43. Clocks

Clock name	Description
ipg_clk	Module clock
ipg_clk_rmii	Module clock for RMII
ipg_clk_s	Peripheral access clock
ipg_clk_time (ts_clk)	Peripheral clock
txc_sampling_clk	Transmit sampling clock

41.4 External Signals

The table found here describes the external signals of ENET.

Table 41-44. ENET External Signals

Signal	Description	Mode	Direction
1588_EVENT_IN	Capture/compare block input/ output event bus signal. When configured for capture and a rising edge is detected, the current timer value is latched and transferred into the corresponding ENET_TCCRn register for inspection by software. When configured for compare, the corresponding signal 1588_EVENT is asserted for one cycle when the timer reaches the compare value programmed in register ENET_TCCRn. An interrupt or DMA request can be triggered if the corresponding bit in ENET_TCSRn[TIE] or ENET_TCSRn[TDRE] is set.	MII/RMII	I
1588_EVENT_OUT	Capture/compare block input/ output event bus signal. When configured for capture and a rising edge is detected, the current timer value is latched and transferred into the	MII/RMII	O

Table continues on the next page...

Table 41-44. ENET External Signals (continued)

Signal	Description	Mode	Direction
	corresponding ENET_TCCRn register for inspection by software. When configured for compare, the corresponding signal 1588_EVENT is asserted for one cycle when the timer reaches the compare value programmed in register ENET_TCCRn. An interrupt or DMA request can be triggered if the corresponding bit in ENET_TCSRn[TIE] or ENET_TCSRn[TDRE] is set. NOTE: ENET_1588_EVENT0_OUT has a programmable output width, see IOMUXC_GPR[CLK_STRETCH], delayed one clock cycle in relation to all other EVENTx_OUT signals.		
COL	Asserted upon detection of a collision and remains asserted while the collision persists. This signal is not defined for full duplex mode.	MII	I
CRS	Carrier sense. When asserted, indicates transmit or receive medium is not idle. In RMII mode, this signal is present on the CRS_DV pin.	MII	I
MDC	Output clock provides a timing reference to the PHY for data transfers on the MDIO signal.	MII/RMII	O
MDIO	Transfers control information between the external PHY and the media access controller. Data is synchronous to MDC. This signal is an input after reset.	MII/RMII	IO
RX_CLK	In MII mode, provides a timing reference for RX_EN, RX_DATA[3:0], and RX_ER.	MII	I
RX_ER	When asserted with RXDV, indicates the PHY detects an error in the current frame.	MII/RMII	I
RX_EN	Asserting this input indicates the PHY has valid nibbles present on the MII. RX_EN must remain asserted from the first recovered nibble of the frame through to the last	MII/RMII	I

Table continues on the next page...

Table 41-44. ENET External Signals (continued)

Signal	Description	Mode	Direction
	nibble. Asserting RX_EN must start no later than the SFD and exclude any EOF. In RMII mode, this pin also generates the CRS signal.		
TX_CLK	Input clock, which provides a timing reference for TX_EN, TX_DATA[3:0], and TX_ER.	MII	O
TX_ER	When asserted for one or more clock cycles while TXEN is also asserted, PHY sends one or more illegal symbols.	MII/RMII	O
TX_EN	Indicates when valid nibbles are present on the MII. This signal is asserted with the first nibble of a preamble and is deasserted before the first TX_CLK following the final nibble of the frame.	MII/RMII	I
RDATA	Contains the Ethernet input data transferred from the PHY to the media-access controller when RX_EN is asserted.	MII/RMII	I
TDATA	Serial output Ethernet data. Only valid during TX_EN assertion.	MII/RMII	O
REF_CLK1	In RMII mode, this signal is the reference clock for receive, transmit, and the control interface.	RMII	IO

41.5 Memory map/register definition

ENET registers must be read or written with 32-bit accesses. Non-32 bit accesses will terminate with an error.

Reserved bits should be written with 0 and ignored on read. Unused registers read zero and a write has no effect.

This table shows Ethernet registers organization.

Table 41-45. Register map summary

Offset Address	Section	Description
0x0000 – 0x01FF	Configuration	Core control and status registers

Table continues on the next page...

Table 41-45. Register map summary (continued)

Offset Address	Section	Description
0x0200 – 0x03FF	Statistics counters	MIB and Remote Network Monitoring (RFC 2819) registers
0x0400 – 0x0430	1588 control	1588 adjustable timer (TSM) and 1588 frame control
0x0600 – 0x07FC	Capture/Compare block	Registers for the Capture/Compare block

41.5.1 ENET register descriptions

41.5.1.1 ENET memory map

ENET base address: 402D_8000h

ENET2 base address: 402D_4000h

Offset	Register	Width (In bits)	Access	Reset value
4h	Interrupt Event Register (EIR)	32	WOZ	0000_0000h
8h	Interrupt Mask Register (EIMR)	32	RW	0000_0000h
10h	Receive Descriptor Active Register - Ring 0 (RDAR)	32	RW	0000_0000h
14h	Transmit Descriptor Active Register - Ring 0 (TDAR)	32	RW	0000_0000h
24h	Ethernet Control Register (ECR)	32	RW	7000_0000h
40h	MII Management Frame Register (MMFR)	32	RW	0000_0000h
44h	MII Speed Control Register (MSCR)	32	RW	0000_0000h
64h	MIB Control Register (MIBC)	32	RW	C000_0000h
84h	Receive Control Register (RCR)	32	RW	05EE_0001h
C4h	Transmit Control Register (TCR)	32	RW	0000_0000h
E4h	Physical Address Lower Register (PALR)	32	RW	0000_0000h
E8h	Physical Address Upper Register (PAUR)	32	RW	0000_8808h
ECh	Opcode/Pause Duration Register (OPD)	32	RW	0001_0000h
F0h	Transmit Interrupt Coalescing Register (TXIC0)	32	RW	0000_0000h
100h	Receive Interrupt Coalescing Register (RXIC0)	32	RW	0000_0000h
118h	Descriptor Individual Upper Address Register (IAUR)	32	RW	0000_0000h
11Ch	Descriptor Individual Lower Address Register (IALR)	32	RW	0000_0000h
120h	Descriptor Group Upper Address Register (GAUR)	32	RW	0000_0000h
124h	Descriptor Group Lower Address Register (GALR)	32	RW	0000_0000h
144h	Transmit FIFO Watermark Register (TFWR)	32	RW	0000_0000h
180h	Receive Descriptor Ring 0 Start Register (RDSR)	32	RW	0000_0000h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
184h	Transmit Buffer Descriptor Ring 0 Start Register (TDSR)	32	RW	0000_0000h
188h	Maximum Receive Buffer Size Register - Ring 0 (MRBR)	32	RW	0000_0000h
190h	Receive FIFO Section Full Threshold (RSFL)	32	RW	0000_0000h
194h	Receive FIFO Section Empty Threshold (RSEM)	32	RW	0000_0000h
198h	Receive FIFO Almost Empty Threshold (RAEM)	32	RW	0000_0004h
19Ch	Receive FIFO Almost Full Threshold (RAFL)	32	RW	0000_0004h
1A0h	Transmit FIFO Section Empty Threshold (TSEM)	32	RW	0000_0000h
1A4h	Transmit FIFO Almost Empty Threshold (TAEM)	32	RW	0000_0004h
1A8h	Transmit FIFO Almost Full Threshold (TAFL)	32	RW	0000_0008h
1ACh	Transmit Inter-Packet Gap (TIPG)	32	RW	0000_000Ch
1B0h	Frame Truncation Length (FTRL)	32	RW	0000_07FFh
1C0h	Transmit Accelerator Function Configuration (TACC)	32	RW	0000_0000h
1C4h	Receive Accelerator Function Configuration (RACC)	32	RW	0000_0000h
204h	Tx Packet Count Statistic Register (RMON_T_PACKETS)	32	RO	0000_0000h
208h	Tx Broadcast Packets Statistic Register (RMON_T_BC_PKT)	32	RO	0000_0000h
20Ch	Tx Multicast Packets Statistic Register (RMON_T_MC_PKT)	32	RO	0000_0000h
210h	Tx Packets with CRC/Align Error Statistic Register (RMON_T_CRC_ALIGN)	32	RO	0000_0000h
214h	Tx Packets Less Than Bytes and Good CRC Statistic Register (RMON_T_UNDERSIZE)	32	RO	0000_0000h
218h	Tx Packets GT MAX_FL bytes and Good CRC Statistic Register (RMON_T_OVERSIZE)	32	RO	0000_0000h
21Ch	Tx Packets Less Than 64 Bytes and Bad CRC Statistic Register (RMON_T_FRAG)	32	RO	0000_0000h
220h	Tx Packets Greater Than MAX_FL bytes and Bad CRC Statistic Register (RMON_T_JAB)	32	RO	0000_0000h
224h	Tx Collision Count Statistic Register (RMON_T_COL)	32	RO	0000_0000h
228h	Tx 64-Byte Packets Statistic Register (RMON_T_P64)	32	RO	0000_0000h
22Ch	Tx 65- to 127-byte Packets Statistic Register (RMON_T_P65TO127)	32	RO	0000_0000h
230h	Tx 128- to 255-byte Packets Statistic Register (RMON_T_P128TO255)	32	RO	0000_0000h
234h	Tx 256- to 511-byte Packets Statistic Register (RMON_T_P256TO511)	32	RO	0000_0000h
238h	Tx 512- to 1023-byte Packets Statistic Register (RMON_T_P512TO1023)	32	RO	0000_0000h
23Ch	Tx 1024- to 2047-byte Packets Statistic Register (RMON_T_P1024TO2047)	32	RO	0000_0000h
240h	Tx Packets Greater Than 2048 Bytes Statistic Register (RMON_T_P_GTE2048)	32	RO	0000_0000h
244h	Tx Octets Statistic Register (RMON_T_OCTETS)	32	RO	0000_0000h
24Ch	Frames Transmitted OK Statistic Register (IEEE_T_FRAME_OK)	32	RO	0000_0000h
250h	Frames Transmitted with Single Collision Statistic Register (IEEE_T_1COL)	32	RO	0000_0000h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
254h	Frames Transmitted with Multiple Collisions Statistic Register (IEEE_T_MCOL)	32	RO	0000_0000h
258h	Frames Transmitted after Deferral Delay Statistic Register (IEEE_T_DEF)	32	RO	0000_0000h
25Ch	Frames Transmitted with Late Collision Statistic Register (IEEE_T_LCOL)	32	RO	0000_0000h
260h	Frames Transmitted with Excessive Collisions Statistic Register (IEEE_T_EXCOL)	32	RO	0000_0000h
264h	Frames Transmitted with Tx FIFO Underrun Statistic Register (IEEE_T_MACERR)	32	RO	0000_0000h
268h	Frames Transmitted with Carrier Sense Error Statistic Register (IEEE_T_CSERR)	32	RO	0000_0000h
26Ch	Reserved Statistic Register (IEEE_T_SQE)	32	RO	0000_0000h
270h	Flow Control Pause Frames Transmitted Statistic Register (IEEE_T_FDXFC)	32	RO	0000_0000h
274h	Octet Count for Frames Transmitted w/o Error Statistic Register (IEEE_T_OCTETS_OK)	32	RO	0000_0000h
284h	Rx Packet Count Statistic Register (RMON_R_PACKETS)	32	RO	0000_0000h
288h	Rx Broadcast Packets Statistic Register (RMON_R_BC_PKT)	32	RO	0000_0000h
28Ch	Rx Multicast Packets Statistic Register (RMON_R_MC_PKT)	32	RO	0000_0000h
290h	Rx Packets with CRC/Align Error Statistic Register (RMON_R_CRC_ALIGN)	32	RO	0000_0000h
294h	Rx Packets with Less Than 64 Bytes and Good CRC Statistic Register (RMON_R_UNDERSIZE)	32	RO	0000_0000h
298h	Rx Packets Greater Than MAX_FL and Good CRC Statistic Register (RMON_R_OVERSIZE)	32	RO	0000_0000h
29Ch	Rx Packets Less Than 64 Bytes and Bad CRC Statistic Register (RMON_R_FRAG)	32	RO	0000_0000h
2A0h	Rx Packets Greater Than MAX_FL Bytes and Bad CRC Statistic Register (RMON_R_JAB)	32	RO	0000_0000h
2A8h	Rx 64-Byte Packets Statistic Register (RMON_R_P64)	32	RO	0000_0000h
2ACh	Rx 65- to 127-Byte Packets Statistic Register (RMON_R_P65TO127)	32	RO	0000_0000h
2B0h	Rx 128- to 255-Byte Packets Statistic Register (RMON_R_P128TO255)	32	RO	0000_0000h
2B4h	Rx 256- to 511-Byte Packets Statistic Register (RMON_R_P256TO511)	32	RO	0000_0000h
2B8h	Rx 512- to 1023-Byte Packets Statistic Register (RMON_R_P512TO1023)	32	RO	0000_0000h
2BCh	Rx 1024- to 2047-Byte Packets Statistic Register (RMON_R_P1024TO2047)	32	RO	0000_0000h
2C0h	Rx Packets Greater than 2048 Bytes Statistic Register (RMON_R_P_GTE2048)	32	RO	0000_0000h
2C4h	Rx Octets Statistic Register (RMON_R_OCTETS)	32	RO	0000_0000h
2C8h	Frames not Counted Correctly Statistic Register (IEEE_R_DROP)	32	RO	0000_0000h
2CCh	Frames Received OK Statistic Register (IEEE_R_FRAME_OK)	32	RO	0000_0000h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
2D0h	Frames Received with CRC Error Statistic Register (IEEE_R_CRC)	32	RO	0000_0000h
2D4h	Frames Received with Alignment Error Statistic Register (IEEE_R_ALIGN)	32	RO	0000_0000h
2D8h	Receive FIFO Overflow Count Statistic Register (IEEE_R_MACERR)	32	RO	0000_0000h
2DCh	Flow Control Pause Frames Received Statistic Register (IEEE_R_FDXFC)	32	RO	0000_0000h
2E0h	Octet Count for Frames Received without Error Statistic Register (IEEE_R_OCTETS_OK)	32	RO	0000_0000h
400h	Adjustable Timer Control Register (ATCR)	32	RW	0000_0000h
404h	Timer Value Register (ATVR)	32	RW	0000_0000h
408h	Timer Offset Register (ATOFF)	32	RW	0000_0000h
40Ch	Timer Period Register (ATPER)	32	RW	3B9A_CA00h
410h	Timer Correction Register (ATCOR)	32	RW	0000_0000h
414h	Time-Stamping Clock Period Register (ATINC)	32	RW	0000_0000h
418h	Timestamp of Last Transmitted Frame (ATSTMP)	32	RO	0000_0000h
604h	Timer Global Status Register (TGSR)	32	W1C	0000_0000h
608h	Timer Control Status Register (TCSR0)	32	RW	0000_0000h
60Ch	Timer Compare Capture Register (TCCR0)	32	RW	0000_0000h
610h	Timer Control Status Register (TCSR1)	32	RW	0000_0000h
614h	Timer Compare Capture Register (TCCR1)	32	RW	0000_0000h
618h	Timer Control Status Register (TCSR2)	32	RW	0000_0000h
61Ch	Timer Compare Capture Register (TCCR2)	32	RW	0000_0000h
620h	Timer Control Status Register (TCSR3)	32	RW	0000_0000h
624h	Timer Compare Capture Register (TCCR3)	32	RW	0000_0000h

41.5.1.2 Interrupt Event Register (EIR)

41.5.1.2.1 Offset

Register	Offset
EIR	4h

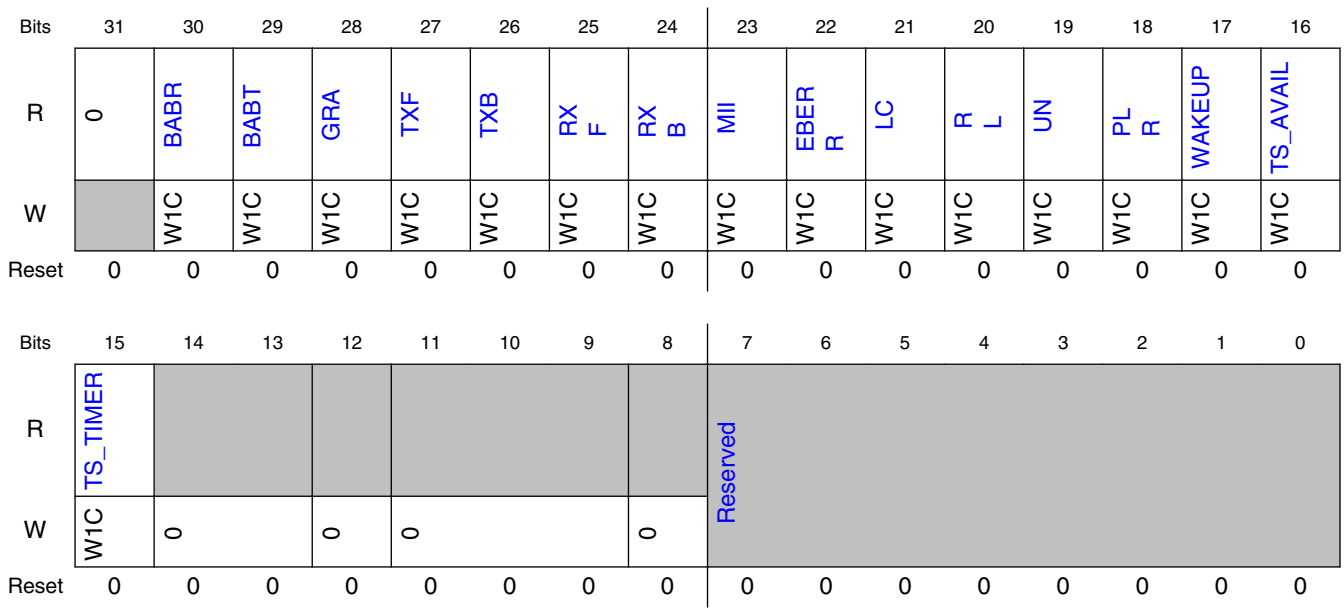
41.5.1.2.2 Function

When an event occurs that sets a bit in EIR, an interrupt occurs if the corresponding bit in the interrupt mask register (EIMR) is also set. Writing a 1 to an EIR bit clears it; writing 0 has no effect. This register is cleared upon hardware reset.

NOTE

TxBD[INT] and RxBD[INT] must be set to 1 to allow setting the corresponding EIR register flags in enhanced mode, ENET_ECR[EN1588] = 1. Legacy mode does not require these flags to be enabled.

41.5.1.2.3 Diagram



41.5.1.2.4 Fields

Field	Function
31 —	Reserved
30 BABR	Babbling Receive Error Indicates a frame was received with length in excess of RCR[MAX_FL] bytes.
29 BABT	Babbling Transmit Error Indicates the transmitted frame length exceeds RCR[MAX_FL] bytes. Usually this condition is caused when a frame that is too long is placed into the transmit data buffer(s). Truncation does not occur.
28 GRA	Graceful Stop Complete This interrupt is asserted after the transmitter is put into a pause state after completion of the frame currently being transmitted. See Graceful Transmit Stop (GTS) for conditions that lead to graceful stop. **NOTE:** The GRA interrupt is asserted only when the TX transitions into the stopped state. If this bit is cleared by writing 1 and the TX is still stopped, the bit is not set again.
27 TXF	Transmit Frame Interrupt Indicates a frame has been transmitted and the last corresponding buffer descriptor has been updated.

Table continues on the next page...

Field	Function
26 TXB	Transmit Buffer Interrupt Indicates a transmit buffer descriptor has been updated.
25 RXF	Receive Frame Interrupt Indicates a frame has been received and the last corresponding buffer descriptor has been updated.
24 RXB	Receive Buffer Interrupt Indicates a receive buffer descriptor is not the last in the frame has been updated.
23 MII	MII Interrupt. Indicates that the MII has completed the data transfer requested.
22 EBERR	Ethernet Bus Error Indicates a system bus error occurred when a uDMA transaction is underway. When this bit is set, ECR[ETHEREN] is cleared, halting frame processing by the MAC. When this occurs, software must ensure proper actions, possibly resetting the system, to resume normal operation.
21 LC	Late Collision Indicates a collision occurred beyond the collision window (slot time) in half-duplex mode. The frame truncates with a bad CRC and the remainder of the frame is discarded.
20 RL	Collision Retry Limit Indicates a collision occurred on each of 16 successive attempts to transmit the frame. The frame is discarded without being transmitted and transmission of the next frame commences. This error can only occur in half-duplex mode.
19 UN	Transmit FIFO Underrun Indicates the transmit FIFO became empty before the complete frame was transmitted. **NOTE:** In situations where the device has various masters generating high traffic, a FIFO underrun can occur on the transmit FIFO. To avoid transmit FIFO underrun, store and forward can be enabled in ENET_TFWR[STRFWD]. See Transmit FIFO Watermark Register (TFWR). Also, a higher priority can be set for ENET traffic using available means on the central bus fabric connecting the ENET module.
18 PLR	Payload Receive Error Indicates a frame was received with a payload length error. See Frame Length/Type Verification: Payload Length Check for more information.
17 WAKEUP	Node Wakeup Request Indication Read-only status bit to indicate that a magic packet has been detected. Will act only if ECR[MAGICEN] is set.
16 TS_AVAIL	Transmit Timestamp Available Indicates that the timestamp of the last transmitted timing frame is available in the ATSTMP register.
15 TS_TIMER	Timestamp Timer The adjustable timer reached the period event. A period event interrupt can be generated if ATCR[PEREN] is set and the timer wraps according to the periodic setting in the ATPER register. Set the timer period value before setting ATCR[PEREN].
14-13 —	This write-only field is reserved. It must always be written with the value 0.
12 —	This write-only field is reserved. It must always be written with the value 0.
11-9	This write-only field is reserved. It must always be written with the value 0.

Table continues on the next page...

Field	Function
—	
8 —	This write-only field is reserved. It must always be written with the value 0.
7-0 —	This write-only field is reserved. It must always be written with the value 0.

41.5.1.3 Interrupt Mask Register (EIMR)

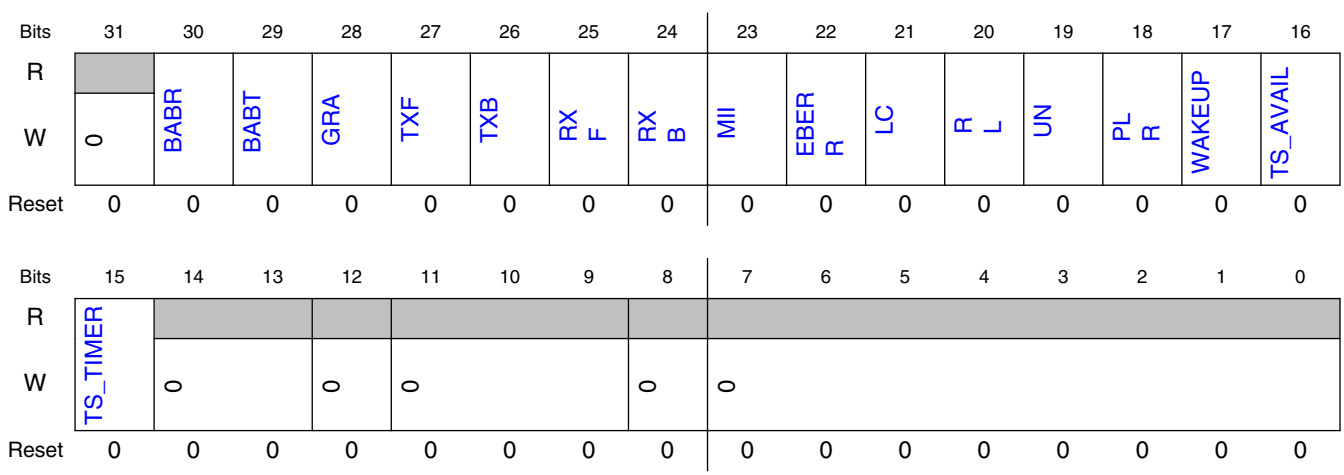
41.5.1.3.1 Offset

Register	Offset
EIMR	8h

41.5.1.3.2 Function

EIMR controls which interrupt events are allowed to generate actual interrupts. A hardware reset clears this register. If the corresponding bits in the EIR and EIMR registers are set, an interrupt is generated. The interrupt signal remains asserted until a 1 is written to the EIR field (write 1 to clear) or a 0 is written to the EIMR field.

41.5.1.3.3 Diagram



41.5.1.3.4 Fields

Field	Function
31 —	This write-only field is reserved. It must always be written with the value 0.
30 BABR	BABR Interrupt Mask Corresponds to interrupt source EIR[BABR] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR BABR field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared. 0b - The corresponding interrupt source is masked. 1b - The corresponding interrupt source is not masked.
29 BABT	BABT Interrupt Mask Corresponds to interrupt source EIR[BABT] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR BABT field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared. 0b - The corresponding interrupt source is masked. 1b - The corresponding interrupt source is not masked.
28 GRA	GRA Interrupt Mask Corresponds to interrupt source EIR[GRA] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR GRA field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared. 0b - The corresponding interrupt source is masked. 1b - The corresponding interrupt source is not masked.
27 TXF	TXF Interrupt Mask Corresponds to interrupt source EIR[TXF] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR TXF field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared. 0b - The corresponding interrupt source is masked. 1b - The corresponding interrupt source is not masked.
26 TXB	TXB Interrupt Mask Corresponds to interrupt source EIR[TXB] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR TXF field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared. 0b - The corresponding interrupt source is masked. 1b - The corresponding interrupt source is not masked.
25 RXF	RXF Interrupt Mask Corresponds to interrupt source EIR[RXF] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR RXF field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared. 0b - The corresponding interrupt source is masked. 1b - The corresponding interrupt source is not masked.
24 RXB	RXB Interrupt Mask Corresponds to interrupt source EIR[RXB] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The

Table continues on the next page...

Field	Function
	corresponding EIR RXB field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared. 0b - The corresponding interrupt source is masked. 1b - The corresponding interrupt source is not masked.
23 MII	MII Interrupt Mask Corresponds to interrupt source EIR[MII] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR MII field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared. 0b - The corresponding interrupt source is masked. 1b - The corresponding interrupt source is not masked.
22 EBERR	EBERR Interrupt Mask Corresponds to interrupt source EIR[EBERR] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR EBERR field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared. 0b - The corresponding interrupt source is masked. 1b - The corresponding interrupt source is not masked.
21 LC	LC Interrupt Mask Corresponds to interrupt source EIR[LC] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR LC field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared. 0b - The corresponding interrupt source is masked. 1b - The corresponding interrupt source is not masked.
20 RL	RL Interrupt Mask Corresponds to interrupt source EIR[RL] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR RL field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared. 0b - The corresponding interrupt source is masked. 1b - The corresponding interrupt source is not masked.
19 UN	UN Interrupt Mask Corresponds to interrupt source EIR[UN] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR UN field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared. 0b - The corresponding interrupt source is masked. 1b - The corresponding interrupt source is not masked.
18 PLR	PLR Interrupt Mask Corresponds to interrupt source EIR[PLR] and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR PLR field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared. 0b - The corresponding interrupt source is masked. 1b - The corresponding interrupt source is not masked.
17 WAKEUP	WAKEUP Interrupt Mask Corresponds to interrupt source EIR[WAKEUP] register and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR WAKEUP field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared.

Table continues on the next page...

Field	Function
	0b - The corresponding interrupt source is masked. 1b - The corresponding interrupt source is not masked.
16 TS_AVAIL	TS_AVAIL Interrupt Mask Corresponds to interrupt source EIR[TS_AVAIL] register and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR TS_AVAIL field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared. 0b - The corresponding interrupt source is masked. 1b - The corresponding interrupt source is not masked.
15 TS_TIMER	TS_TIMER Interrupt Mask Corresponds to interrupt source EIR[TS_TIMER] register and determines whether an interrupt condition can generate an interrupt. At every module clock, the EIR samples the signal generated by the interrupting source. The corresponding EIR TS_TIMER field reflects the state of the interrupt signal even if the corresponding EIMR field is cleared. 0b - The corresponding interrupt source is masked. 1b - The corresponding interrupt source is not masked.
14-13 —	This write-only field is reserved. It must always be written with the value 0.
12 —	This write-only field is reserved. It must always be written with the value 0.
11-9 —	This write-only field is reserved. It must always be written with the value 0.
8 —	This write-only field is reserved. It must always be written with the value 0.
7-0 —	This write-only field is reserved. It must always be written with the value 0.

41.5.1.4 Receive Descriptor Active Register - Ring 0 (RDAR)

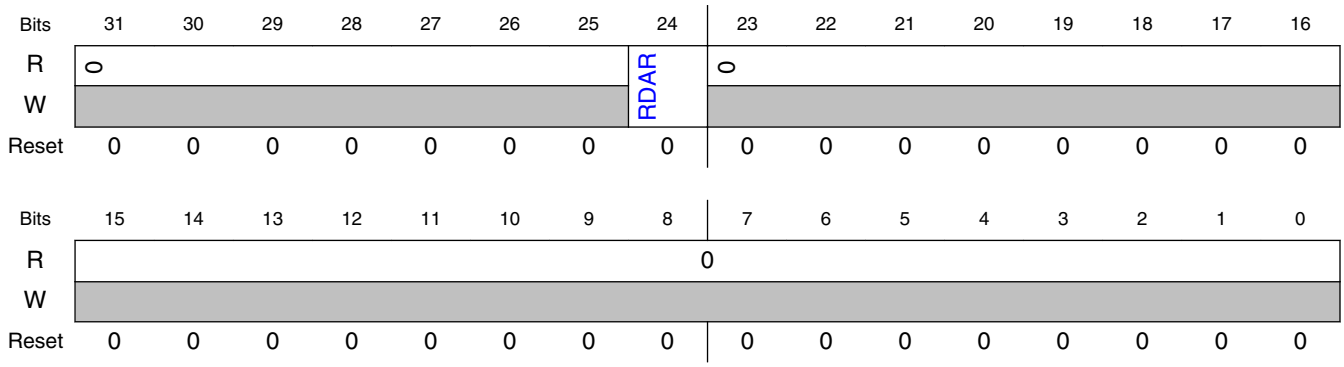
41.5.1.4.1 Offset

Register	Offset
RDAR	10h

41.5.1.4.2 Function

RDAR is a command register, written by the user, to indicate that the receive descriptor ring has been updated, that is, that the driver produced empty receive buffers with the empty bit set.

41.5.1.4.3 Diagram



41.5.1.4.4 Fields

Field	Function
31-25 —	Reserved
24 RDAR	Receive Descriptor Active Always set to 1 when this register is written, regardless of the value written. This field is cleared by the MAC device when no additional empty descriptors remain in the receive ring. It is also cleared when ECR[ETHEREN] transitions from set to cleared or when ECR[RESET] is set.
23-0 —	Reserved

41.5.1.5 Transmit Descriptor Active Register - Ring 0 (TDAR)

41.5.1.5.1 Offset

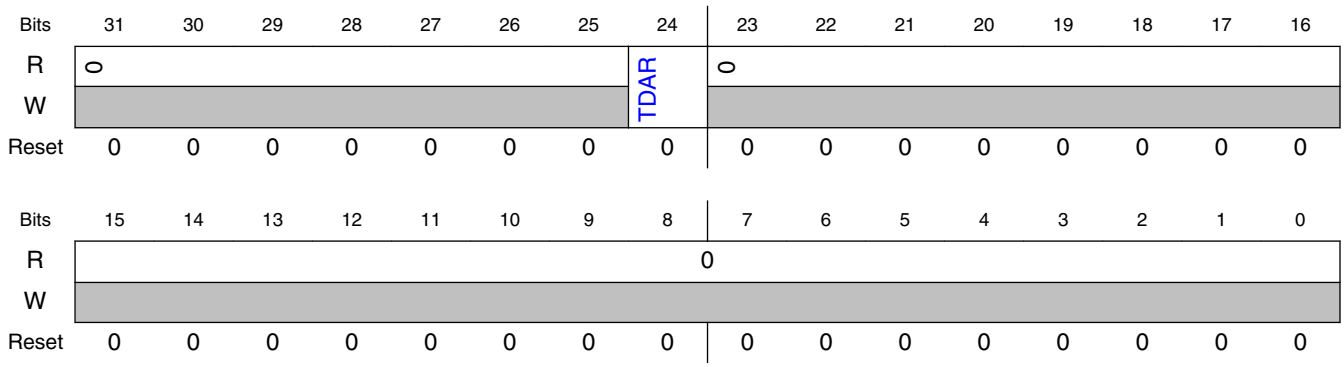
Register	Offset
TDAR	14h

41.5.1.5.2 Function

The TDAR is a command register that the user writes to indicate that the transmit descriptor ring has been updated, that is, that transmit buffers have been produced by the driver with the ready bit set in the buffer descriptor.

The TDAR register is cleared at reset, when ECR[ETHEREN] transitions from set to cleared, or when ECR[RESET] is set.

41.5.1.5.3 Diagram



41.5.1.5.4 Fields

Field	Function
31-25 —	Reserved
24 TDAR	Transmit Descriptor Active Always set to 1 when this register is written, regardless of the value written. This bit is cleared by the MAC device when no additional ready descriptors remain in the transmit ring. Also cleared when ECR[ETHEREN] transitions from set to cleared or when ECR[RESET] is set.
23-0 —	Reserved

41.5.1.6 Ethernet Control Register (ECR)

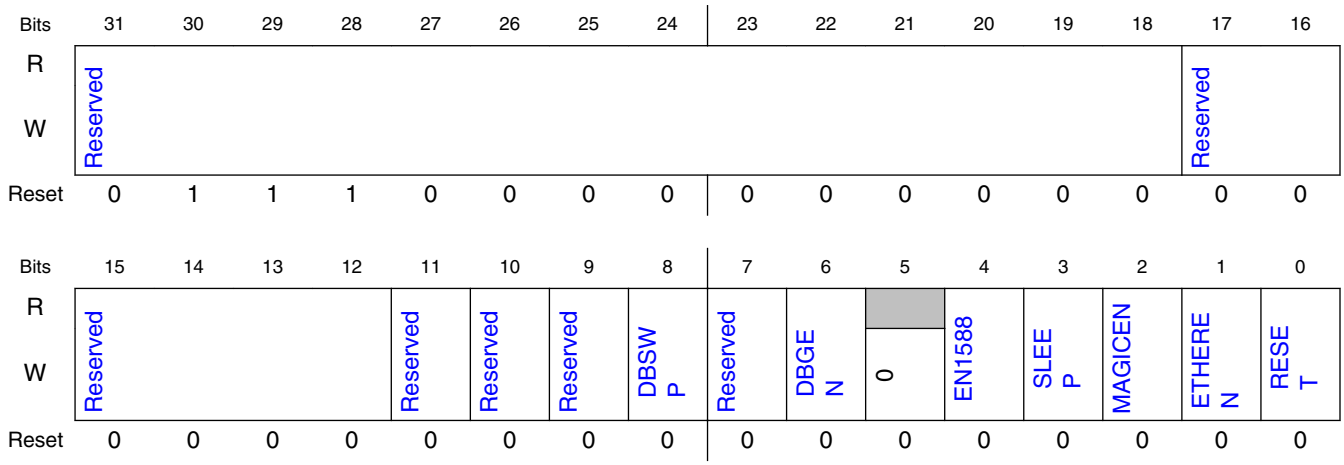
41.5.1.6.1 Offset

Register	Offset
ECR	24h

41.5.1.6.2 Function

ECR is a read/write user register, though hardware may also alter fields in this register. It controls many of the high level features of the Ethernet MAC, including legacy FEC support through the EN1588 field.

41.5.1.6.3 Diagram



41.5.1.6.4 Fields

Field	Function
31-18 —	Always write 01110000000000b to this field.
17-12 —	Always write 0 to this field.
11 —	Always write 0 to this field.
10 —	Always write 0 to this field.
9 —	Always write 0 to this field.
8 DBSWP	Descriptor Byte Swapping Enable Swaps the byte locations of the buffer descriptors. **NOTE:** This field must be written to 1 after reset. 0b - The buffer descriptor bytes are not swapped to support big-endian devices. 1b - The buffer descriptor bytes are swapped to support little-endian devices.
7 —	Always write 0 to this field.
6 DBGEN	Debug Enable Enables the MAC to enter hardware freeze mode when the device enters debug mode. 0b - MAC continues operation in debug mode. 1b - MAC enters hardware freeze mode when the processor is in debug mode.
5 —	This write-only field is reserved. It must always be written with the value 0.
4	EN1588 Enable

Table continues on the next page...

Field	Function
EN1588	Enables enhanced functionality of the MAC. 0b - Legacy FEC buffer descriptors and functions enabled. 1b - Enhanced frame time-stamping functions enabled. Has no effect within the MAC besides controlling the DMA control bit ena_1588.
3 SLEEP	Sleep Mode Enable 0b - Normal operating mode. 1b - Sleep mode.
2 MAGICEN	Magic Packet Detection Enable Enables/disables magic packet detection. **NOTE:** MAGICEN is relevant only if the SLEEP field is set. If MAGICEN is set, changing the SLEEP field enables/disables sleep mode and magic packet detection. **NOTE:** EIMR[WAKEUP] must be written to one if Magic packet wakeup is programed to wake up the chip from low power mode. 0b - Magic detection logic disabled. 1b - The MAC core detects magic packets and asserts EIR[WAKEUP] when a frame is detected.
1 ETHEREN	Ethernet Enable Enables/disables the Ethernet MAC. When the MAC is disabled, the buffer descriptors for an aborted transmit frame are not updated. The uDMA, buffer descriptor, and FIFO control logic are reset, including the buffer descriptor and FIFO pointers. Hardware clears this field under the following conditions: • RESET is set by software • An error condition causes the EBERR field to set. **NOTE:** • ETHEREN must be set at the very last step during ENET configuration/setup/initialization, only *after* all other ENET-related registers have been configured. • If ETHEREN is cleared to 0 by software then next time ETHEREN is set, the EIR interrupts must cleared to 0 due to previous pending interrupts. 0b - Reception immediately stops and transmission stops after a bad CRC is appended to any currently transmitted frame. 1b - MAC is enabled, and reception and transmission are possible.
0 RESET	Ethernet MAC Reset When this field is set, it clears the ETHEREN field.

41.5.1.7 MII Management Frame Register (MMFR)

41.5.1.7.1 Offset

Register	Offset
MMFR	40h

41.5.1.7.2 Function

Writing to MMFR triggers a management frame transaction to the PHY device unless MSCR is programmed to zero.

If MSCR is changed from zero to non-zero during a write to MMFR, an MII frame is generated with the data previously written to the MMFR. This allows MMFR and MSCR to be programmed in either order if MSCR is currently zero.

If the MMFR register is written while frame generation is in progress, the frame contents are altered. Software must use the EIR[MII] interrupt indication to avoid writing to the MMFR register while frame generation is in progress.

41.5.1.7.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ST		OP		PA				RA						TA	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

41.5.1.7.4 Fields

Field	Function
31-30 ST	Start Of Frame Delimiter See Table 41-38 (Clause 22) or Table 41-40 (Clause 45) for correct value.
29-28 OP	Operation Code See Table 41-38 (Clause 22) or Table 41-40 (Clause 45) for correct value.
27-23 PA	PHY Address See Table 41-38 (Clause 22) or Table 41-40 (Clause 45) for correct value.
22-18 RA	Register Address See Table 41-38 (Clause 22) or Table 41-40 (Clause 45) for correct value.
17-16 TA	Turn Around This field must be programmed to 10 to generate a valid MII management frame.
15-0 DATA	Management Frame Data This is the field for data to be written to or read from the PHY register.

41.5.1.8 MII Speed Control Register (MSCR)

41.5.1.8.1 Offset

Register	Offset
MSCR	44h

41.5.1.8.2 Function

MSCR provides control of the MII clock (MDC pin) frequency and allows a preamble drop on the MII management frame.

The MII_SPEED field must be programmed with a value to provide an MDC frequency of less than or equal to 2.5 MHz to be compliant with the IEEE 802.3 MII specification. The MII_SPEED must be set to a non-zero value to source a read or write management frame. After the management frame is complete, the MSCR register may optionally be cleared to turn off MDC. The MDC signal generated has a 50% duty cycle except when MII_SPEED changes during operation. This change takes effect following a rising or falling edge of MDC.

For example, if the internal module clock (that is, peripheral bus clock) is 25 MHz, programming MII_SPEED to 0x4 results in an MDC as given in the following equation:

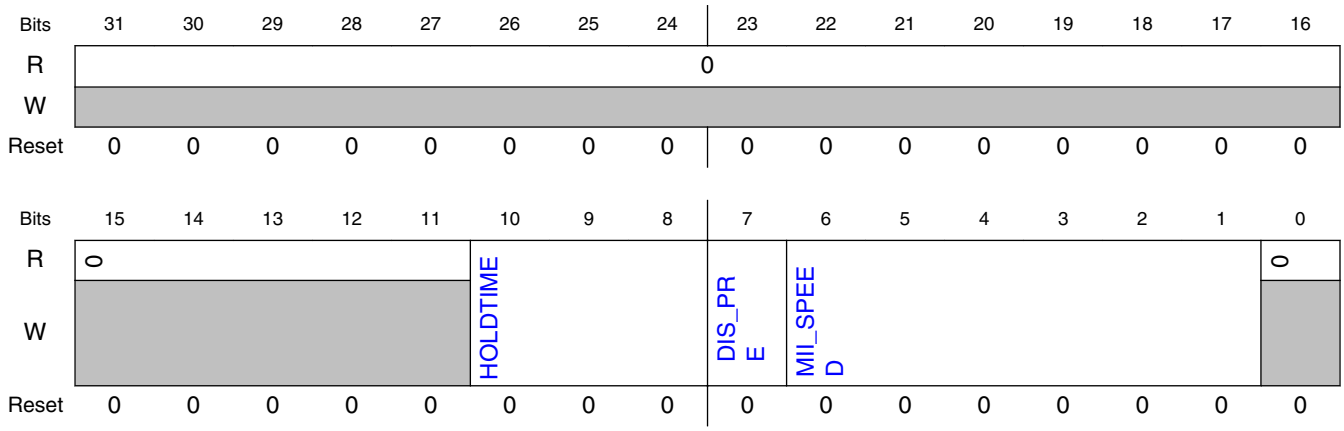
MII clock frequency = 25 MHz / ((4 + 1) x 2) = 2.5 MHz

The following table shows the optimum values for MII_SPEED as a function of IPS bus clock frequency.

Table 41-46. Programming Examples for MSCR

Internal module clock frequency	MSCR [MII_SPEED]	MDC frequency
25 MHz	0x4	2.50 MHz
33 MHz	0x6	2.36 MHz
40 MHz	0x7	2.50 MHz
50 MHz	0x9	2.50 MHz
66 MHz	0xD	2.36 MHz

41.5.1.8.3 Diagram



41.5.1.8.4 Fields

Field	Function
31-11 —	Reserved
10-8 HOLDTIME	Hold time On MDIO Output IEEE802.3 clause 22 defines a minimum of 10 ns for the hold time on the MDIO output. Depending on the host bus frequency, the setting may need to be increased. 000b - 1 internal module clock cycle 001b - 2 internal module clock cycles 010b - 3 internal module clock cycles 111b - 8 internal module clock cycles
7 DIS_PRE	Disable Preamble Enables/disables prepending a preamble to the MII management frame. The MII standard allows the preamble to be dropped if the attached PHY devices do not require it. 0b - Preamble enabled. 1b - Preamble (32 ones) is not prepended to the MII management frame.
6-1 MII_SPEED	MII Speed Controls the frequency of the MII management interface clock (MDC) relative to the internal module clock. A value of 0 in this field turns off MDC and leaves it in low voltage state. Any non-zero value results in the MDC frequency of: 1/((MII_SPEED + 1) x 2) of the internal module clock frequency
0 —	Reserved

41.5.1.9 MIB Control Register (MIBC)

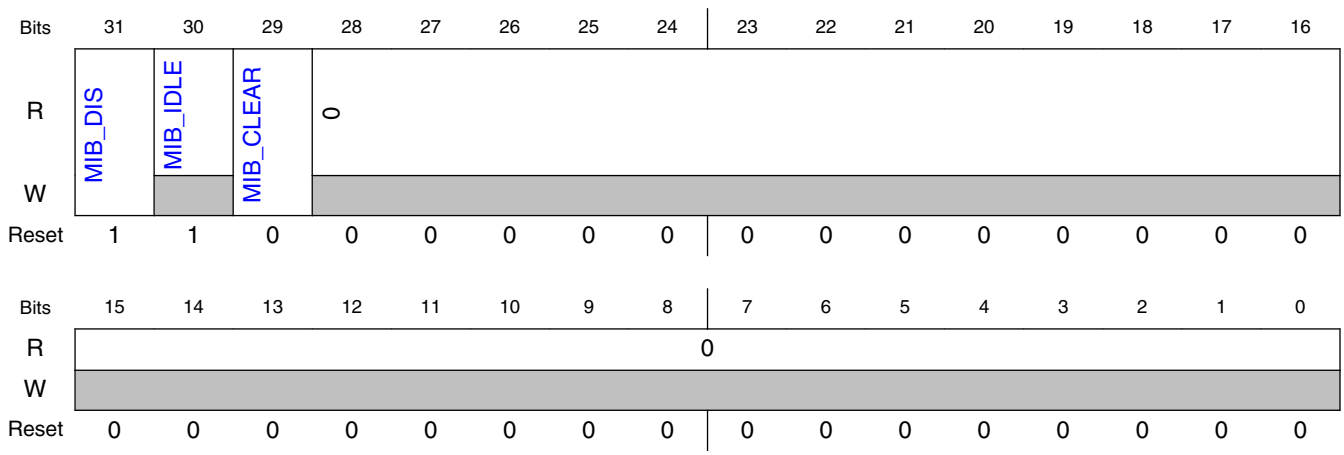
41.5.1.9.1 Offset

Register	Offset
MIBC	64h

41.5.1.9.2 Function

MIBC is a read/write register controlling and observing the state of the MIB block. Access this register to disable the MIB block operation or clear the MIB counters. The MIB_DIS field resets to 1.

41.5.1.9.3 Diagram



41.5.1.9.4 Fields

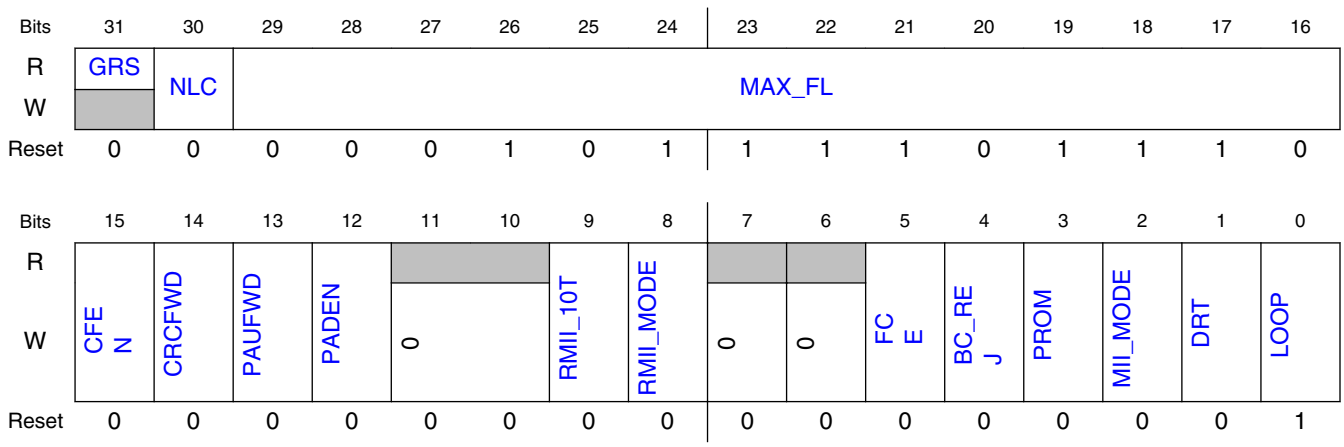
Field	Function
31 MIB_DIS	Disable MIB Logic If this control field is set, 0b - MIB logic is enabled. 1b - MIB logic is disabled. The MIB logic halts and does not update any MIB counters.
30 MIB_IDLE	MIB Idle 0b - The MIB block is updating MIB counters. 1b - The MIB block is not currently updating any MIB counters.
29 MIB_CLEAR	MIB Clear **NOTE:** This field is not self-clearing. To clear the MIB counters set and then clear this field. 0b - See note above. 1b - All statistics counters are reset to 0.
28-0 —	Reserved

41.5.1.10 Receive Control Register (RCR)

41.5.1.10.1 Offset

Register	Offset
RCR	84h

41.5.1.10.2 Diagram



41.5.1.10.3 Fields

Field	Function
31 GRS	Graceful Receive Stopped Read-only status indicating that the MAC receive datapath is stopped. 0b - Receive not stopped 1b - Receive stopped
30 NLC	Payload Length Check Disable Enables/disables a payload length check. 0b - The payload length check is disabled. 1b - The core checks the frame's payload length with the frame length/type field. Errors are indicated in the EIR[PLR] field.
29-16 MAX_FL	Maximum Frame Length Resets to decimal 1518. Length is measured starting at DA and includes the CRC at the end of the frame. Transmit frames longer than MAX_FL cause the BABT interrupt to occur. Receive frames longer than MAX_FL cause the BABR interrupt to occur and set the LG field in the end of frame receive buffer descriptor. The recommended default value to be programmed is 1518 or 1522 if VLAN tags are supported.
15	MAC Control Frame Enable

Table continues on the next page...

Field	Function
CFEN	Enables/disables the MAC control frame. 0b - MAC control frames with any opcode other than 0x0001 (pause frame) are accepted and forwarded to the client interface. 1b - MAC control frames with any opcode other than 0x0001 (pause frame) are silently discarded.
14 CRCFWD	Terminate/Forward Received CRC Specifies whether the CRC field of received frames is transmitted or stripped. **NOTE:** If padding function is enabled (PADEN = 1), CRCFWD is ignored and the CRC field is checked and always terminated and removed. 0b - The CRC field of received frames is transmitted to the user application. 1b - The CRC field is stripped from the frame.
13 PAUFWD	Terminate/Forward Pause Frames Specifies whether pause frames are terminated or forwarded. 0b - Pause frames are terminated and discarded in the MAC. 1b - Pause frames are forwarded to the user application.
12 PADEN	Enable Frame Padding Remove On Receive Specifies whether the MAC removes padding from received frames. 0b - No padding is removed on receive by the MAC. 1b - Padding is removed from received frames.
11-10 —	This write-only field is reserved. It must always be written with the value 0.
9 RMII_10T	Enables 10-Mbit/s mode of the RMII . 0b - 100-Mbit/s operation. 1b - 10-Mbit/s operation.
8 RMII_MODE	RMII Mode Enable Specifies whether the MAC is configured for MII mode or RMII operation . 0b - MAC configured for MII mode. 1b - MAC configured for RMII operation.
7 —	This write-only field is reserved. It must always be written with the value 0.
6 —	This write-only field is reserved. It must always be written with the value 0.
5 FCE	Flow Control Enable If set, the receiver detects PAUSE frames. Upon PAUSE frame detection, the transmitter stops transmitting data frames for a given duration. 0b - Disable flow control 1b - Enable flow control
4 BC_REJ	Broadcast Frame Reject If set, frames with destination address (DA) equal to 0xFFFF_FFFF_FFFF are rejected unless the PROM field is set. If BC_REJ and PROM are set, frames with broadcast DA are accepted and the MISS (M) is set in the receive buffer descriptor. 0b - Will not reject frames as described above 1b - Will reject frames as described above
3 PROM	Promiscuous Mode All frames are accepted regardless of address matching. 0b - Disabled. 1b - Enabled.

Table continues on the next page...

Field	Function
2 MII_MODE	Media Independent Interface Mode This field must always be set. 0b - Reserved. 1b - MII or RMII mode, as indicated by the RMII_MODE field.
1 DRT	Disable Receive On Transmit 0b - Receive path operates independently of transmit (i.e., full-duplex mode). Can also be used to monitor transmit activity in half-duplex mode. 1b - Disable reception of frames while transmitting. (Normally used for half-duplex mode.)
0 LOOP	Internal Loopback This is an MII internal loopback, therefore MII_MODE must be written to 1 and RMII_MODE must be written to 0. 0b - Loopback disabled. 1b - Transmitted frames are looped back internal to the device and transmit MII output signals are not asserted. DRT must be cleared.

41.5.1.11 Transmit Control Register (TCR)

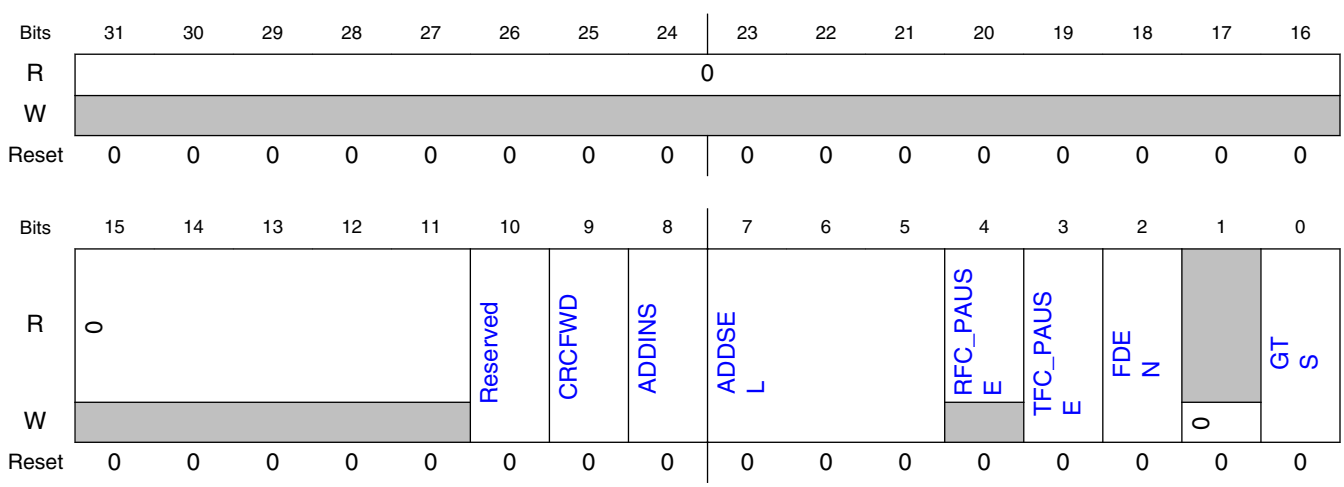
41.5.1.11.1 Offset

Register	Offset
TCR	C4h

41.5.1.11.2 Function

TCR is read/write and configures the transmit block. This register is cleared at system reset. FDEN can only be modified when ECR[ETHEREN] is cleared.

41.5.1.11.3 Diagram



41.5.1.11.4 Fields

Field	Function
31-11 —	Reserved
10 —	This field is read/write and must be set to 0.
9 CRCFWD	Forward Frame From Application With CRC 0b - TxBD[TC] controls whether the frame has a CRC from the application. 1b - The transmitter does not append any CRC to transmitted frames, as it is expecting a frame with CRC from the application.
8 ADDINS	Set MAC Address On Transmit 0b - The source MAC address is not modified by the MAC. 1b - The MAC overwrites the source MAC address with the programmed MAC address according to ADDSEL.
7-5 ADDSEL	Source MAC Address Select On Transmit If ADDINS is set, indicates the MAC address that overwrites the source MAC address. 000b - Node MAC address programmed on PADDR1/2 registers. 100b - Reserved. 101b - Reserved. 110b - Reserved.
4 RFC_PAUSE	Receive Frame Control Pause This status field is set when a full-duplex flow control pause frame is received and the transmitter pauses for the duration defined in this pause frame. This field automatically clears when the pause duration is complete.
3 TFC_PAUSE	Transmit Frame Control Pause Pauses frame transmission. When this field is set, EIR[GRA] is set. With transmission of data frames stopped, the MAC transmits a MAC control PAUSE frame. Next, the MAC clears TFC_PAUSE and resumes transmitting data frames. If the transmitter pauses due to user assertion of GTS or reception of a PAUSE frame, the MAC may continue transmitting a MAC control PAUSE frame. 0b - No PAUSE frame transmitted. 1b - The MAC stops transmission of data frames after the current transmission is complete.
2 FDEN	Full-Duplex Enable If this field is set, frames transmit independent of carrier sense and collision inputs. Only modify this bit when ECR[ETHEREN] is cleared. 0b - Disable full-duplex 1b - Enable full-duplex
1 —	This write-only field is reserved. It must always be written with the value 0.
0 GTS	Graceful Transmit Stop When this field is set, MAC stops transmission after any frame currently transmitted is complete and EIR[GRA] is set. If frame transmission is not currently underway, the GRA interrupt is asserted immediately. After transmission finishes, clear GTS to restart. The next frame in the transmit FIFO is then transmitted. If an early collision occurs during transmission when GTS is set, transmission stops after the collision. The frame is transmitted again after GTS is cleared. There may be old frames in the transmit FIFO that transmit when GTS is reasserted. To avoid this, clear ECR[ETHEREN] following the GRA interrupt.

Field	Function
	0b - Disable graceful transmit stop 1b - Enable graceful transmit stop

41.5.1.12 Physical Address Lower Register (PALR)

41.5.1.12.1 Offset

Register	Offset
PALR	E4h

41.5.1.12.2 Function

PALR contains the lower 32 bits (bytes 0, 1, 2, 3) of the 48-bit address used in the address recognition process to compare with the destination address (DA) field of receive frames with an individual DA. In addition, this register is used in bytes 0 through 3 of the six-byte source address field when transmitting PAUSE frames.

41.5.1.12.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PADDR1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PADDR1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

41.5.1.12.4 Fields

Field	Function
31-0 PADDR1	Pause Address Bytes 0 (bits 31:24), 1 (bits 23:16), 2 (bits 15:8), and 3 (bits 7:0) of the 6-byte individual address are used for exact match and the source address field in PAUSE frames.

41.5.1.13 Physical Address Upper Register (PAUR)

41.5.1.13.1 Offset

Register	Offset
PAUR	E8h

41.5.1.13.2 Function

PAUR contains the upper 16 bits (bytes 4 and 5) of the 48-bit address used in the address recognition process to compare with the destination address (DA) field of receive frames with an individual DA. In addition, this register is used in bytes 4 and 5 of the six-byte source address field when transmitting PAUSE frames. Bits 15:0 of PAUR contain a constant type field (0x8808) for transmission of PAUSE frames.

41.5.1.13.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PADDR2															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TYPE															
W																
Reset	1	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0

41.5.1.13.4 Fields

Field	Function
31-16 PADDR2	Bytes 4 (bits 31:24) and 5 (bits 23:16) of the 6-byte individual address used for exact match, and the source address field in PAUSE frames.
15-0 TYPE	Type Field In PAUSE Frames These fields have a constant value of 0x8808.

41.5.1.14 Opcode/Pause Duration Register (OPD)

41.5.1.14.1 Offset

Register	Offset
OPD	ECh

41.5.1.14.2 Function

OPD is read/write accessible. This register contains the 16-bit opcode and 16-bit pause duration fields used in transmission of a PAUSE frame. The opcode field is a constant value, 0x0001. When another node detects a PAUSE frame, that node pauses transmission for the duration specified in the pause duration field. The lower 16 bits of this register are not reset and you must initialize it.

41.5.1.14.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	OPCODE															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PAUSE_DUR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

41.5.1.14.4 Fields

Field	Function
31-16 OPCODE	Opcode Field In PAUSE Frames These fields have a constant value of 0x0001.
15-0 PAUSE_DUR	Pause Duration Pause duration field used in PAUSE frames.

41.5.1.15 Transmit Interrupt Coalescing Register (TXIC0)

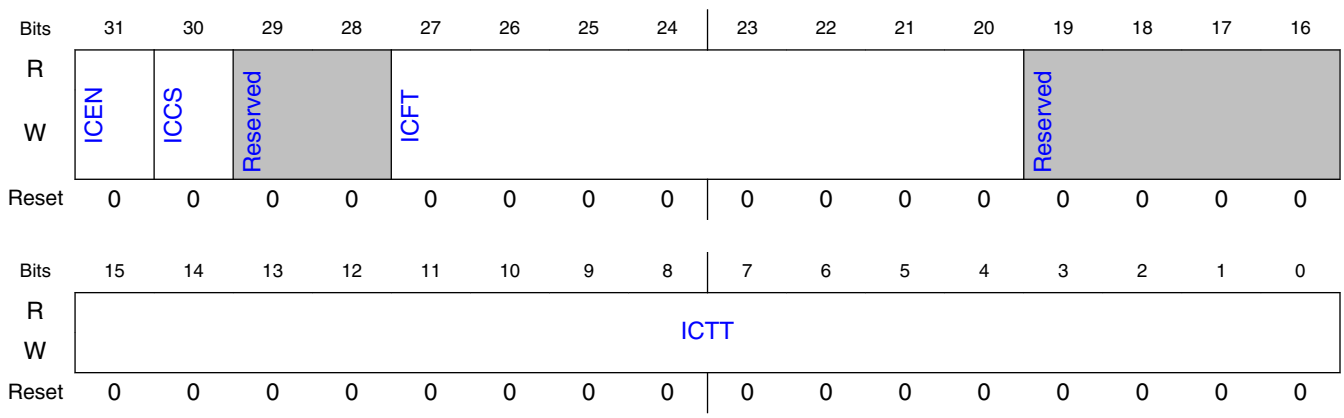
41.5.1.15.1 Offset

Register	Offset
TXIC0	F0h

41.5.1.15.2 Function

See Interrupt coalescence for more information.

41.5.1.15.3 Diagram



41.5.1.15.4 Fields

Field	Function
31 ICEN	Interrupt Coalescing Enable 0b - Disable Interrupt coalescing. 1b - Enable Interrupt coalescing.
30 ICCS	Interrupt Coalescing Timer Clock Source Select 0b - Use MII/GMII TX clocks. 1b - Use ENET system clock.
29-28 —	This field must be set to 0.
27-20 ICFT	Interrupt coalescing frame count threshold This value determines the number of frames needed to be transmitted for raising an interrupt. Frame counter restarts after reaching this threshold value or after the expiring of the coalescing timer. Must be greater than zero to avoid unpredictable behavior.
19-16 —	This field must be set to 0.
15-0 ICTT	Interrupt coalescing timer threshold

Field	Function
	Interrupt coalescing timer threshold in units of 64 clock periods. This value determines the maximum amount of time after transmitting a frame before raising an interrupt. The threshold timer is disabled after expiring or number of frame transmission defined by ICFT and starts again upon transmission of the next first frame. Must be greater than zero to avoid unpredictable behavior.

41.5.1.16 Receive Interrupt Coalescing Register (RXIC0)

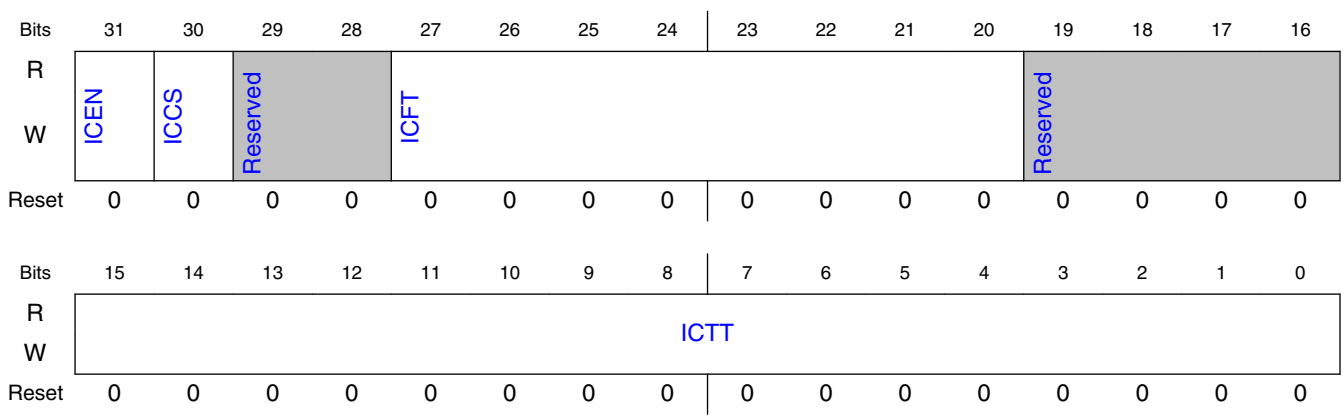
41.5.1.16.1 Offset

Register	Offset
RXIC0	100h

41.5.1.16.2 Function

See Interrupt coalescence for more information.

41.5.1.16.3 Diagram



41.5.1.16.4 Fields

Field	Function
31 ICEN	Interrupt Coalescing Enable 0b - Disable Interrupt coalescing. 1b - Enable Interrupt coalescing.
30	Interrupt Coalescing Timer Clock Source Select 0b - Use MII/GMII TX clocks.

Table continues on the next page...

Field	Function
ICCS	1b - Use ENET system clock.
29-28 —	This field must be set to 0.
27-20 ICFT	Interrupt coalescing frame count threshold This value determines the number of frames needed to be received for raising an interrupt. Frame counter restarts after reaching this threshold value or after the expiring of the coalescing timer. Must be greater than zero to avoid unpredictable behavior.
19-16 —	This field must be set to 0.
15-0 ICTT	Interrupt coalescing timer threshold Interrupt coalescing timer threshold in units of 64 clock periods. This value determines the maximum amount of time after receiving a frame before raising an interrupt. The threshold timer is disabled after expiring or number of frame reception defined by ICFT and starts again upon reception of the next first frame. Must be greater than zero to avoid unpredictable behavior.

41.5.1.17 Descriptor Individual Upper Address Register (IAUR)

41.5.1.17.1 Offset

Register	Offset
IAUR	118h

41.5.1.17.2 Function

IAUR contains the upper 32 bits of the 64-bit individual address hash table. The address recognition process uses this table to check for a possible match with the destination address (DA) field of receive frames with an individual DA. This register is not reset and you must initialize it.

41.5.1.17.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	IADDR1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IADDR1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

41.5.1.17.4 Fields

Field	Function
31-0 IADDR1	Contains the upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a unicast address. Bit 31 of IADDR1 contains hash index bit 63. Bit 0 of IADDR1 contains hash index bit 32.

41.5.1.18 Descriptor Individual Lower Address Register (IALR)

41.5.1.18.1 Offset

Register	Offset
IALR	11Ch

41.5.1.18.2 Function

IALR contains the lower 32 bits of the 64-bit individual address hash table. The address recognition process uses this table to check for a possible match with the DA field of receive frames with an individual DA. This register is not reset and you must initialize it.

41.5.1.18.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	IADDR2															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	IADDR2															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

41.5.1.18.4 Fields

Field	Function
31-0 IADDR2	Contains the lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a unicast address. Bit 31 of IADDR2 contains hash index bit 31. Bit 0 of IADDR2 contains hash index bit 0.

41.5.1.19 Descriptor Group Upper Address Register (GAUR)

41.5.1.19.1 Offset

Register	Offset
GAUR	120h

41.5.1.19.2 Function

GAUR contains the upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. You must initialize this register.

41.5.1.19.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	GADDR1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GADDR1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

41.5.1.19.4 Fields

Field	Function
31-0 GADDR1	Contains the upper 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. Bit 31 of GADDR1 contains hash index bit 63. Bit 0 of GADDR1 contains hash index bit 32.

41.5.1.20 Descriptor Group Lower Address Register (GALR)

41.5.1.20.1 Offset

Register	Offset
GALR	124h

41.5.1.20.2 Function

GALR contains the lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. You must initialize this register.

41.5.1.20.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	GADDR2															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GADDR2															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

41.5.1.20.4 Fields

Field	Function
31-0 GADDR2	Contains the lower 32 bits of the 64-bit hash table used in the address recognition process for receive frames with a multicast address. Bit 31 of GADDR2 contains hash index bit 31. Bit 0 of GADDR2 contains hash index bit 0.

41.5.1.21 Transmit FIFO Watermark Register (TFWR)

41.5.1.21.1 Offset

Register	Offset
TFWR	144h

41.5.1.21.2 Function

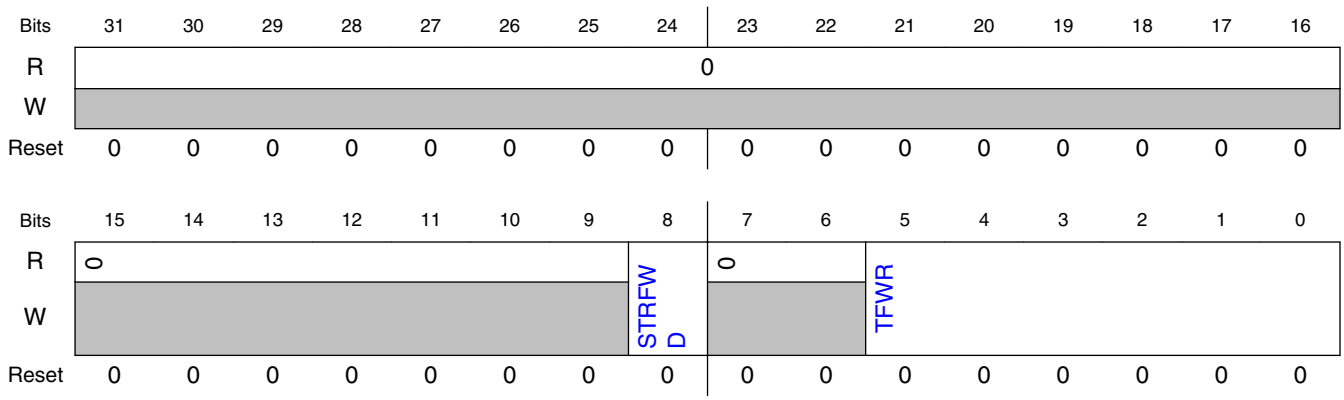
If TFWR[STRFWD] is cleared, TFWR[TFWR] controls the amount of data required in the transmit FIFO before transmission of a frame can begin. This allows you to minimize transmit latency (TFWR = 00 or 01) or allow for larger bus access latency (TFWR = 11) due to contention for the system bus. Setting the watermark to a high value minimizes the risk of transmit FIFO underrun due to contention for the system bus. The byte counts associated with the TFWR field may need to be modified to match a given system requirement, for example, worst-case bus access latency by the transmit data uDMA channel.

When the FIFO level reaches the value the TFWR field and when the STR_FWD is set to ‘0’, the MAC transmit control logic starts frame transmission even before the end-of-frame is available in the FIFO (cut-through operation).

If a complete frame has a size smaller than the threshold programmed with TFWR, the MAC also transmits the Frame to the line.

To enable store and forward on the Transmit path, set STR_FWD to ‘1’. In this case, the MAC starts to transmit data only when a complete frame is stored in the Transmit FIFO.

41.5.1.21.3 Diagram



41.5.1.21.4 Fields

Field	Function
31-9 —	Reserved
8 STRFWD	Store And Forward Enable 0b - Reset. The transmission start threshold is programmed in TFWR[TFWR]. 1b - Enabled.
7-6 —	Reserved
5-0 TFWR	Transmit FIFO Write If TFWR[STRFWD] is cleared, this field indicates the number of bytes, in steps of 64 bytes, written to the transmit FIFO before transmission of a frame begins. **NOTE:** If a frame with less than the threshold is written, it is still sent independently of this threshold setting. The threshold is relevant only if the frame is larger than the threshold given. 000000b - 64 bytes written. 000001b - 64 bytes written. 000010b - 128 bytes written. 000011b - 192 bytes written. 011111b - 1984 bytes written.

41.5.1.22 Receive Descriptor Ring 0 Start Register (RDSR)

41.5.1.22.1 Offset

Register	Offset
RDSR	180h

41.5.1.22.2 Function

RDSR points to the beginning of the circular receive buffer descriptor queue in external memory. This pointer must be 64-bit aligned (bits 2–0 must be zero); however, for optimal performance the pointer should be 512-bit aligned, that is, evenly divisible by 64.

NOTE

This register must be initialized prior to operation

41.5.1.22.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	R_DES_START															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	R_DES_START														0	
W														0		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

41.5.1.22.4 Fields

Field	Function
31-3 R_DES_START	Pointer to the beginning of the receive buffer descriptor queue.
2 —	This write-only field is reserved. It must always be written with the value 0.
1-0 —	Reserved

41.5.1.23 Transmit Buffer Descriptor Ring 0 Start Register (TDSR)

41.5.1.23.1 Offset

Register	Offset
TDSR	184h

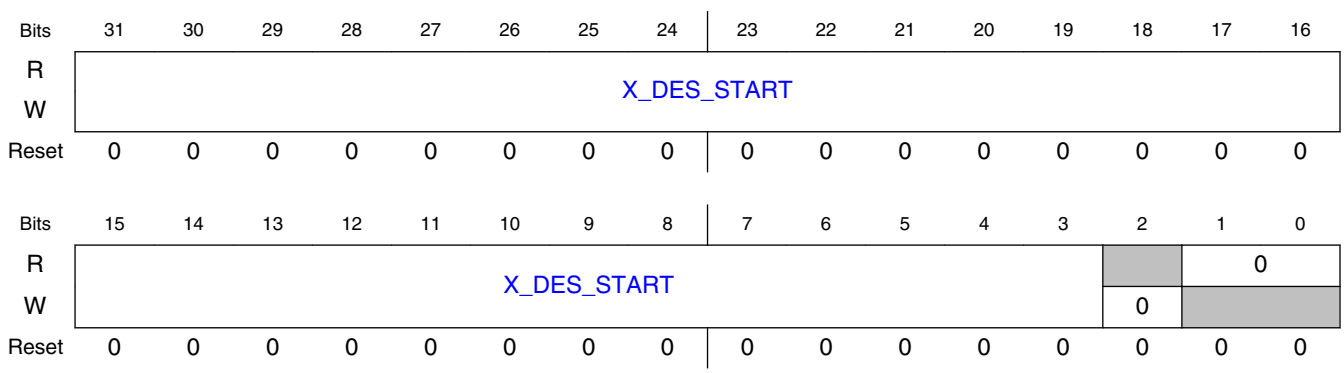
41.5.1.23.2 Function

TDSR provides a pointer to the beginning of the circular transmit buffer descriptor queue in external memory. This pointer must be 64-bit aligned (bits 2–0 must be zero); however, for optimal performance the pointer should be 512-bit aligned, that is, evenly divisible by 64.

NOTE

This register must be initialized prior to operation.

41.5.1.23.3 Diagram



41.5.1.23.4 Fields

Field	Function
31-3 X_DES_START	Pointer to the beginning of the transmit buffer descriptor queue.
2 —	This write-only field is reserved. It must always be written with the value 0.
1-0 —	Reserved

41.5.1.24 Maximum Receive Buffer Size Register - Ring 0 (MRBR)

41.5.1.24.1 Offset

Register	Offset
MRBR	188h

41.5.1.24.2 Function

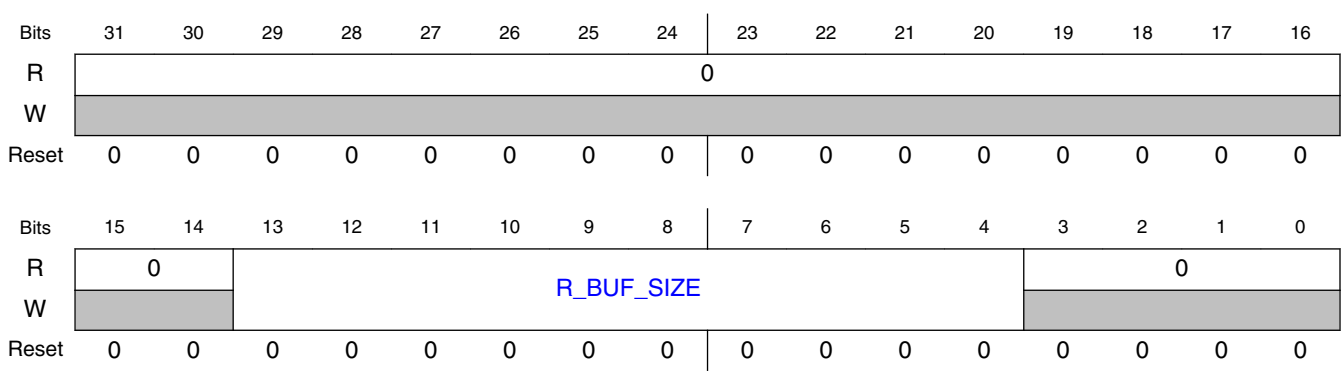
The MRBR is a user-programmable register that dictates the maximum size of all receive buffers. This value should take into consideration that the receive CRC is always written into the last receive buffer.

- R_BUF_SIZE is concatentated with the four least-significant bits of this register and are used as the maximum receive buffer size.
- To allow one maximum size frame per buffer, MRBR must be set to RCR[MAX_FL] or larger.
- To properly align the buffer, MRBR must be evenly divisible by 64. To ensure this, set the lower two bits of R_BUF_SIZE to zero. The lower four bits of this register are already set to zero by the device.
- To minimize bus usage (descriptor fetches), set MRBR greater than or equal to 256 bytes.

NOTE

This register must be initialized before operation.

41.5.1.24.3 Diagram



41.5.1.24.4 Fields

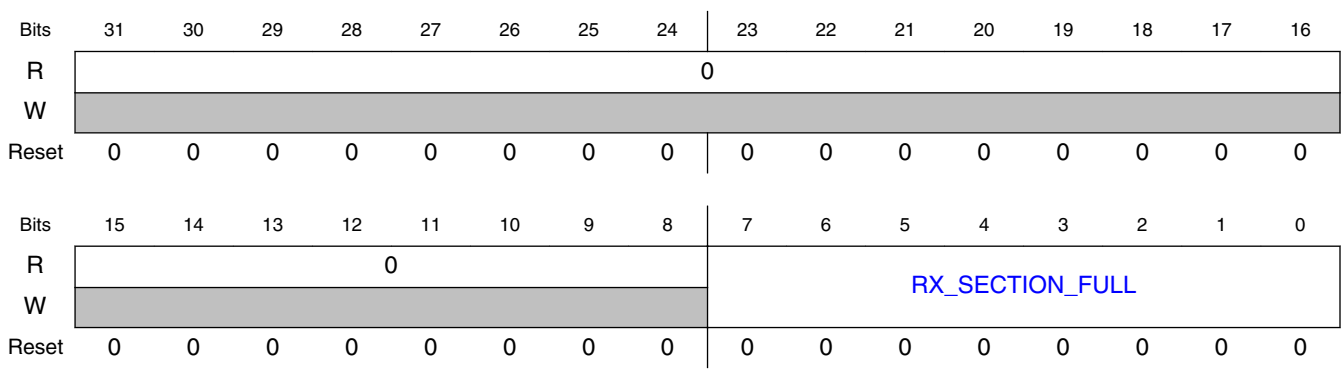
Field	Function
31-14 —	Reserved
13-4 R_BUF_SIZE	Receive buffer size in bytes. This value, concatenated with the four least-significant bits of this register (which are always zero), is the effective maximum receive buffer size.
3-0 —	This field, which is always zero, is the four least-significant bits of the maximum receive buffer size.

41.5.1.25 Receive FIFO Section Full Threshold (RSFL)

41.5.1.25.1 Offset

Register	Offset
RSFL	190h

41.5.1.25.2 Diagram



41.5.1.25.3 Fields

Field	Function
31-8 —	Reserved
7-0	Value Of Receive FIFO Section Full Threshold

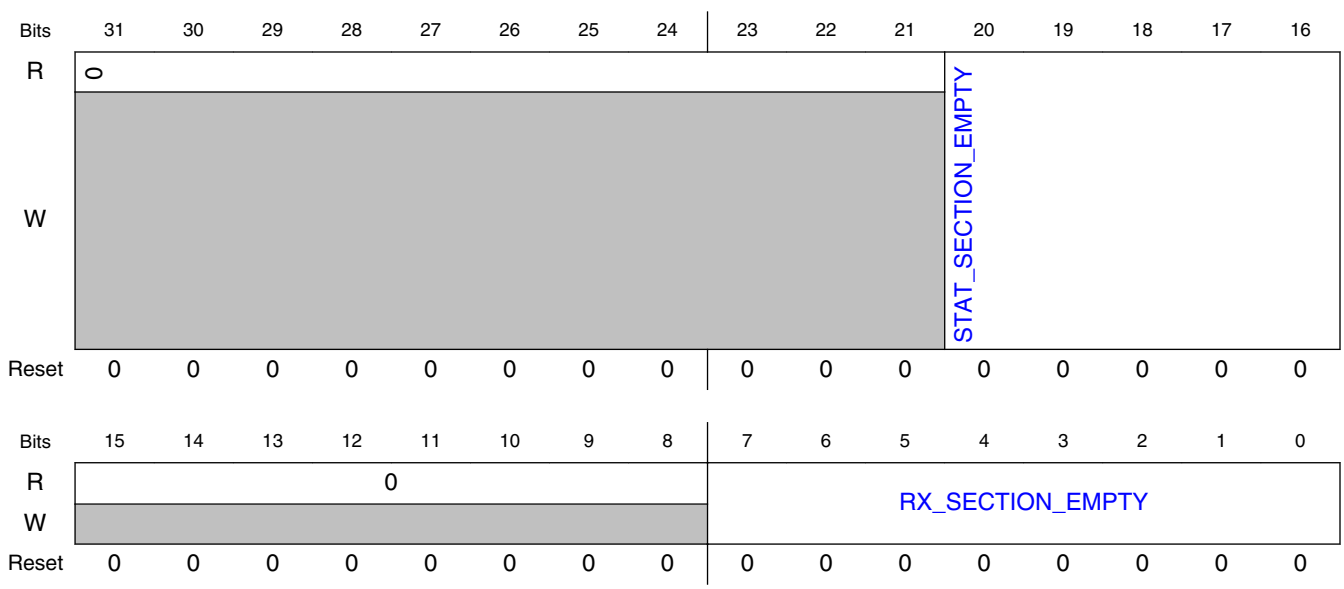
Field	Function
RX_SECTION_FULL	Value, in 64-bit words, of the receive FIFO section full threshold. Clear this field to enable store and forward on the RX FIFO. When programming a value greater than 0 (cut-through operation), it must be greater than RAEM[RX_ALMOST_EMPTY]. When the FIFO level reaches the value in this field, data is available in the Receive FIFO (cut-through operation).

41.5.1.26 Receive FIFO Section Empty Threshold (RSEM)

41.5.1.26.1 Offset

Register	Offset
RSEM	194h

41.5.1.26.2 Diagram



41.5.1.26.3 Fields

Field	Function
31-21 —	Reserved
20-16	RX Status FIFO Section Empty Threshold

Table continues on the next page...

Field	Function
STAT_SECTION_EMPTY	Defines number of frames in the receive FIFO, independent of its size, that can be accepted. If the limit is reached, reception will continue normally, however a pause frame will be triggered to indicate a possible congestion to the remote device to avoid FIFO overflow. A value of 0 disables automatic pause frame generation
15-8 —	Reserved
7-0 RX_SECTION_EMPTY	Value Of The Receive FIFO Section Empty Threshold Value, in 64-bit words, of the receive FIFO section empty threshold. When the FIFO has reached this level, a pause frame will be issued. A value of 0 disables automatic pause frame generation. When the FIFO level goes below the value programmed in this field, an XON pause frame is issued to indicate the FIFO congestion is cleared to the remote Ethernet client. **NOTE:** The section-empty threshold indications from both FIFOs are OR'ed to cause XOFF pause frame generation.

41.5.1.27 Receive FIFO Almost Empty Threshold (RAEM)

41.5.1.27.1 Offset

Register	Offset
RAEM	198h

41.5.1.27.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								RX_ALMOST_EMPTY							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

41.5.1.27.3 Fields

Field	Function
31-8	Reserved

Table continues on the next page...

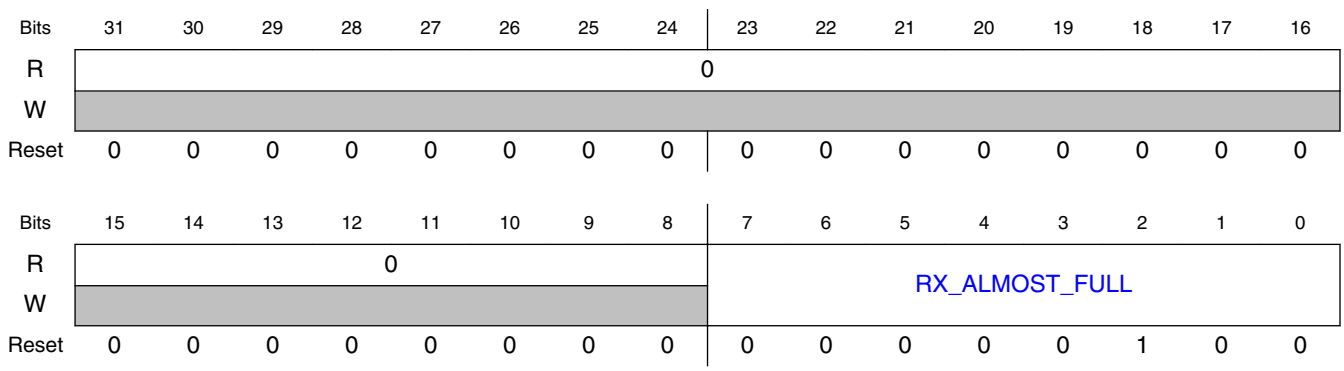
Field	Function
—	
7-0 RX_ALMOST_EMPTY	Value Of The Receive FIFO Almost Empty Threshold Value, in 64-bit words, of the receive FIFO almost empty threshold. When the FIFO level reaches the value programmed in this field and the end-of-frame has not been received for the frame yet, the core receive read control stops FIFO read (and subsequently stops transferring data to the MAC client application). It continues to deliver the frame, if again more data than the threshold or the end-of-frame is available in the FIFO. A minimum value of 4 should be set.

41.5.1.28 Receive FIFO Almost Full Threshold (RAFL)

41.5.1.28.1 Offset

Register	Offset
RAFL	19Ch

41.5.1.28.2 Diagram



41.5.1.28.3 Fields

Field	Function
31-8 —	Reserved
7-0 RX_ALMOST_FULL	Value Of The Receive FIFO Almost Full Threshold Value, in 64-bit words, of the receive FIFO almost full threshold. When the FIFO level comes close to the maximum, so that there is no more space for at least RX_ALMOST_FULL number of words, the MAC stops writing data in the FIFO and truncates the received frame to avoid FIFO overflow. The corresponding error status will be set when the frame is delivered to the application. A minimum value of 4 should be set.

41.5.1.29 Transmit FIFO Section Empty Threshold (TSEM)

41.5.1.29.1 Offset

Register	Offset
TSEM	1A0h

41.5.1.29.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								TX_SECTION_EMPTY							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

41.5.1.29.3 Fields

Field	Function
31-8 —	Reserved
7-0 TX_SECTION_EMPTY	Value Of The Transmit FIFO Section Empty Threshold Value, in 64-bit words, of the transmit FIFO section empty threshold. See Transmit FIFO for more information.

41.5.1.30 Transmit FIFO Almost Empty Threshold (TAEM)

41.5.1.30.1 Offset

Register	Offset
TAEM	1A4h

41.5.1.30.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								TX_ALMOST_EMPTY							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

41.5.1.30.3 Fields

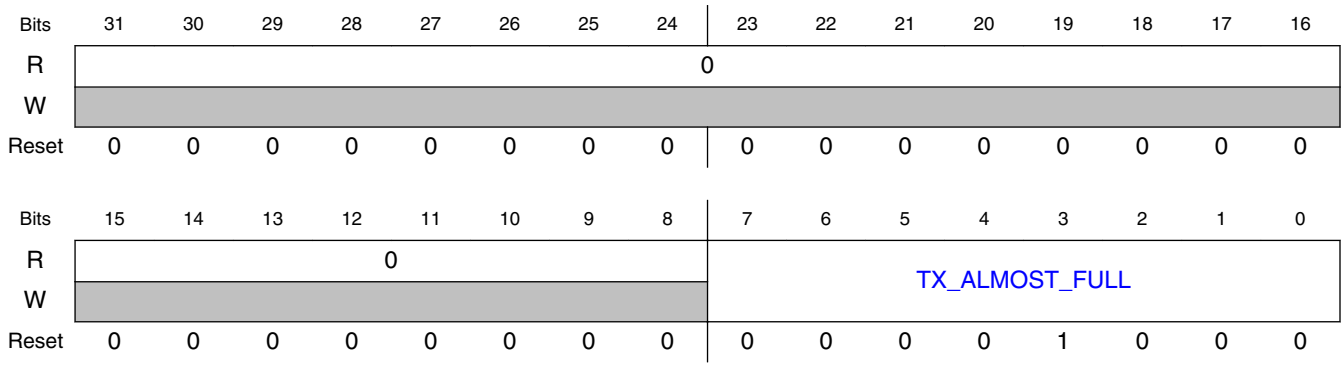
Field	Function
31-8 —	Reserved
7-0 TX_ALMOST_EMPTY	Value of Transmit FIFO Almost Empty Threshold Value, in 64-bit words, of the transmit FIFO almost empty threshold. When the FIFO level reaches the value programmed in this field, and no end-of-frame is available for the frame, the MAC transmit logic, to avoid FIFO underflow, stops reading the FIFO and transmits a frame with an MII error indication. See Transmit FIFO for more information. A minimum value of 4 should be set.

41.5.1.31 Transmit FIFO Almost Full Threshold (TAFL)

41.5.1.31.1 Offset

Register	Offset
TAFL	1A8h

41.5.1.31.2 Diagram



41.5.1.31.3 Fields

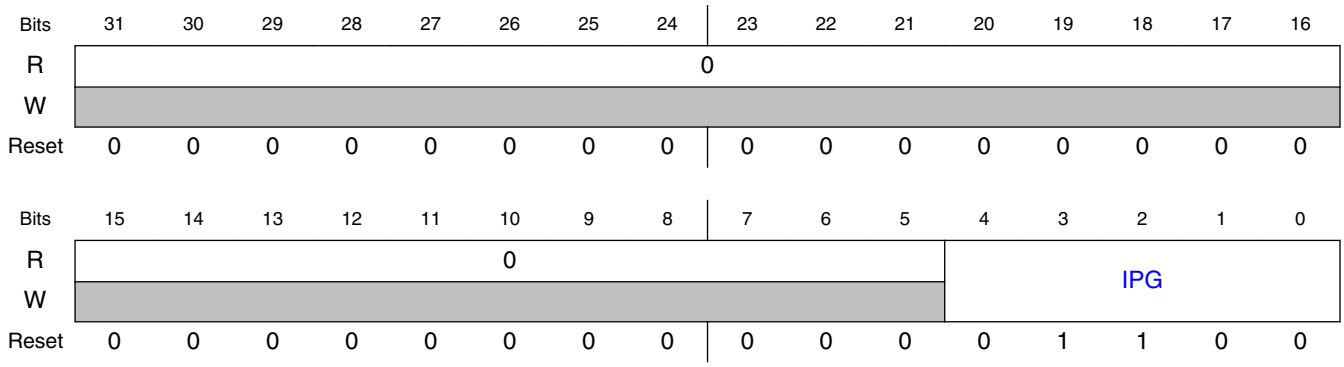
Field	Function
31-8 —	Reserved
7-0 TX_ALMOST_FULL	Value Of The Transmit FIFO Almost Full Threshold Value, in 64-bit words, of the transmit FIFO almost full threshold. A minimum value of six is required . A recommended value of at least 8 should be set allowing a latency of two clock cycles to the application. If more latency is required the value can be increased as necessary (latency = TAFL - 5). When the FIFO level comes close to the maximum, so that there is no more space for at least TX_ALMOST_FULL number of words, the pin ff_tx_rdy is deasserted. If the application does not react on this signal, the FIFO write control logic, to avoid FIFO overflow, truncates the current frame and sets the error status. As a result, the frame will be transmitted with an GMII/MII error indication. See Transmit FIFO for more information. **NOTE:** A FIFO overflow is a fatal error and requires a global reset on the transmit datapath or at least deassertion of ETHEREN.

41.5.1.32 Transmit Inter-Packet Gap (TIPG)

41.5.1.32.1 Offset

Register	Offset
TIPG	1ACh

41.5.1.32.2 Diagram



41.5.1.32.3 Fields

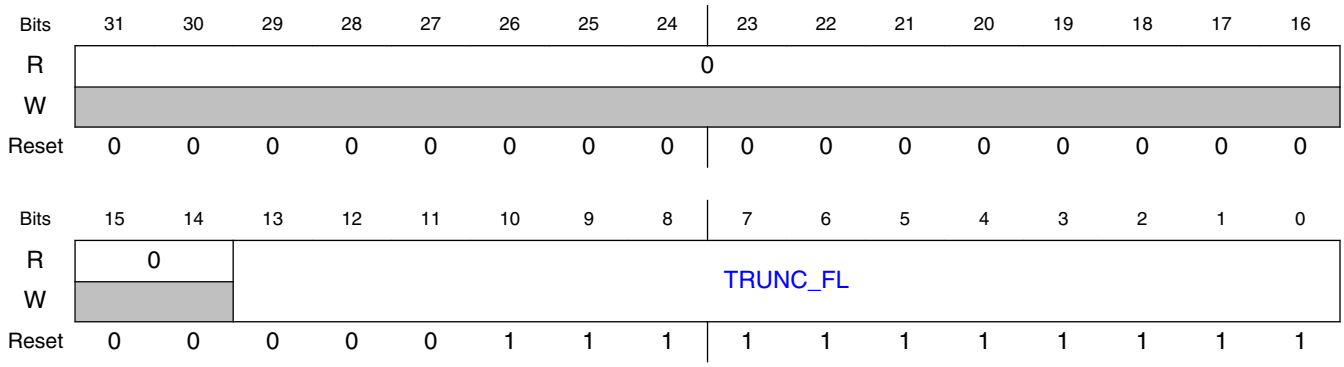
Field	Function
31-5 —	Reserved
4-0 IPG	Transmit Inter-Packet Gap Indicates the IPG, in bytes, between transmitted frames. Valid values range from 8 to 26. If the written value is less than 8 or greater than 26, the internal (effective) IPG is 12. **NOTE:** The IPG value read will be the value that was written, even if it is out of range.

41.5.1.33 Frame Truncation Length (FTRL)

41.5.1.33.1 Offset

Register	Offset
FTRL	1B0h

41.5.1.33.2 Diagram



41.5.1.33.3 Fields

Field	Function
31-14 —	Reserved
13-0 TRUNC_FL	Frame Truncation Length Indicates the value a receive frame is truncated, if it is greater than this value. Must be greater than or equal to RCR[MAX_FL]. **NOTE:** Truncation happens at TRUNC_FL. However, when truncation occurs, the application (FIFO) may receive less data, guaranteeing that it never receives more than the set limit.

41.5.1.34 Transmit Accelerator Function Configuration (TACC)

41.5.1.34.1 Offset

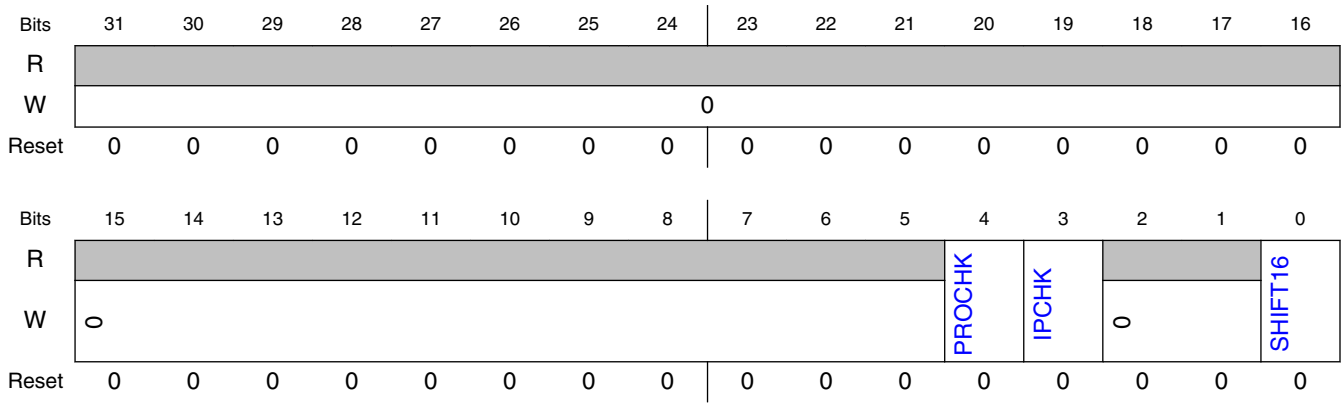
Register	Offset
TACC	1C0h

41.5.1.34.2 Function

TACC controls accelerator actions when sending frames. The register can be changed before or after each frame, but it must remain unmodified during frame writes into the transmit FIFO.

The TFWR[STRFWD] field must be set to use the checksum feature.

41.5.1.34.3 Diagram



41.5.1.34.4 Fields

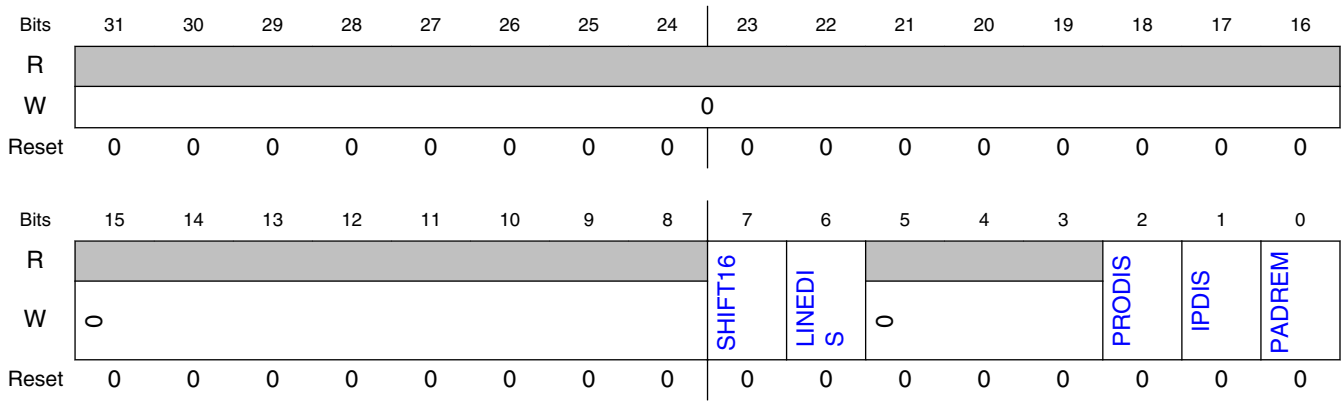
Field	Function
31-5 —	This write-only field is reserved. It must always be written with the value 0.
4 PROCHK	Enables insertion of protocol checksum. 0b - Checksum not inserted. 1b - If an IP frame with a known protocol is transmitted, the checksum is inserted automatically into the frame. The checksum field must be cleared. The other frames are not modified.
3 IPCHK	Enables insertion of IP header checksum. 0b - Checksum is not inserted. 1b - If an IP frame is transmitted, the checksum is inserted automatically. The IP header checksum field must be cleared. If a non-IP frame is transmitted the frame is not modified.
2-1 —	This write-only field is reserved. It must always be written with the value 0.
0 SHIFT16	TX FIFO Shift-16 0b - Disabled. 1b - Indicates to the transmit data FIFO that the written frames contain two additional octets before the frame data. This means the actual frame begins at bit 16 of the first word written into the FIFO. This function allows putting the frame payload on a 32-bit boundary in memory, as the 14-byte Ethernet header is extended to a 16-byte header.

41.5.1.35 Receive Accelerator Function Configuration (RACC)

41.5.1.35.1 Offset

Register	Offset
RACC	1C4h

41.5.1.35.2 Diagram



41.5.1.35.3 Fields

Field	Function
31-8 —	This write-only field is reserved. It must always be written with the value 0.
7 SHIFT16	RX FIFO Shift-16 When this field is set, the actual frame data starts at bit 16 of the first word read from the RX FIFO aligning the Ethernet payload on a 32-bit boundary. **NOTE:** This function only affects the FIFO storage and has no influence on the statistics, which use the actual length of the frame received. 0b - Disabled. 1b - Instructs the MAC to write two additional bytes in front of each frame received into the RX FIFO.
6 LINEDIS	Enable Discard Of Frames With MAC Layer Errors 0b - Frames with errors are not discarded. 1b - Any frame received with a CRC, length, or PHY error is automatically discarded and not forwarded to the user application interface.
5-3 —	This write-only field is reserved. It must always be written with the value 0.
2 PRODIS	Enable Discard Of Frames With Wrong Protocol Checksum 0b - Frames with wrong checksum are not discarded. 1b - If a TCP/IP, UDP/IP, or ICMP/IP frame is received that has a wrong TCP, UDP, or ICMP checksum, the frame is discarded. Discarding is only available when the RX FIFO operates in store and forward mode (RSFL cleared).
1 IPDIS	Enable Discard Of Frames With Wrong IPv4 Header Checksum 0b - Frames with wrong IPv4 header checksum are not discarded. 1b - If an IPv4 frame is received with a mismatching header checksum, the frame is discarded. IPv6 has no header checksum and is not affected by this setting. Discarding is only available when the RX FIFO operates in store and forward mode (RSFL cleared).
0 PADREM	Enable Padding Removal For Short IP Frames 0b - Padding not removed. 1b - Any bytes following the IP payload section of the frame are removed from the frame.

41.5.1.36 Tx Packet Count Statistic Register (RMON_T_PACKETS)

41.5.1.36.1 Offset

Register	Offset
RMON_T_PACKETS	204h

41.5.1.36.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TXPKTS															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

41.5.1.36.3 Fields

Field	Function
31-16 —	Reserved
15-0 TXPKTS	Packet count Transmit packet count

41.5.1.37 Tx Broadcast Packets Statistic Register (RMON_T_BC_PKT)

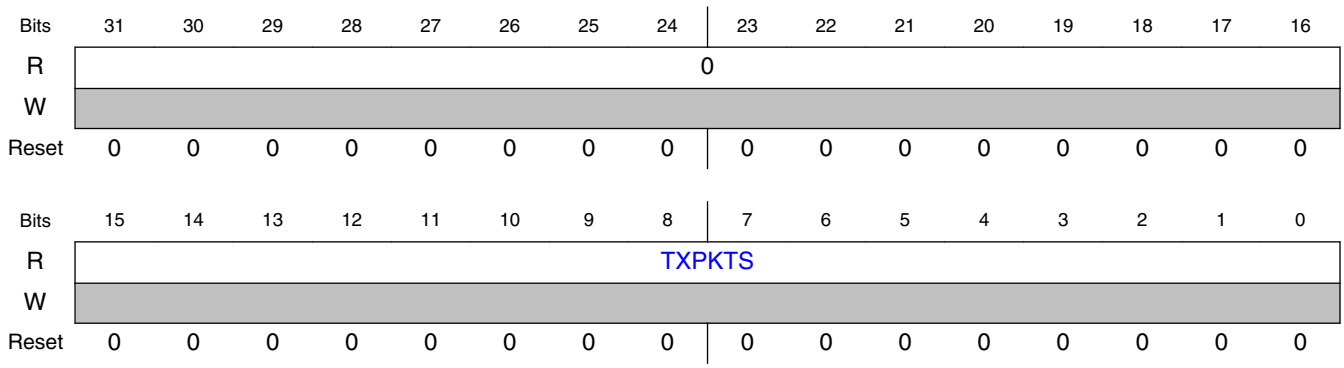
41.5.1.37.1 Offset

Register	Offset
RMON_T_BC_PKT	208h

41.5.1.37.2 Function

RMON Tx Broadcast Packets

41.5.1.37.3 Diagram



41.5.1.37.4 Fields

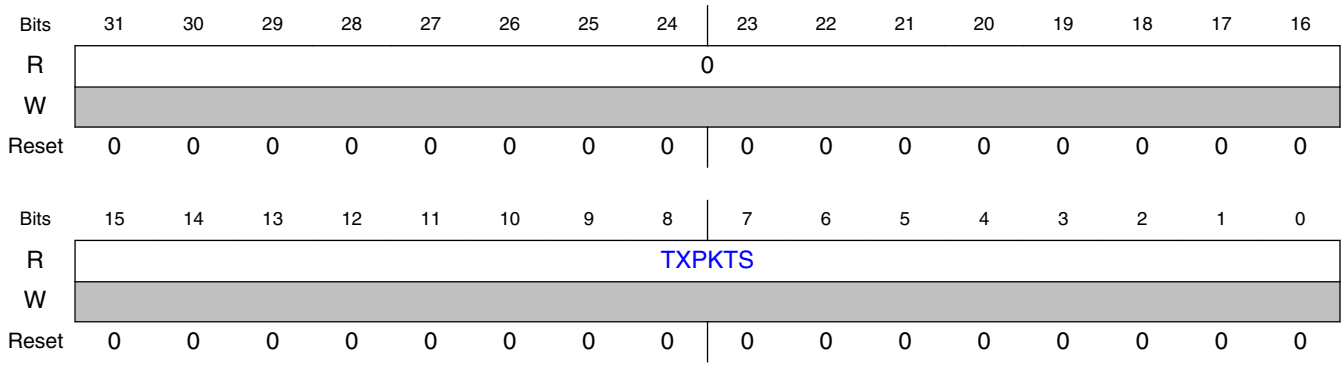
Field	Function
31-16 —	Reserved
15-0 TXPKTS	Number of broadcast packets

41.5.1.38 Tx Multicast Packets Statistic Register (RMON_T_MC_PKT)

41.5.1.38.1 Offset

Register	Offset
RMON_T_MC_PKT	20Ch

41.5.1.38.2 Diagram



41.5.1.38.3 Fields

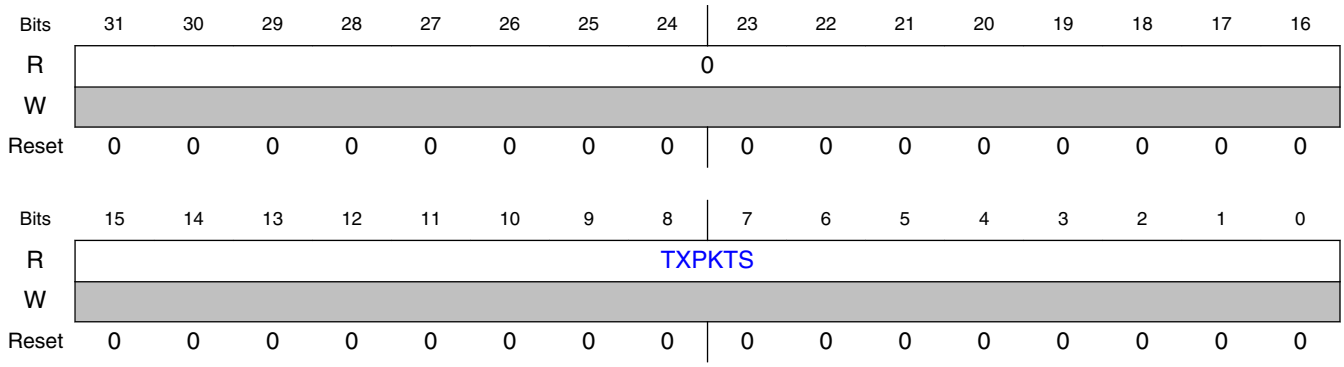
Field	Function
31-16 —	Reserved
15-0 TXPKTS	Number of multicast packets

41.5.1.39 Tx Packets with CRC/Align Error Statistic Register (RMON_T_CRC_ALIGN)

41.5.1.39.1 Offset

Register	Offset
RMON_T_CRC_ALIGN	210h

41.5.1.39.2 Diagram



41.5.1.39.3 Fields

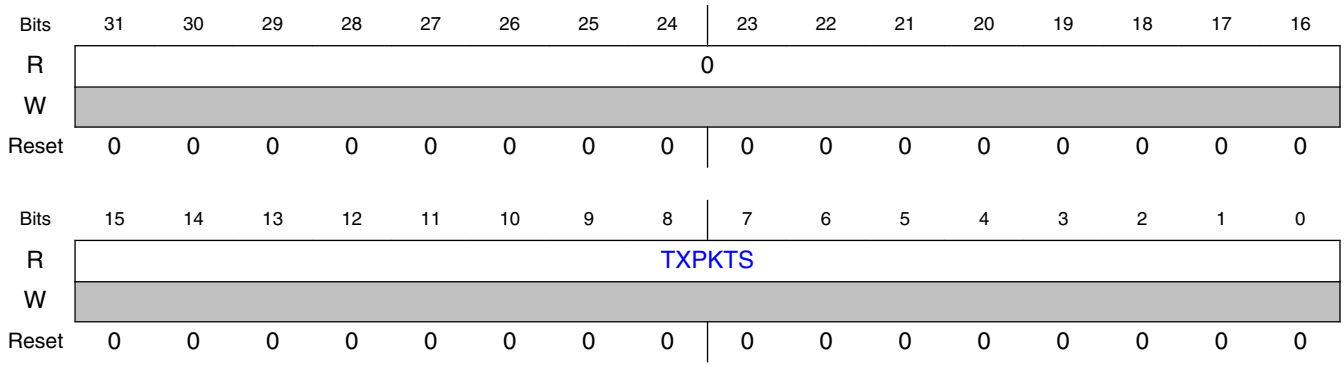
Field	Function
31-16 —	Reserved
15-0 TXPKTS	Number of packets with CRC/align error

41.5.1.40 Tx Packets Less Than Bytes and Good CRC Statistic Register (RMON_T_UNDERSIZE)

41.5.1.40.1 Offset

Register	Offset
RMON_T_UNDERSIZE	214h

41.5.1.40.2 Diagram



41.5.1.40.3 Fields

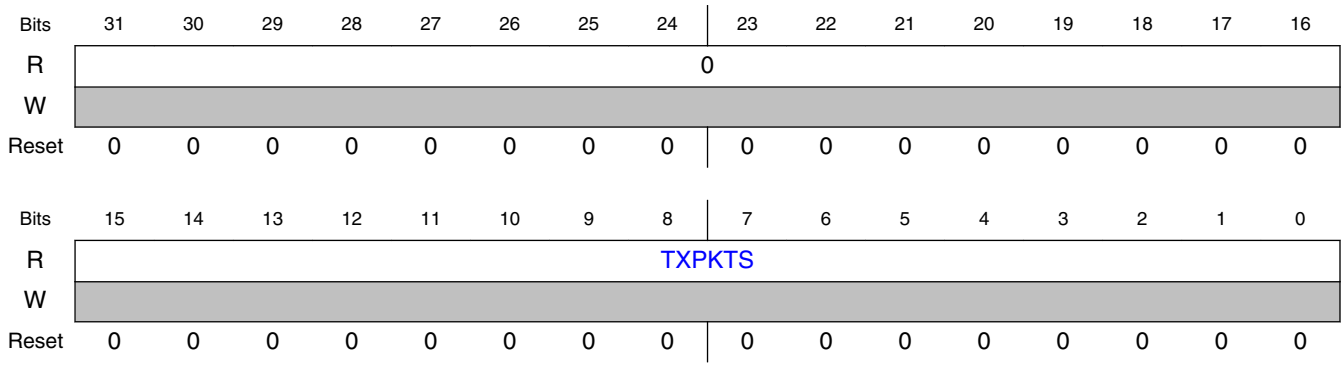
Field	Function
31-16 —	Reserved
15-0 TXPKTS	Number of transmit packets less than 64 bytes with good CRC

41.5.1.41 Tx Packets GT MAX_FL bytes and Good CRC Statistic Register (RMON_T_OVERSIZE)

41.5.1.41.1 Offset

Register	Offset
RMON_T_OVERSIZE	218h

41.5.1.41.2 Diagram



41.5.1.41.3 Fields

Field	Function
31-16 —	Reserved
15-0 TXPKTS	Number of transmit packets greater than MAX_FL bytes with good CRC

41.5.1.42 Tx Packets Less Than 64 Bytes and Bad CRC Statistic Register (RMON_T_FRAG)

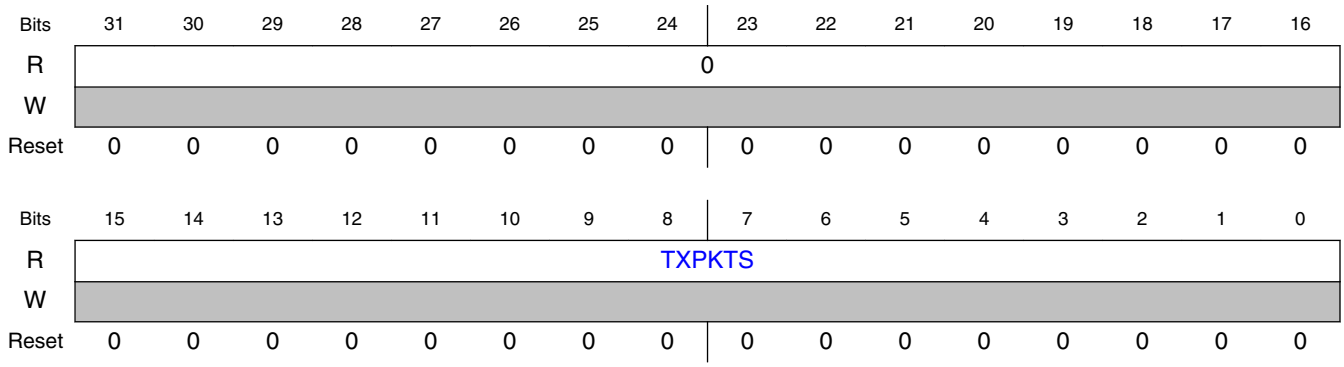
41.5.1.42.1 Offset

Register	Offset
RMON_T_FRAG	21Ch

41.5.1.42.2 Function

.

41.5.1.42.3 Diagram



41.5.1.42.4 Fields

Field	Function
31-16 —	Reserved
15-0 TXPKTS	Number of packets less than 64 bytes with bad CRC

41.5.1.43 Tx Packets Greater Than MAX_FL bytes and Bad CRC Statistic Register (RMON_T_JAB)

41.5.1.43.1 Offset

Register	Offset
RMON_T_JAB	220h

41.5.1.43.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TXPKTS															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

41.5.1.43.3 Fields

Field	Function
31-16 —	Reserved
15-0 TXPKTS	Number of transmit packets greater than MAX_FL bytes and bad CRC

41.5.1.44 Tx Collision Count Statistic Register (RMON_T_COL)

41.5.1.44.1 Offset

Register	Offset
RMON_T_COL	224h

41.5.1.44.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TXPKTS															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

41.5.1.44.3 Fields

Field	Function
31-16 —	Reserved
15-0 TXPKTS	Number of transmit collisions

41.5.1.45 Tx 64-Byte Packets Statistic Register (RMON_T_P64)

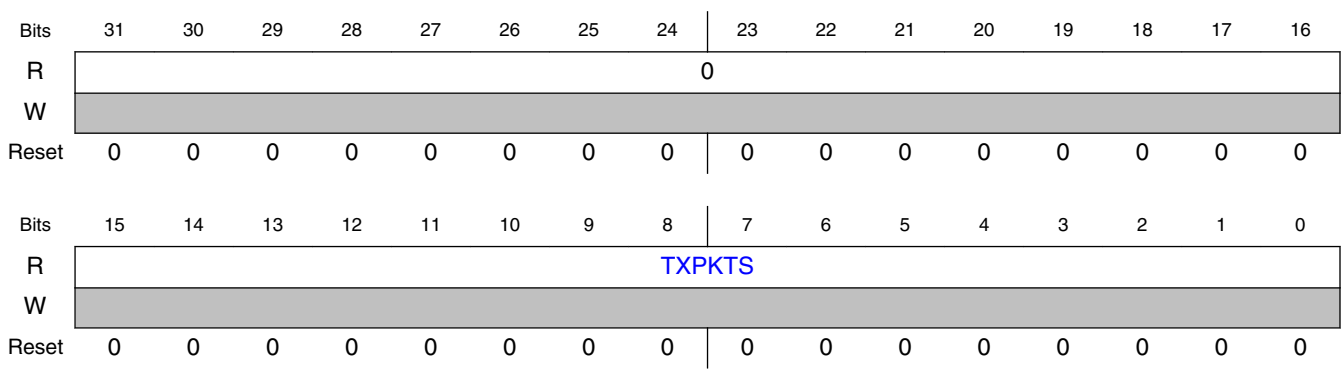
41.5.1.45.1 Offset

Register	Offset
RMON_T_P64	228h

41.5.1.45.2 Function

.

41.5.1.45.3 Diagram



41.5.1.45.4 Fields

Field	Function
31-16	Reserved

Table continues on the next page...

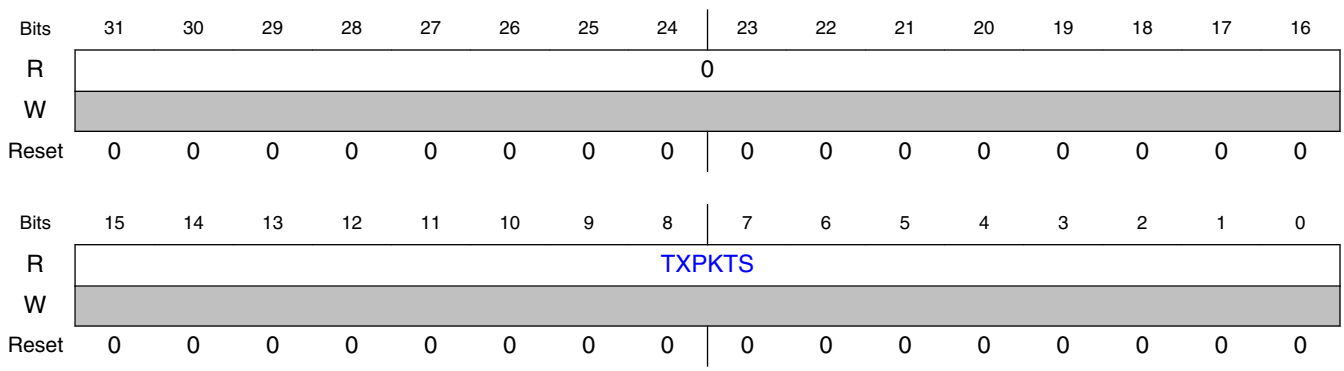
Field	Function
—	
15-0 TXPKTS	Number of 64-byte transmit packets

41.5.1.46 Tx 65- to 127-byte Packets Statistic Register (RMON_T_P65TO127)

41.5.1.46.1 Offset

Register	Offset
RMON_T_P65TO127	22Ch

41.5.1.46.2 Diagram



41.5.1.46.3 Fields

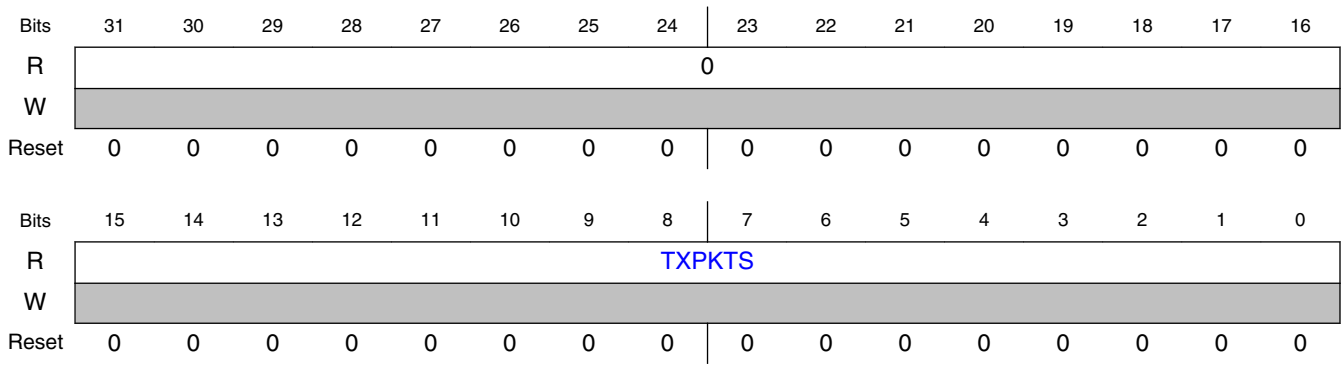
Field	Function
31-16 —	Reserved
15-0 TXPKTS	Number of 65- to 127-byte transmit packets

41.5.1.47 Tx 128- to 255-byte Packets Statistic Register (RMON_T_P128TO255)

41.5.1.47.1 Offset

Register	Offset
RMON_T_P128TO255	230h

41.5.1.47.2 Diagram



41.5.1.47.3 Fields

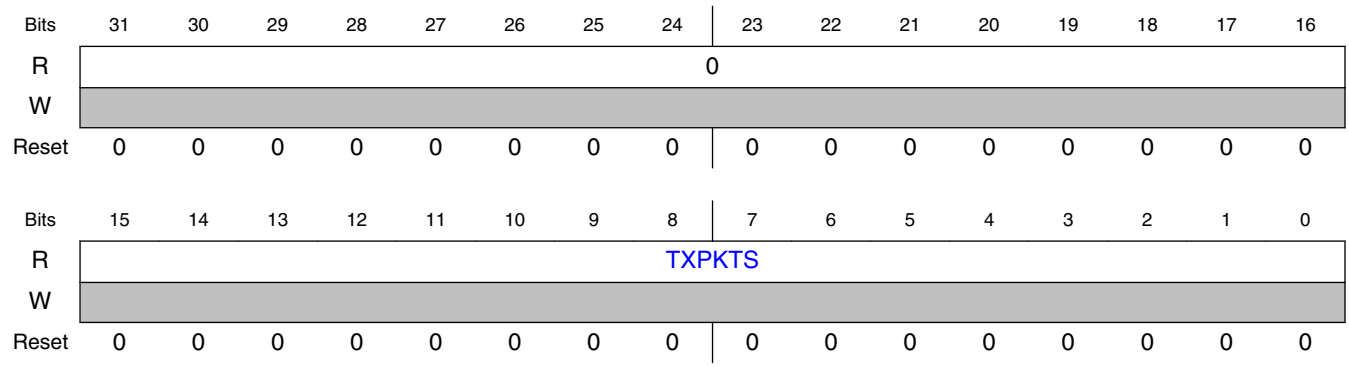
Field	Function
31-16 —	Reserved
15-0 TXPKTS	Number of 128- to 255-byte transmit packets

41.5.1.48 Tx 256- to 511-byte Packets Statistic Register (RMON_T_P256TO511)

41.5.1.48.1 Offset

Register	Offset
RMON_T_P256TO511	234h

41.5.1.48.2 Diagram



41.5.1.48.3 Fields

Field	Function
31-16 —	Reserved
15-0 TXPKTS	Number of 256- to 511-byte transmit packets

41.5.1.49 Tx 512- to 1023-byte Packets Statistic Register (RMON_T_P512TO1023)

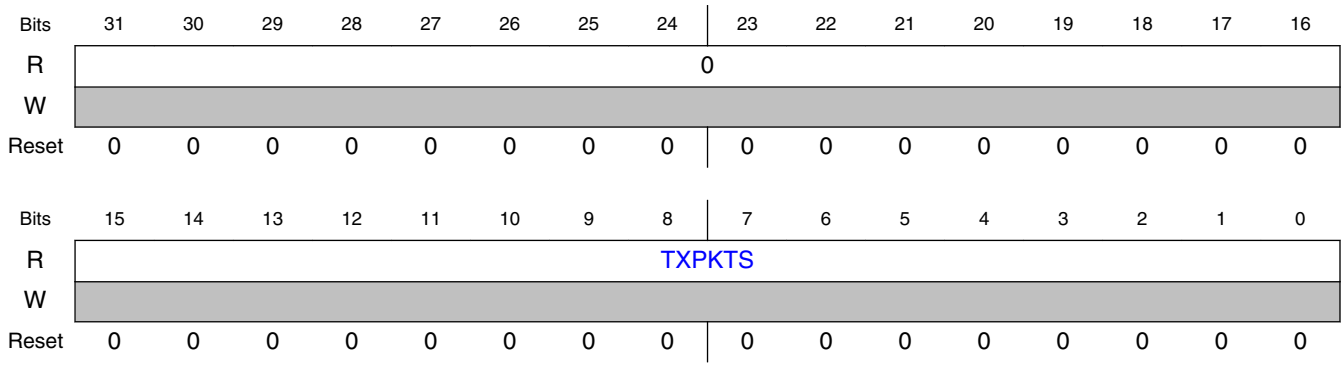
41.5.1.49.1 Offset

Register	Offset
RMON_T_P512TO1023	238h

41.5.1.49.2 Function

.

41.5.1.49.3 Diagram



41.5.1.49.4 Fields

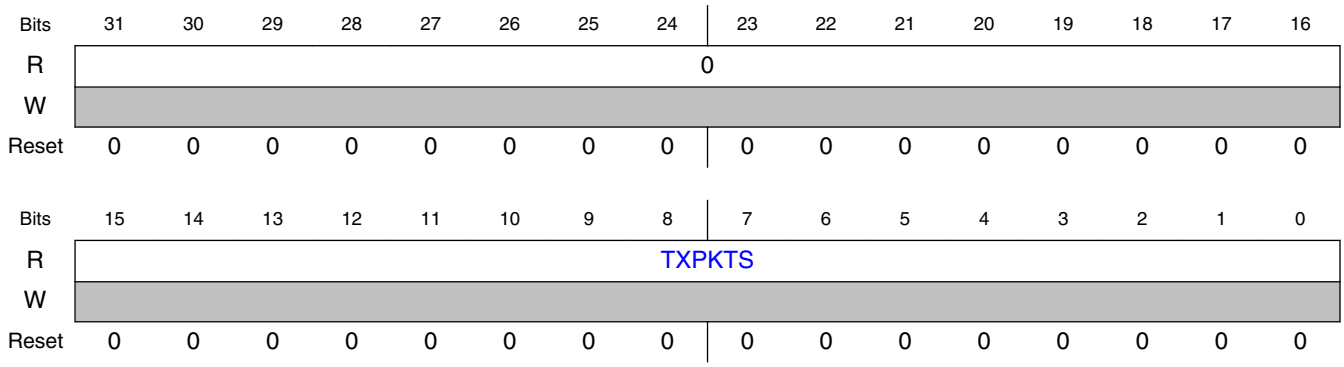
Field	Function
31-16 —	Reserved
15-0 TXPKTS	Number of 512- to 1023-byte transmit packets

41.5.1.50 Tx 1024- to 2047-byte Packets Statistic Register (RMON_T_P1024TO2047)

41.5.1.50.1 Offset

Register	Offset
RMON_T_P1024TO2047	23Ch

41.5.1.50.2 Diagram



41.5.1.50.3 Fields

Field	Function
31-16 —	Reserved
15-0 TXPKTS	Number of 1024- to 2047-byte transmit packets

41.5.1.51 Tx Packets Greater Than 2048 Bytes Statistic Register (RMON_T_P_GTE2048)

41.5.1.51.1 Offset

Register	Offset
RMON_T_P_GTE2048	240h

41.5.1.51.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TXPKTS															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

41.5.1.51.3 Fields

Field	Function
31-16 —	Reserved
15-0 TXPKTS	Number of transmit packets greater than 2048 bytes

41.5.1.52 Tx Octets Statistic Register (RMON_T_OCTETS)

41.5.1.52.1 Offset

Register	Offset
RMON_T_OCTETS	244h

41.5.1.52.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TXOCTS															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TXOCTS															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

41.5.1.52.3 Fields

Field	Function
31-0 TXOCTS	Number of transmit octets

41.5.1.53 Frames Transmitted OK Statistic Register (IEEE_T_FRAME_OK)

41.5.1.53.1 Offset

Register	Offset
IEEE_T_FRAME_OK	24Ch

41.5.1.53.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	COUNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

41.5.1.53.3 Fields

Field	Function
31-16 —	Reserved
15-0 COUNT	Number of frames transmitted OK **NOTE:** Does not increment for the broadcast frames when broadcast reject is enabled and promiscuous mode is disabled within the receive control register (RCR).

41.5.1.54 Frames Transmitted with Single Collision Statistic Register (IEEE_T_1COL)

41.5.1.54.1 Offset

Register	Offset
IEEE_T_1COL	250h

41.5.1.54.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	COUNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

41.5.1.54.3 Fields

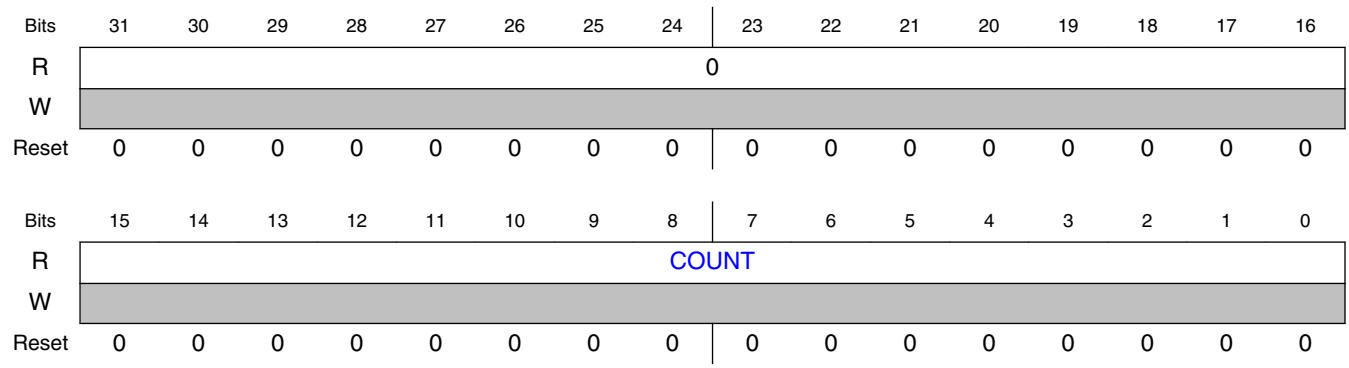
Field	Function
31-16 —	Reserved
15-0 COUNT	Number of frames transmitted with one collision

41.5.1.55 Frames Transmitted with Multiple Collisions Statistic Register (IEEE_T_MCOL)

41.5.1.55.1 Offset

Register	Offset
IEEE_T_MCOL	254h

41.5.1.55.2 Diagram



41.5.1.55.3 Fields

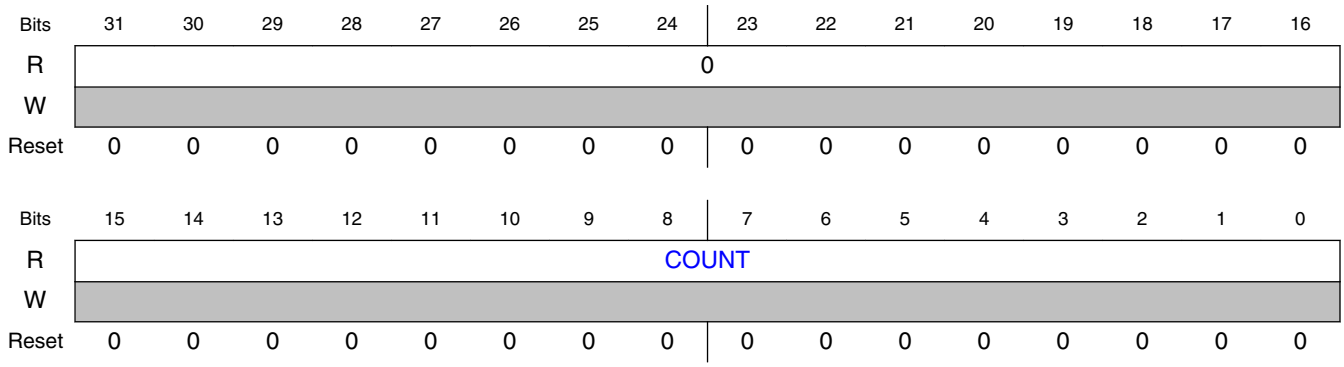
Field	Function
31-16 —	Reserved
15-0 COUNT	Number of frames transmitted with multiple collisions

41.5.1.56 Frames Transmitted after Deferral Delay Statistic Register (IEEE_T_DEF)

41.5.1.56.1 Offset

Register	Offset
IEEE_T_DEF	258h

41.5.1.56.2 Diagram



41.5.1.56.3 Fields

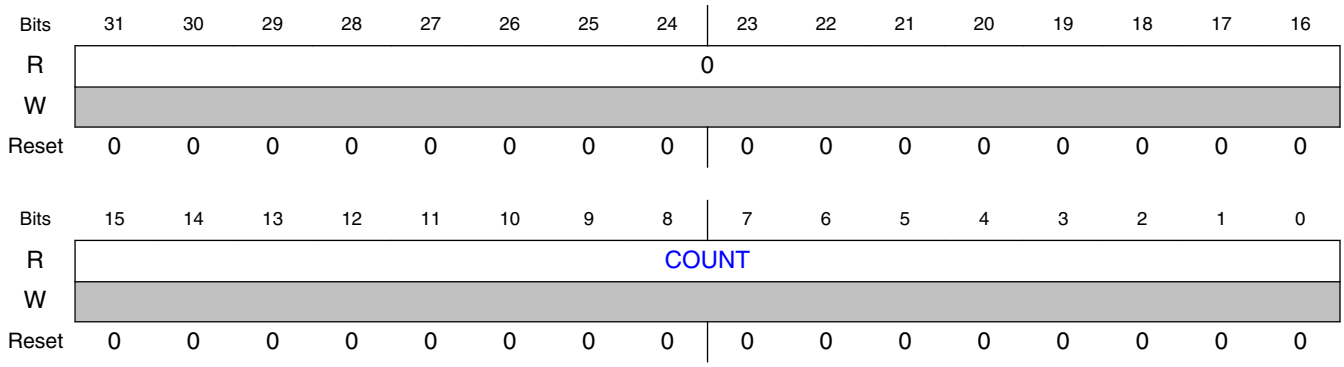
Field	Function
31-16 —	Reserved
15-0 COUNT	Number of frames transmitted with deferral delay

41.5.1.57 Frames Transmitted with Late Collision Statistic Register (IEEE_T_LCOL)

41.5.1.57.1 Offset

Register	Offset
IEEE_T_LCOL	25Ch

41.5.1.57.2 Diagram



41.5.1.57.3 Fields

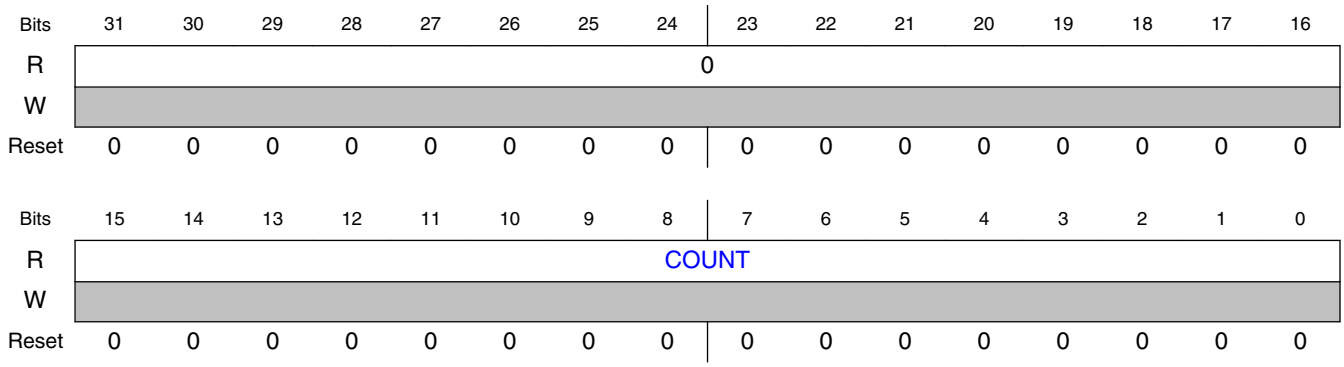
Field	Function
31-16 —	Reserved
15-0 COUNT	Number of frames transmitted with late collision

41.5.1.58 Frames Transmitted with Excessive Collisions Statistic Register (IEEE_T_EXCOL)

41.5.1.58.1 Offset

Register	Offset
IEEE_T_EXCOL	260h

41.5.1.58.2 Diagram



41.5.1.58.3 Fields

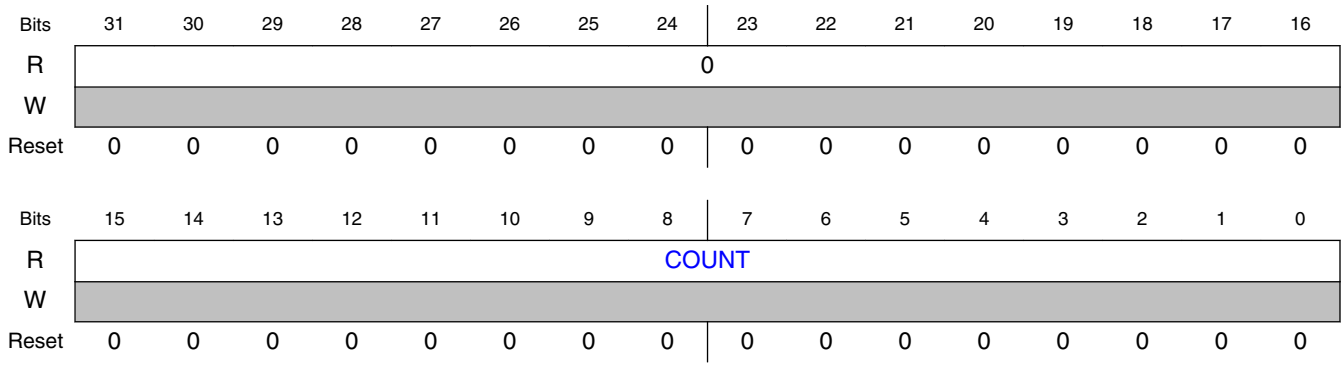
Field	Function
31-16 —	Reserved
15-0 COUNT	Number of frames transmitted with excessive collisions

41.5.1.59 Frames Transmitted with Tx FIFO Underrun Statistic Register (IEEE_T_MACERR)

41.5.1.59.1 Offset

Register	Offset
IEEE_T_MACERR	264h

41.5.1.59.2 Diagram



41.5.1.59.3 Fields

Field	Function
31-16 —	Reserved
15-0 COUNT	Number of frames transmitted with transmit FIFO underrun

41.5.1.60 Frames Transmitted with Carrier Sense Error Statistic Register (IEEE_T_CSERR)

41.5.1.60.1 Offset

Register	Offset
IEEE_T_CSERR	268h

41.5.1.60.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	COUNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

41.5.1.60.3 Fields

Field	Function
31-16 —	Reserved
15-0 COUNT	Number of frames transmitted with carrier sense error

41.5.1.61 Reserved Statistic Register (IEEE_T_SQE)

41.5.1.61.1 Offset

Register	Offset
IEEE_T_SQE	26Ch

41.5.1.61.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	COUNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

41.5.1.61.3 Fields

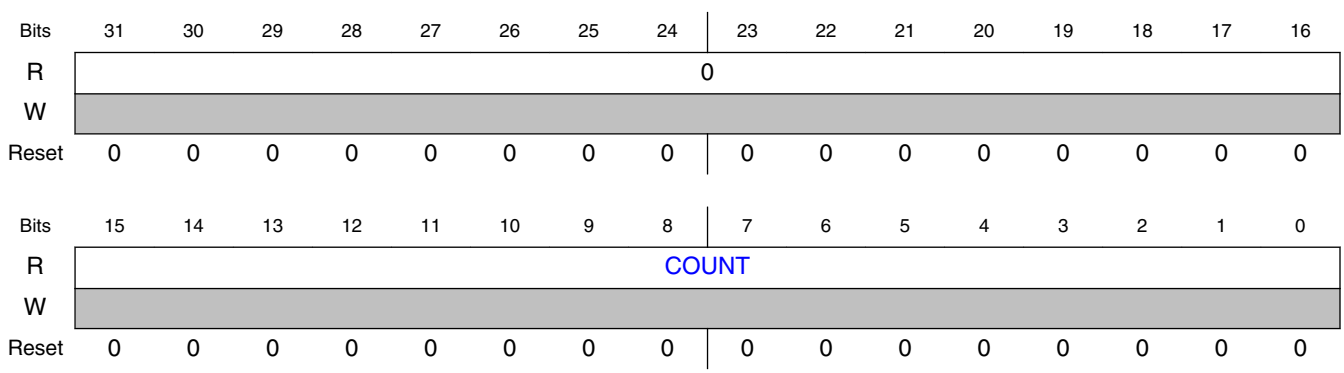
Field	Function
31-16 —	Reserved
15-0 COUNT	This read-only field is reserved and always has the value 0 This read-only field is reserved and always has the value 0. **NOTE:** Counter not implemented as no SQE information is available.

41.5.1.62 Flow Control Pause Frames Transmitted Statistic Register (IEEE_T_FDXFC)

41.5.1.62.1 Offset

Register	Offset
IEEE_T_FDXFC	270h

41.5.1.62.2 Diagram



41.5.1.62.3 Fields

Field	Function
31-16 —	Reserved

Table continues on the next page...

Field	Function
15-0 COUNT	Number of flow-control pause frames transmitted

41.5.1.63 Octet Count for Frames Transmitted w/o Error Statistic Register (IEEE_T_OCTETS_OK)

41.5.1.63.1 Offset

Register	Offset
IEEE_T_OCTETS_OK	274h

41.5.1.63.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	COUNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	COUNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

41.5.1.63.3 Fields

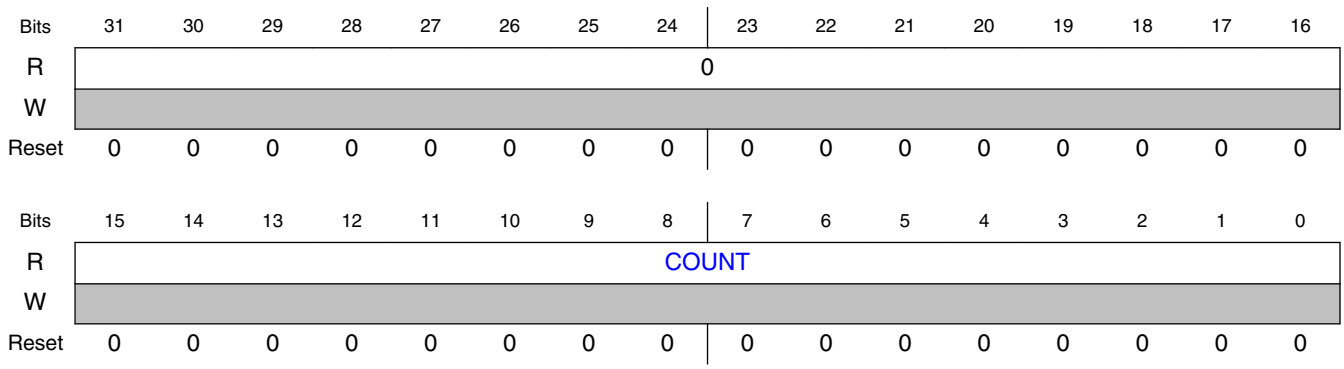
Field	Function
31-0 COUNT	Octet count for frames transmitted without error Counts total octets (includes header and FCS fields).

41.5.1.64 Rx Packet Count Statistic Register (RMON_R_PACKETS)

41.5.1.64.1 Offset

Register	Offset
RMON_R_PACKETS	284h

41.5.1.64.2 Diagram



41.5.1.64.3 Fields

Field	Function
31-16 —	Reserved
15-0 COUNT	Number of packets received

41.5.1.65 Rx Broadcast Packets Statistic Register (RMON_R_BC_PKT)

41.5.1.65.1 Offset

Register	Offset
RMON_R_BC_PKT	288h

41.5.1.65.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	COUNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

41.5.1.65.3 Fields

Field	Function
31-16 —	Reserved
15-0 COUNT	Number of receive broadcast packets

41.5.1.66 Rx Multicast Packets Statistic Register (RMON_R_MC_PKT)

41.5.1.66.1 Offset

Register	Offset
RMON_R_MC_PKT	28Ch

41.5.1.66.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	COUNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

41.5.1.66.3 Fields

Field	Function
31-16 —	Reserved
15-0 COUNT	Number of receive multicast packets

41.5.1.67 Rx Packets with CRC/Align Error Statistic Register (RMON_R_CRC_ALIGN)

41.5.1.67.1 Offset

Register	Offset
RMON_R_CRC_ALIGN	290h

41.5.1.67.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	COUNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

41.5.1.67.3 Fields

Field	Function
31-16 —	Reserved
15-0 COUNT	Number of receive packets with CRC or align error

41.5.1.68 Rx Packets with Less Than 64 Bytes and Good CRC Statistic Register (RMON_R_UNDERSIZE)

41.5.1.68.1 Offset

Register	Offset
RMON_R_UNDERSIZE	294h

41.5.1.68.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	COUNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

41.5.1.68.3 Fields

Field	Function
31-16 —	Reserved
15-0 COUNT	Number of receive packets with less than 64 bytes and good CRC

41.5.1.69 Rx Packets Greater Than MAX_FL and Good CRC Statistic Register (RMON_R_OVERSIZE)

41.5.1.69.1 Offset

Register	Offset
RMON_R_OVERSIZE	298h

41.5.1.69.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	COUNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

41.5.1.69.3 Fields

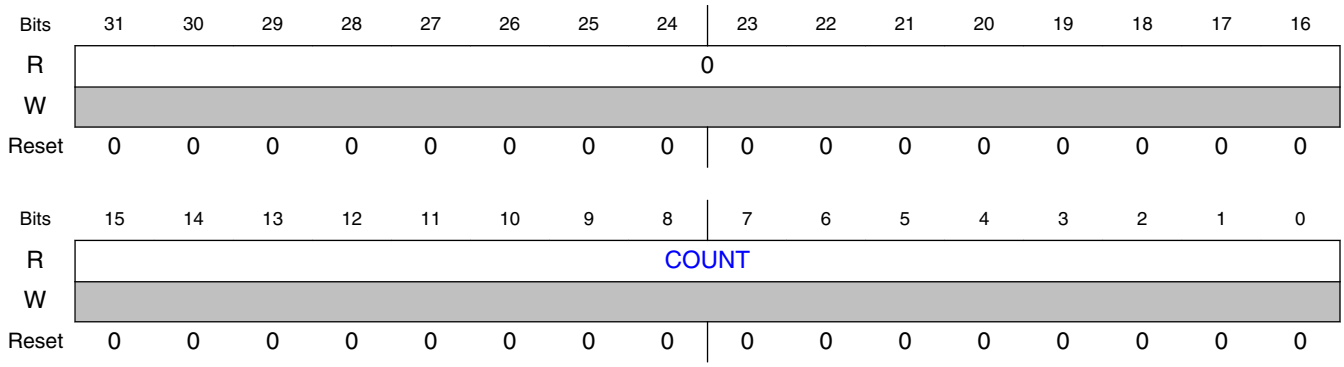
Field	Function
31-16 —	Reserved
15-0 COUNT	Number of receive packets greater than MAX_FL and good CRC

41.5.1.70 Rx Packets Less Than 64 Bytes and Bad CRC Statistic Register (RMON_R_FRAG)

41.5.1.70.1 Offset

Register	Offset
RMON_R_FRAG	29Ch

41.5.1.70.2 Diagram



41.5.1.70.3 Fields

Field	Function
31-16 —	Reserved
15-0 COUNT	Number of receive packets with less than 64 bytes and bad CRC

41.5.1.71 Rx Packets Greater Than MAX_FL Bytes and Bad CRC Statistic Register (RMON_R_JAB)

41.5.1.71.1 Offset

Register	Offset
RMON_R_JAB	2A0h

41.5.1.71.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	COUNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

41.5.1.71.3 Fields

Field	Function
31-16 —	Reserved
15-0 COUNT	Number of receive packets greater than MAX_FL and bad CRC

41.5.1.72 Rx 64-Byte Packets Statistic Register (RMON_R_P64)

41.5.1.72.1 Offset

Register	Offset
RMON_R_P64	2A8h

41.5.1.72.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	COUNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

41.5.1.72.3 Fields

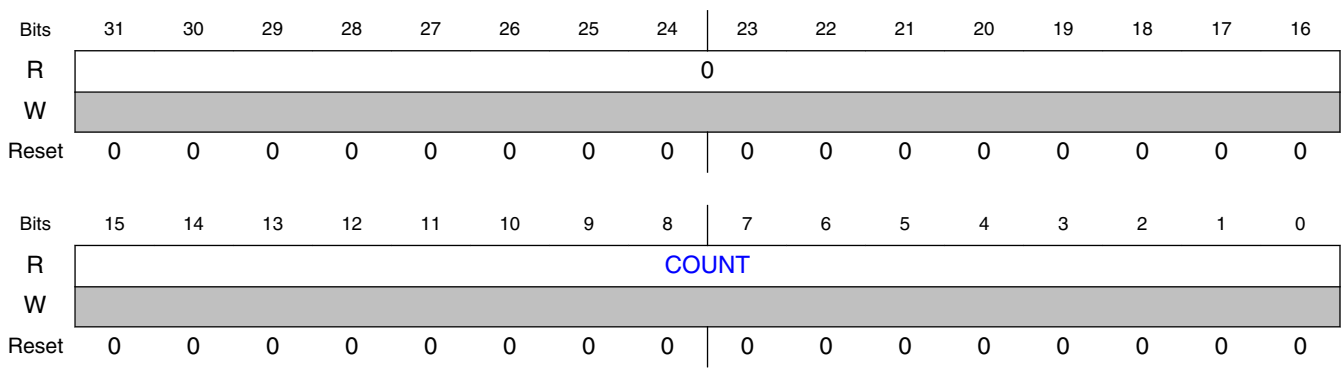
Field	Function
31-16 —	Reserved
15-0 COUNT	Number of 64-byte receive packets

41.5.1.73 Rx 65- to 127-Byte Packets Statistic Register (RMON_R_P65TO127)

41.5.1.73.1 Offset

Register	Offset
RMON_R_P65TO127	2ACh

41.5.1.73.2 Diagram



41.5.1.73.3 Fields

Field	Function
31-16 —	Reserved
15-0 COUNT	Number of 65- to 127-byte recieve packets

41.5.1.74 Rx 128- to 255-Byte Packets Statistic Register (RMON_R_P128TO255)

41.5.1.74.1 Offset

Register	Offset
RMON_R_P128TO255	2B0h

41.5.1.74.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	COUNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

41.5.1.74.3 Fields

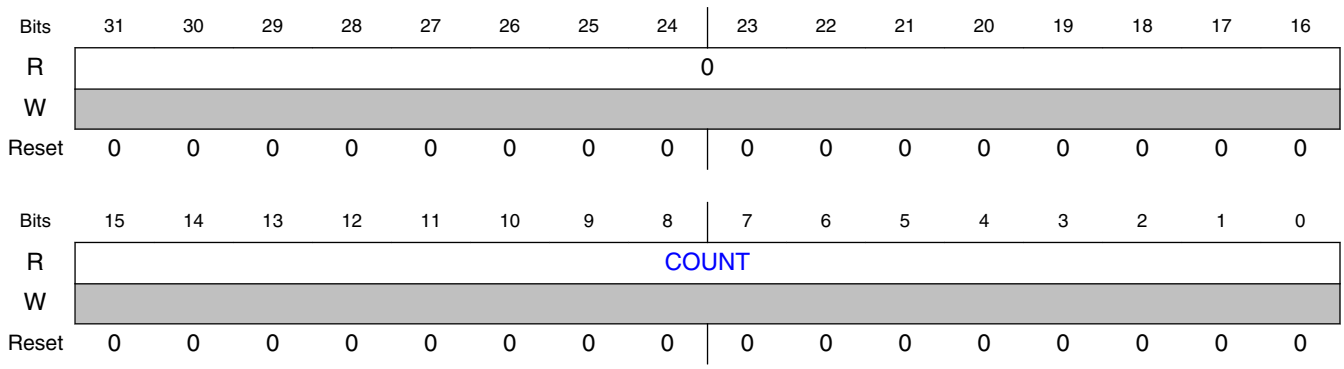
Field	Function
31-16 —	Reserved
15-0 COUNT	Number of 128- to 255-byte recieve packets

41.5.1.75 Rx 256- to 511-Byte Packets Statistic Register (RMON_R_P256TO511)

41.5.1.75.1 Offset

Register	Offset
RMON_R_P256TO511	2B4h

41.5.1.75.2 Diagram



41.5.1.75.3 Fields

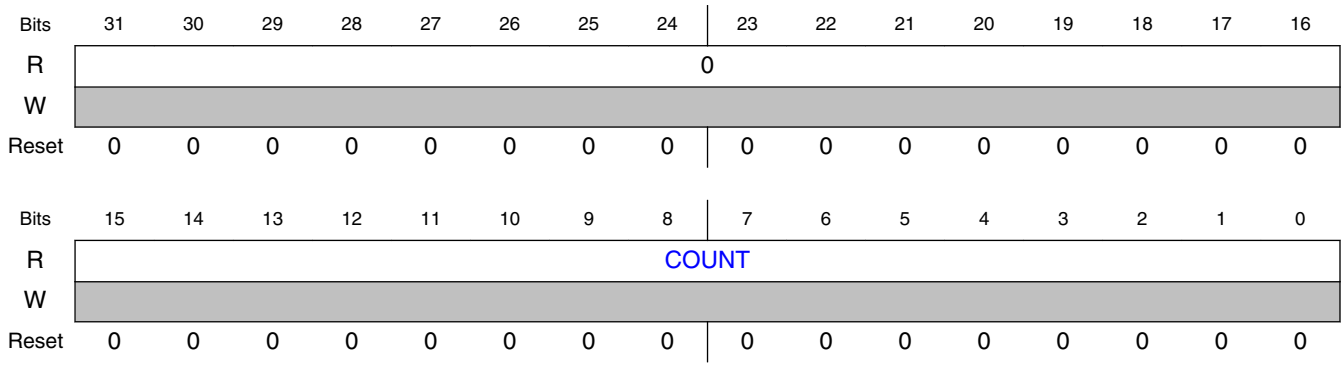
Field	Function
31-16 —	Reserved
15-0 COUNT	Number of 256- to 511-byte recieve packets

41.5.1.76 Rx 512- to 1023-Byte Packets Statistic Register (RMON_R_P512TO1023)

41.5.1.76.1 Offset

Register	Offset
RMON_R_P512TO1023	2B8h

41.5.1.76.2 Diagram



41.5.1.76.3 Fields

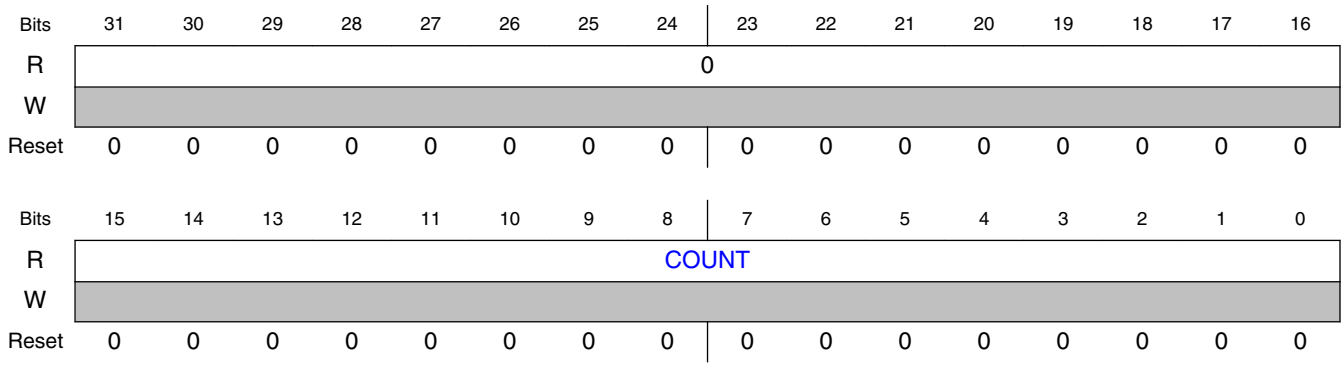
Field	Function
31-16 —	Reserved
15-0 COUNT	Number of 512- to 1023-byte recieve packets

41.5.1.77 Rx 1024- to 2047-Byte Packets Statistic Register (RMON_R_P1024TO2047)

41.5.1.77.1 Offset

Register	Offset
RMON_R_P1024TO2047	2BCh

41.5.1.77.2 Diagram



41.5.1.77.3 Fields

Field	Function
31-16 —	Reserved
15-0 COUNT	Number of 1024- to 2047-byte recieve packets

41.5.1.78 Rx Packets Greater than 2048 Bytes Statistic Register (RMON_R_P_GTE2048)

41.5.1.78.1 Offset

Register	Offset
RMON_R_P_GTE2048	2C0h

41.5.1.78.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	COUNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

41.5.1.78.3 Fields

Field	Function
31-16 —	Reserved
15-0 COUNT	Number of greater-than-2048-byte recieve packets

41.5.1.79 Rx Octets Statistic Register (RMON_R_OCTETS)

41.5.1.79.1 Offset

Register	Offset
RMON_R_OCTETS	2C4h

41.5.1.79.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	COUNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	COUNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

41.5.1.79.3 Fields

Field	Function
31-0 COUNT	Number of receive octets

41.5.1.80 Frames not Counted Correctly Statistic Register (IEEE_R_DROP)

41.5.1.80.1 Offset

Register	Offset
IEEE_R_DROP	2C8h

41.5.1.80.2 Function

Counter increments if a frame with invalid or missing SFD character is detected and has been dropped. None of the other counters increments if this counter increments.

41.5.1.80.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	COUNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

41.5.1.80.4 Fields

Field	Function
31-16	Reserved

Table continues on the next page...

Field	Function
—	
15-0 COUNT	Frame count

41.5.1.81 Frames Received OK Statistic Register (IEEE_R_FRAME_OK)

41.5.1.81.1 Offset

Register	Offset
IEEE_R_FRAME_OK	2CCh

41.5.1.81.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	COUNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

41.5.1.81.3 Fields

Field	Function
31-16 —	Reserved
15-0 COUNT	Number of frames received OK

41.5.1.82 Frames Received with CRC Error Statistic Register (IEEE_R_CRC)

41.5.1.82.1 Offset

Register	Offset
IEEE_R_CRC	2D0h

41.5.1.82.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	COUNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

41.5.1.82.3 Fields

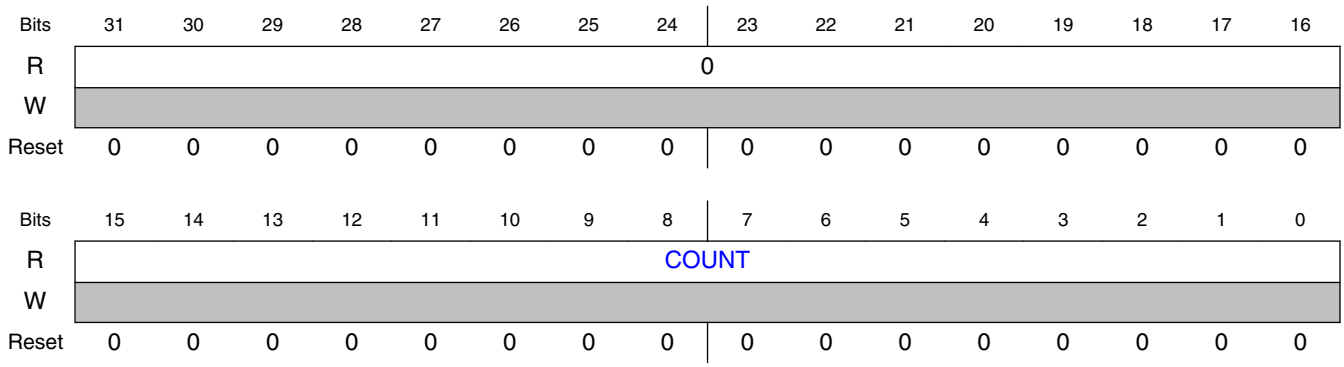
Field	Function
31-16 —	Reserved
15-0 COUNT	Number of frames received with CRC error

41.5.1.83 Frames Received with Alignment Error Statistic Register (IEEE_R_ALIGN)

41.5.1.83.1 Offset

Register	Offset
IEEE_R_ALIGN	2D4h

41.5.1.83.2 Diagram



41.5.1.83.3 Fields

Field	Function
31-16 —	Reserved
15-0 COUNT	Number of frames received with alignment error

41.5.1.84 Receive FIFO Overflow Count Statistic Register (IEEE_R_MACERR)

41.5.1.84.1 Offset

Register	Offset
IEEE_R_MACERR	2D8h

41.5.1.84.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	COUNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

41.5.1.84.3 Fields

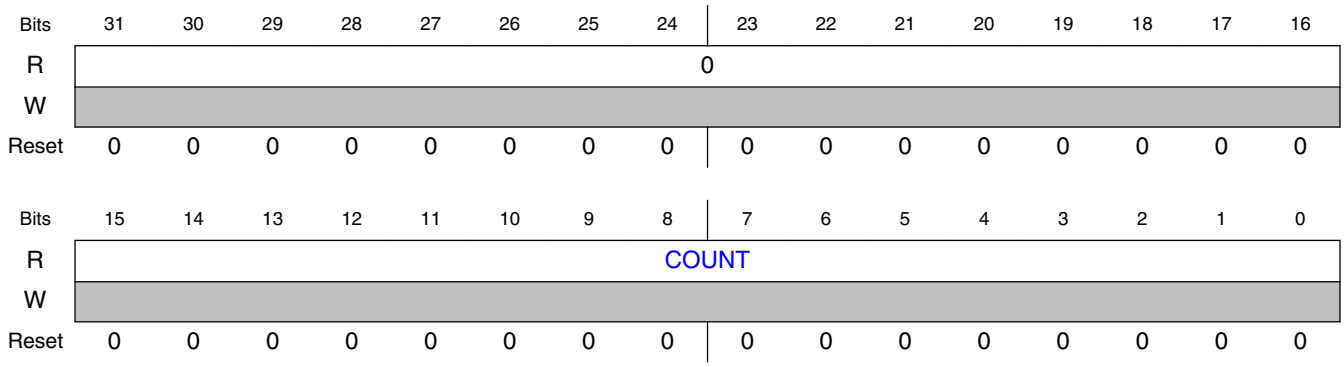
Field	Function
31-16 —	Reserved
15-0 COUNT	Receive FIFO overflow count

41.5.1.85 Flow Control Pause Frames Received Statistic Register (IEEE_R_FDXFC)

41.5.1.85.1 Offset

Register	Offset
IEEE_R_FDXFC	2DCh

41.5.1.85.2 Diagram



41.5.1.85.3 Fields

Field	Function
31-16 —	Reserved
15-0 COUNT	Number of flow-control pause frames received

41.5.1.86 Octet Count for Frames Received without Error Statistic Register (IEEE_R_OCTETS_OK)

41.5.1.86.1 Offset

Register	Offset
IEEE_R_OCTETS_OK	2E0h

41.5.1.86.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	COUNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	COUNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

41.5.1.86.3 Fields

Field	Function
31-0 COUNT	Number of octets for frames received without error **NOTE:** Counts total octets (includes header and FCS fields). Does not increment for the broadcast frames when broadcast reject is enabled and promiscuous mode is disabled within the receive control register (RCR).

41.5.1.87 Adjustable Timer Control Register (ATCR)

41.5.1.87.1 Offset

Register	Offset
ATCR	400h

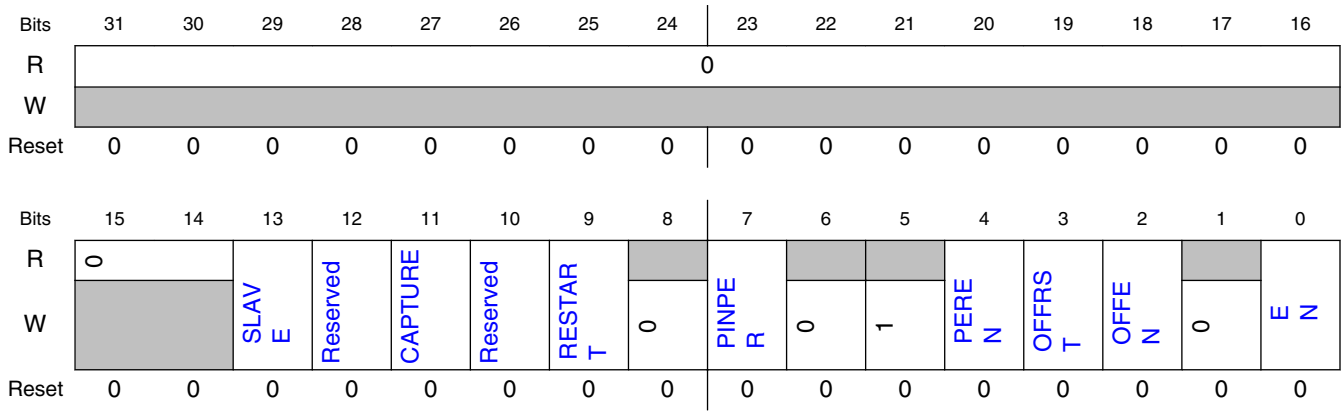
41.5.1.87.2 Function

ATCR command fields can trigger the corresponding events directly. It is not necessary to preserve any of the configuration fields when a command field is set in the register, that is, no read-modify-write is required.

NOTE

The CAPTURE and RESTART fields and bits 12 and 10 must be 0 in order to write to the other fields in this register.

41.5.1.87.3 Diagram



41.5.1.87.4 Fields

Field	Function
31-14 —	Reserved
13 SLAVE	Enable Timer Slave Mode 0b - The timer is active and all configuration fields in this register are relevant. 1b - The internal timer is disabled and the externally provided timer value is used. All other fields, except CAPTURE, in this register have no effect. CAPTURE can still be used to capture the current timer value.
12 —	Always write 0 to this field Always write 0 to this field.
11 CAPTURE	Capture Timer Value When this field is set, all other fields are ignored during a write. This field automatically clears to 0 after the command completes. 0b - No effect. 1b - The current time is captured and can be read from the ATVR register.
10 —	Always write 0 to this field Always write 0 to this field.
9 RESTART	Reset Timer Resets the timer to zero. This has no effect on the counter enable. If the counter is enabled when this field is set, the timer is reset to zero and starts counting from there. When set, all other fields are ignored during a write. This field automatically clears to 0 after the command completes. RESTART should be used when the timer is enabled.
8 —	Reserved
7 PINPER	Enables event signal output external pin frc_evt_period assertion on period event Enables event signal output assertion on period event. **NOTE:** Not all devices contain the event signal output. See the chip configuration details. 0b - Disable. 1b - Enable.

Table continues on the next page...

Field	Function
6 —	Reserved
5 —	This field must be written always with one **NOTE:** This field must be written always with one.
4 PEREN	Enable Periodical Event 0b - Disable. 1b - A period event interrupt can be generated (EIR[TS_TIMER]) and the event signal output is asserted when the timer wraps around according to the periodic setting ATPER. The timer period value must be set before setting this bit. Not all devices contain the event signal output. See the chip configuration details.
3 OFFRST	Reset Timer On Offset Event 0b - The timer is not affected and no action occurs, besides clearing OFFEN, when the offset is reached. 1b - If OFFEN is set, the timer resets to zero when the offset setting is reached. The offset event does not cause a timer interrupt.
2 OFFEN	Enable One-Shot Offset Event 0b - Disable. 1b - The timer can be reset to zero when the given offset time is reached (offset event). The field is cleared when the offset event is reached, so no further event occurs until the field is set again. The timer offset value must be set before setting this field.
1 —	Reserved
0 EN	Enable Timer 0b - The timer stops at the current value. 1b - The timer starts incrementing.

41.5.1.88 Timer Value Register (ATVR)

41.5.1.88.1 Offset

Register	Offset
ATVR	404h

41.5.1.88.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	ATIME															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ATIME															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

41.5.1.88.3 Fields

Field	Function
31-0 ATIME	A write sets the timer. A read returns the last captured value. To read the current value, issue a capture command (i.e., set ATCR[CAPTURE]) prior to reading this register.

41.5.1.89 Timer Offset Register (ATOFF)

41.5.1.89.1 Offset

Register	Offset
ATOFF	408h

41.5.1.89.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	OFFSET															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	OFFSET															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

41.5.1.89.3 Fields

Field	Function
31-0 OFFSET	Offset value for one-shot event generation. When the timer reaches the value, an event can be generated to reset the counter. If the increment value in ATINC is given in true nanoseconds, this value is also given in true nanoseconds.

41.5.1.90 Timer Period Register (ATPER)

41.5.1.90.1 Offset

Register	Offset
ATPER	40Ch

41.5.1.90.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PERIOD															
Reset	0	0	1	1	1	0	1	1	1	0	0	1	1	0	1	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PERIOD															
Reset	1	1	0	0	1	0	1	0	0	0	0	0	0	0	0	0

41.5.1.90.3 Fields

Field	Function
31-0 PERIOD	Value for generating periodic events Value for generating periodic events. Each instance the timer reaches this value, the period event occurs and the timer restarts. If the increment value in ATINC is given in true nanoseconds, this value is also given in true nanoseconds. The value should be initialized to 1,000,000,000 (1×10^9) to represent a timer wrap around of one second. The increment value set in ATINC should be set to the true nanoseconds of the period of clock ts_clk, hence implementing a true 1 second counter. **NOTE:** The value of PERIOD has the following constraint: 2^{32} – ENET_ATINC[INC_COR] – 3×ENET_ATINC[INC] ≥ PERIOD > 0.

41.5.1.91 Timer Correction Register (ATCOR)

41.5.1.91.1 Offset

Register	Offset
ATCOR	410h

41.5.1.91.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	COR														
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	COR															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

41.5.1.91.3 Fields

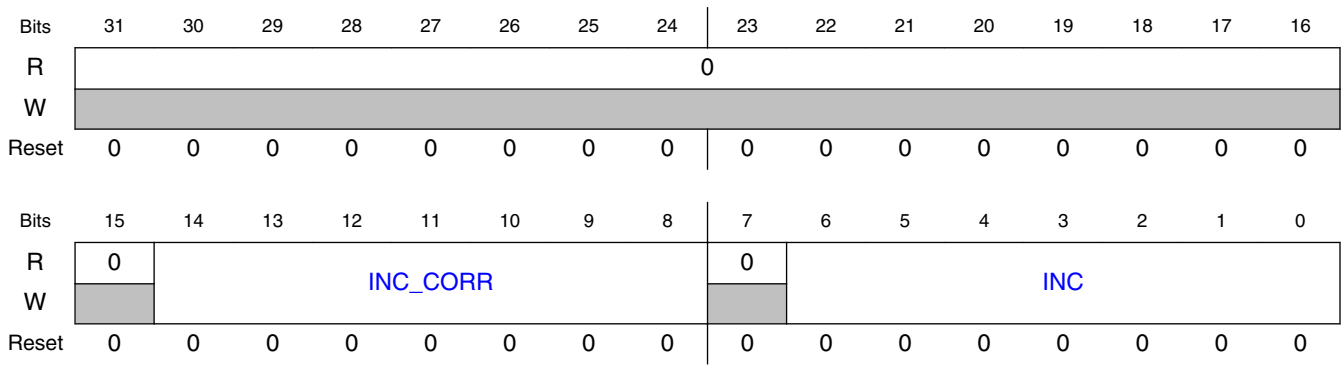
Field	Function
31 —	Reserved
30-0 COR	Correction Counter Wrap-Around Value Defines after how many timer clock cycles (ts_clk) the correction counter should be reset and trigger a correction increment on the timer. The amount of correction is defined in ATINC[INC_CORR]. A value of 0 disables the correction counter and no corrections occur. **NOTE:** This value is given in clock cycles, not in nanoseconds as all other values.

41.5.1.92 Time-Stamping Clock Period Register (ATINC)

41.5.1.92.1 Offset

Register	Offset
ATINC	414h

41.5.1.92.2 Diagram



41.5.1.92.3 Fields

Field	Function
31-15 —	Reserved
14-8 INC_CORR	Correction Increment Value This value is added every time the correction timer expires (every clock cycle given in ATCOR). A value less than INC slows down the timer. A value greater than INC speeds up the timer.
7 —	Reserved
6-0 INC	Clock Period Of The Timestamping Clock (ts_clk) In Nanoseconds The timer increments by this amount each clock cycle. For example, set to 10 for 100 MHz, 8 for 125 MHz, 5 for 200 MHz. **NOTE:** For highest precision, use a value that is an integer fraction of the period set in ATPER.

41.5.1.93 Timestamp of Last Transmitted Frame (ATSTMP)

41.5.1.93.1 Offset

Register	Offset
ATSTMP	418h

41.5.1.93.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TIMESTAMP															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TIMESTAMP															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

41.5.1.93.3 Fields

Field	Function
31-0 TIMESTAMP	Timestamp of the last frame transmitted by the core that had TxBD[TS] set the ff_tx_ts_frm signal asserted from the user application Timestamp of the last frame transmitted by the core that had TxBD[TS] set . This register is only valid when EIR[TS_AVAIL] is set.

41.5.1.94 Timer Global Status Register (TGSR)

41.5.1.94.1 Offset

Register	Offset
TGSR	604h

41.5.1.94.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0												TF3	TF2	TF1	TF0
W													W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

41.5.1.94.3 Fields

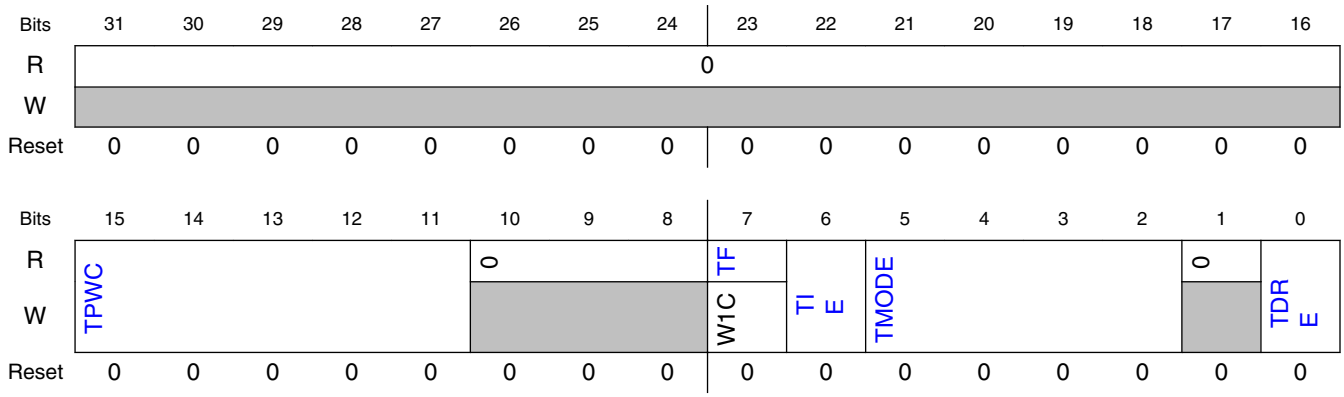
Field	Function
31-4 —	Reserved
3 TF3	Copy Of Timer Flag For Channel 3 0b - Timer Flag for Channel 3 is clear 1b - Timer Flag for Channel 3 is set
2 TF2	Copy Of Timer Flag For Channel 2 0b - Timer Flag for Channel 2 is clear 1b - Timer Flag for Channel 2 is set
1 TF1	Copy Of Timer Flag For Channel 1 0b - Timer Flag for Channel 1 is clear 1b - Timer Flag for Channel 1 is set
0 TF0	Copy Of Timer Flag For Channel 0 0b - Timer Flag for Channel 0 is clear 1b - Timer Flag for Channel 0 is set

41.5.1.95 Timer Control Status Register (TCSR0 - TCSR3)

41.5.1.95.1 Offset

Register	Offset
TCSR0	608h
TCSR1	610h
TCSR2	618h
TCSR3	620h

41.5.1.95.2 Diagram



41.5.1.95.3 Fields

Field	Function
31-16 —	Reserved
15-11 TPWC	Timer PulseWidth Control Specifies the pulse width associated with TMODE values of 1110 or 11X1. Updating this field takes a few cycles to register because it is synchronized to the 1588 clock. When changing this field: 1. Always disable the channel and read the TMODE field to verify that the channel is disabled. 2. Set TPWC to the desired value. 3. Reenable the channel. 00000b - Pulse width is one 1588-clock cycle. 00001b - Pulse width is two 1588-clock cycles. 00010b - Pulse width is three 1588-clock cycles. 00011b - Pulse width is four 1588-clock cycles. 11111b - Pulse width is 32 1588-clock cycles.
10-8 —	Reserved
7 TF	Timer Flag Sets when input capture or output compare occurs. This flag is double buffered between the module clock and 1588 clock domains. When this field is 1, it can be cleared to 0 by writing 1 to it. 0b - Input Capture or Output Compare has not occurred. 1b - Input Capture or Output Compare has occurred.
6 TIE	Timer Interrupt Enable 0b - Interrupt is disabled 1b - Interrupt is enabled
5-2 TMODE	Timer Mode Updating the Timer Mode field takes a few cycles to register because it is synchronized to the 1588 clock. The version of Timer Mode returned on a read is from the 1588 clock domain. When changing Timer Mode, always disable the channel and read this register to verify the channel is disabled first. 0000b - Timer Channel is disabled. 0001b - Timer Channel is configured for Input Capture on rising edge.

Table continues on the next page...

Field	Function
	0010b - Timer Channel is configured for Input Capture on falling edge. 0011b - Timer Channel is configured for Input Capture on both edges. 0100b - Timer Channel is configured for Output Compare - software only. 0101b - Timer Channel is configured for Output Compare - toggle output on compare. 0110b - Timer Channel is configured for Output Compare - clear output on compare. 0111b - Timer Channel is configured for Output Compare - set output on compare. 1000b - Reserved 1010b - Timer Channel is configured for Output Compare - clear output on compare, set output on overflow. 10x1b - Timer Channel is configured for Output Compare - set output on compare, clear output on overflow. 110xb - Reserved 1110b - Timer Channel is configured for Output Compare - pulse output low on compare for 1 to 32 1588-clock cycles as specified by TPWC. 1111b - Timer Channel is configured for Output Compare - pulse output high on compare for 1 to 32 1588-clock cycles as specified by TPWC.
1 —	Reserved
0 TDRE	Timer DMA Request Enable 0b - DMA request is disabled 1b - DMA request is enabled

41.5.1.96 Timer Compare Capture Register (TCCR0 - TCCR3)

41.5.1.96.1 Offset

Register	Offset
TCCR0	60Ch
TCCR1	614h
TCCR2	61Ch
TCCR3	624h

41.5.1.96.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	TCC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TCC															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

41.5.1.96.3 Fields

Field	Function
31-0 TCC	Timer Capture Compare This register is double buffered between the module clock and 1588 clock domains. When configured for compare, the 1588 clock domain updates with the value in the module clock domain whenever the Timer Channel is first enabled and on each subsequent compare. Write to this register with the first compare value before enabling the Timer Channel. When the Timer Channel is enabled, write the second compare value either immediately, or at least before the first compare occurs. After each compare, write the next compare value before the previous compare occurs and before clearing the Timer Flag. The compare occurs one 1588 clock cycle after the IEEE 1588 Counter increments past the compare value in the 1588 clock domain. If the compare value is less than the value of the 1588 Counter when the Timer Channel is first enabled, then the compare does not occur until following the next overflow of the 1588 Counter. If the compare value is greater than the IEEE 1588 Counter when the 1588 Counter overflows, or the compare value is less than the value of the IEEE 1588 Counter after the overflow, then the compare occurs one 1588 clock cycle following the overflow. When configured for capture, the value of the IEEE 1588 Counter is captured into the 1588 clock domain and then updated into the module clock domain, provided the Timer Flag is clear. Always read the capture value before clearing the Timer Flag.

Chapter 42
Universal Serial Bus Controller (USB)

42.1 Chip-specific USB information

Table 42-1. Reference links to related information

Topic	Related module(s)	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
Interrupts, DMA Events and XBAR Assignments	-	Interrupts, DMA Events and XBAR Assignments

42.2 Overview

The USB controller block provides high performance USB functionality that conforms to the *Universal Serial Bus Specification*, Rev. 2.0 (Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, NEC, Philips; 2000), and the *On-The-Go and Embedded Host Supplement to the USB Revision* 2.0 Specification (Hewlett-Packard Company, Intel Corporation, LSI Corporation, Microsoft Corporation, Renesas Electronics Corporation, ST-Ericsson; 2012).

The USB controller consists of two independent USB controller cores: two On-The-Go (OTG) controller cores. Each controller core supports UTMI interfaces according to its feature. See Features for more details. Both the controller cores are single-port cores. For the OTG cores, there is only one port. The port can be used as either a downstream or an upstream port.

The following figure is a block diagram of USB.

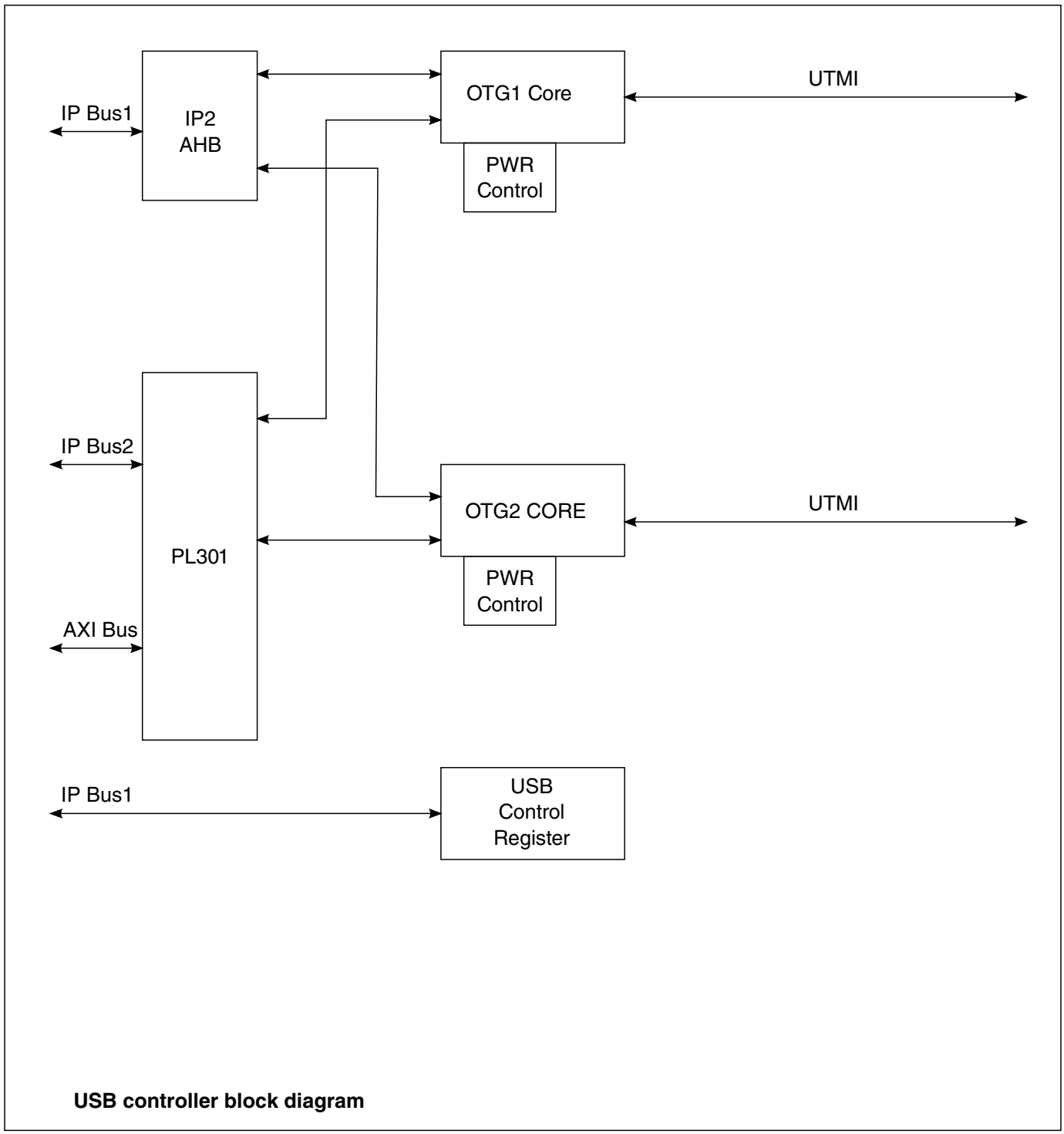


Figure 42-1. USB block diagram

42.2.1 Features

There are two USB 2.0 controller cores in this chip:

- Controller Core 0 is also named 'OTG1 Core'; its connected port is named 'OTG1 port'.
- Controller Core 1 is also named 'OTG2 Core'; its connected port is named 'OTG2 port'.

The following list provides features of each controller core.

- USB 2.0 Controller Core 0
  - High-Speed/Full-Speed/Low-Speed OTG core
  - HS/FS/LS UTMI compliant interface
  - High Speed, Full Speed and Low Speed operation in HOST mode (with UTMI transceiver)
  - High Speed, and Full Speed operation in Peripheral mode (with UTMI transceiver)
  - Hardware support for OTG signaling, session request protocol, and host negotiation protocol
  - Up to 8 bidirectional endpoints
  - Support charger detection
- USB 2.0 Controller Core 1
  - High-Speed/Full-Speed/Low-Speed OTG core
  - HS/FS/LS UTMI compliant interface
  - High Speed, Full Speed and Low Speed operation in Host mode (with UTMI transceiver)
  - High Speed, and Full Speed operation in Peripheral mode (with UTMI transceiver)
  - Hardware support for OTG signaling, session request protocol, and host negotiation protocol
  - Up to 8 bidirectional endpoints
- Low-power mode with local and remote wake-up capability
- Serial PHY interfaces configurable for bidirectional/unidirectional and differential/single ended
- Embedded DMA controller in each core

42.2.2 Modes of Operation

The USB has two main modes of operation: normal mode and low power mode.

Each USB OTG controller core can operate in High Speed operation (480 Mbps), Full Speed operation (12 Mbps) and Low Speed operation (1.5 Mbps).

This chapter explains the operation modes.

42.2.2.1 Normal Mode

The OTG controller core can operate in Host mode and Device (Peripheral) mode.

Each USB controller core has its corresponding port, which can work in one or more interface modes.

NOTE

Each controller supports only the interface type listed below. Selecting a different interface type in the PORTSC_PTS field results in unpredictable behavior and may cause the system to hang.

- OTG1 port
  - This port supports on-chip UTMI transceiver only.
- OTG2 port
  - This port supports on-chip UTMI transceiver only.

42.2.2.2 Low-Power Mode

Each USB controller core has a low-power mode (Suspend mode) to save power consumption.

As described in the USB 2.0 specification, the device can go into the Suspend state after it sees a constant Idle state on the upstream facing port. The OTG controller core enters Suspend mode after 3 ms of inactivity on the port when it is in Device Operation mode. Host controllers, including the OTG controller in Host mode, do not suspend automatically but can be placed in Suspend mode by software.

Either the local Arm platform or the remote USB Host/Peripheral can initiate a wake-up sequence to resume USB communication. For details about Suspend/Resume, see USB Power Control.

42.3 External Signals

The table found here describes the external signals of USB.

Table 42-2. USB External Signals

Signal	Description	Pad	Mode	Direction
USB_OTG1_PWR	To control PMIC to supply VBUS voltage	GPIO_AD_B0_02 pin 1	ALT3	O
		GPIO_AD_B1_01 pin 18	ALT0	
USB_OTG1_OC	External Input for VBUS over current detection	GPIO_AD_B0_03 pin 0	ALT3	I
		GPIO_AD_B1_03 pin 15	ALT0	
USB_OTG1_ID	OTG1 ID Signal	GPIO_AD_B0_01	ALT3	I
		GPIO_AD_B1_02 pin 14	ALT0	
USB_OTG2_PWR	To control PMIC to supply VBUS voltage	GPIO_EMC_41	ALT3	O
		GPIO_AD_B0_15	ALT0	
USB_OTG2_OC	External Input for VBUS over current detection	GPIO_EMC_40	ALT3	I
		GPIO_AD_B0_14	ALT0	
USB_OTG2_ID	OTG2 ID Signal	GPIO_AD_B0_00	ALT3	I
		GPIO_AD_B1_00 pin 19	ALT0	

42.4 Functional Description

These sections describe the functionality of the various building blocks of the USB.

42.4.1 USB 2.0 Controller Core 0

The USB 2.0 Controller 0 is an instantiation of an EHCI-compatible core which supports high-, full-, and low-speed operation.

In Host mode, this controller core supports high-, full-, and low-speed operation. In Device mode, it supports high- and full-speed operation.

42.4.1.1 Host Mode

The controller supports direct connection of a HS/FS/LS device with on-chip UTMI transceiver.

Although there is no separate Transaction Translator block in the system, the transaction translator function normally associated with a USB 2.0 high speed hub has been implemented within the DMA and protocol engine blocks to support connection to full and low speed devices.

42.4.1.2 Peripheral (Device) Mode

- Up to eight bidirectional endpoints
- High/full-speed operation
- Support of HNP, and SRP
- Remote wake-up capability

42.4.2 USB 2.0 Controller Core 1

USB 2.0 Controller Core 1 is an instantiation of EHCI-compatible core which supports High Speed / Full Speed / Low Speed operation

42.4.3 USB Power Control

The USB controller supports suspend and wake-up functionality.

The power control block allows for placing the transceiver in USB low power mode when USB bus is IDLE, and supports local and remote wake-up to bring the transceiver out of USB low power mode when needed. Additionally, the power control block can wake-up the Arm platform from core sleep mode by generating an interrupt.

42.4.3.1 Entering Low Power Suspend Mode

In Host operation mode, low power suspend mode is entered as follows:

1. Clear the ASE and PSE bits in USB_USBCMD, and wait until the AS and PS bits in USB_USBSTS become "0".
2. Set the "SUSPEND" bit in USB_PORTSC1
3. Set the "PHCD" bit in USB_PORTSC1
4. Set all PWD bits in USBPHYx_PWD
5. Set CLKGATE in USBPHYx_CTRL

NOTE

Step 3,4,5 shall be done in atomic operation. That is, interrupt should be disabled during these three steps.

For device operation mode, low power suspend mode is entered as follows:

1. After Host drive is IDLE for 3ms, an SLI interrupt is issued (the "DCSUSPEND" or "SLI" bit in USB_USBSTS)
2. Set the "PHCD" bit on USB_PORTSC1
3. Set all PWD bits in USBPHYx_PWD
4. Set CLKGATE in USBPHYx_CTRL

NOTE

Step 2,3,4 shall be done in atomic operation. That is, interrupt should be disabled during these three steps.

42.4.3.2 Wake-Up Events

The power control block monitors the USB bus when the USB core is in the USB suspend state.

Depending on whether the core is on Host or Device mode, a number of wake-up conditions are monitored. Upon detection of a wake-up condition, an interrupt will be generated to Arm platform if the related wake-up interrupt enable bit is set.

USB wake-up interrupt also re-activates the Arm platform clocks if they were stopped during the suspend.

42.4.3.2.1 Host Mode Events

The host controller wakes up on the following events:

- Remote Wake-up Request

A peripheral can request the host to reactivate the bus by driving wake-up signaling on the DM/DP lines. The power control block sends a wake-up request to the USB core when a J-K transition on DM/DP line is detected.

- Wake-Up On Overcurrent

If Wake-Up On Overcurrent is enabled (WKOC bit in the USB core register PORTSC1 is set '1'), the power control block sends a wake-up request to the USB core when an overcurrent event is detected.

- Wake-Up On Disconnect

If Wake-Up On Disconnect is enabled (WKDC bit in the USB core register PORTSC1 is set '1'), the power control block sends a wake-up request to the USB core when a disconnection event is detected (J-SE0/K-SE0 transition on DM/DP line).

- Wake-Up On Connect

If a Wake-Up On Connect is enabled (WKCN bit in the USB core register PORTSC1 is set '1'), the power control sub-block sends a wake-up request to the USB core when the connection event is detected (SE0-J/SE0-K transition on DM/DP line).

For a detailed description of register bits WKOC, WKDC, WKCN, please see Port Status & Control (USB_*n*PORTSC1).

42.4.4 Interrupts

42.4.4.1 USB Core Interrupts

Each USB core uses one dedicated vector in the Interrupt Table. The vector numbers associated with each of the cores can be found in the Interrupt section.

With the exception of the wake-up interrupts, all of the interrupt sources are controlled in the USB Cores. Refer to the Interrupt Enable Register (USB_*n*USBINTR) for details.

42.4.4.2 USB Wake-Up Interrupts

Each USB Core has an associated wake-up interrupt. The wake-up interrupts are generated outside of the USB controller cores, but using the same vector as the corresponding USB controller cores interrupt.

These interrupts are generated by the Power Control blocks which run on the 32 KHz standby clock. The wake-up interrupt is designed to work even when the USB and Arm platform clocks are disabled, such that a wake-up condition on the USB bus can re-activate the Arm platform clocks.

Because the wake-up interrupt is generated and cleared on a 32 KHz clock, this interrupt request responds very slowly to clear actions. For this reason, the software must disable the wake-up interrupt to clear the request flag. Disabling the interrupt masks the request instantaneously as this is clocked by the Arm platform clock. The software should wait for at least three 32 KHz clock cycles before re-enabling this interrupt to allow sufficient

time for the request flag to clear. Because this interrupt is only used during low power modes of the USB, it is sufficient to enable the wake-up interrupt just prior to entering the USB suspend mode.

42.5 USB Operation Model

This section describes the detailed application knowledge for OTG1 and OTG2 ports.

42.5.1 Register Interface

Configuration, control and status registers are divided into three categories, identification, capability and operational registers.

NOTE

USB controller registers support only DWORD (32-bit) access.

- Identification registers are used to declare the slave interface presence along with the complete set of the hardware configuration parameters.
- Static, read only capability registers define the software limits, restrictions, and capabilities of the host/device controller.
- Operational registers are dynamic control or status registers that may be read only, read/write, or read/write-to-clear. The following sections define the use of these registers.

EHCI registers are listed alongside device registers to show the complementary nature of host and device control.

The following table describes the Interface register sets.

Table 42-3. Interface Register Sets

Offset	Register Set	Explanation
000h-07Ch	Identification Registers	Identification registers are used to declare the slave interface presence and include a table of the hardware configuration parameters.
100h-124h	Capability Registers	Capability registers specify the limits, restrictions, and capabilities of a host/device controller implementation. These values are used as parameters to the host/device controller driver.
080h-0FCh 140h-1FCh	Operational Registers	Operational registers are used by the system software to control and monitor the operational state of the host/device controller.

42.5.1.1 Configuration, Control and Status Register Set

The following table describes the Device/Host capability registers.

NOTE

Depending on implementation, "*x*" can have the following values: UOG1, UOG2.

Table 42-4. Device/Host Capability Registers

Offset	Size (Bytes)	Mnemonic	Register Name	Device Mode	Host Mode
000h	4	USB_*x*_ID	Identification Register	O	O
004h	4	USB_*x*_HWGENERAL	General Hardware Parameters	O	O
008h	4	USB_*x*_HWHOST	Host Hardware Parameters	X	O
00Ch	4	USB_*x*_HWDEVICE	Device Hardware Parameters	O	X
010h	4	USB_*x*_HWTXBUF	TX Buffer Hardware Parameters	O	O
014h	4	USB_*x*_HWRXBUF	RX Buffer Hardware Parameters	O	O
018-07Fh		-	Reserved		
080h	4	USB_*x*_GPTIMER0LD	General Purpose Timer #0 Load Register	O	O
084h	4	USB_*x*_GPTIMER0CTRL	General Purpose Timer #0 Control Register	O	O
088h	4	USB_*x*_GPTIMER1LD	General Purpose Timer #1 Load Register	O	O
08Ch	4	USB_*x*_GPTIMER1CTRL	General Purpose Timer #1 Control Register	O	O
090h	4	USB_*x*_SBUSCFG	System Bus Interface Configuration Register	O	O
094-09Fh		-	Reserved		
100h	1	USB_*x*_CAPLENGTH	Capability Register Length	O	O
101h		-	Reserved		
102h	2	USB_*x*_HCIVERSION	Host Controller Interface Version Number	X	O
104h	4	USB_*x*_HCSPARAMS	Host Controller Structural Parameters	X	O
108h	4	USB_*x*_HCCPARAMS	Host Controller Capability Parameters	X	O
10C-11Fh		-	Reserved		
120h	2	USB_*x*_DCIVERSION	Device Controller Interface Version Number	O	X
122h	2	-	Reserved		
124h	4	USB_*x*_DCCPARAMS	Device Controller Capability Parameters	O	X
128-13Fh		-	Reserved		
140h	4	USB_*x*_USBCMD	USB Command Register	O	O
144h	4	USB_*x*_USBSTS	USB Status Register	O	O
148h	4	USB_*x*_USBINTR	USB Interrupt Enable Register	O	O
14Ch	4	USB_*x*_FRINDEX	USB Frame Index	O	O
150h	4	-	Reserved		
154h	4	USB_*x*_PERIODICLISTBASE	Frame List Base Address	X	O

Table continues on the next page...

Table 42-4. Device/Host Capability Registers (continued)

Offset	Size (Bytes)	Mnemonic	Register Name	Device Mode	Host Mode
		USB_*x*_DEVICEADDR	USB Device Address	O	X
158h	4	USB_*x*_ASYNCLISTADDR	Next Asynchronous List Address	X	O
	4	USB_*x*_ENDPOINTLISTADDR	Address at Endpoint list in memory	O	X
15Ch	4	-	Reserved		
160h	4	USB_*x*_BURSTSIZE	Programmable Burst Size	O	O
164h	4	USB_*x*_TXFILLTUNING	Host Transmit Pre-Buffer Packet Tuning	X	O
168h	4	-	Reserved		
170h	4	-	Reserved		
178h	4	USB_*x*_ENDPTNAK	Endpoint NAK register	O	X
17Ch	4	USB_*x*_ENDPTNAKEN	Endpoint NAK Enable register	O	X
180h	4	USB_*x*_CONFIGFLAG	Configured Flag Register	X	O
184h	4	USB_*x*_PORTSC1	Port Status/Control Register 1	O	O
188-1A3h		-	Reserved		
1A4h	4	USB_*x*_OTGSC	On-The-Go Status/Control Register (OTG only)	O	O
1A8h	4	USB_*x*_USBMODE	USB Controller Operating Mode	O	O
1ACh	4	USB_*x*_ENDPTSETUPSTAT	Endpoint Setup Status	O	X
1B0h	4	USB_*x*_ENDPTPRIME	Endpoint Initialization	O	X
1B4h	4	USB_*x*_ENDPTFLUSH	Endpoint De-Initialization	O	X
1B8h	4	USB_*x*_ENDPTSTATUS	Endpoint Status	O	X
1BCh	4	USB_*x*_ENDPTCOMPLETE	Endpoint Complete	O	X
1C0 1C4 ... 1DCh	64	USB_*x*_ENDPTCTRL0 USB_*x*_ENDPTCTRL1 .... USB_*x*_ENDPTCTRL7	Endpoint Control Register 0-7	O	X

NOTE

"O" means the register is available in host/device operation mode;

"X" means the register is reserved in host/device operation mode

42.5.1.2 Identification Registers

Identification registers are used to declare the slave interface presence and include a table of the hardware configuration parameters.

42.5.1.3 OTG Operations

42.5.2 Host Data Structures

This section defines the interface data structures used to communicate control, status, and data between HCD (software) and the Enhanced Host Controller (hardware).

The data structure definitions in this chapter support a 32-bit memory buffer address space. The interface consists of a Periodic Schedule, Periodic Frame List, Asynchronous Schedule, Isochronous Transaction Descriptors, Split-transaction Isochronous Transfer Descriptors, Queue Heads, and Queue Element Transfer Descriptors.

The periodic frame list is the root of all periodic (isochronous and interrupt transfer type) transfers for the host controller. The asynchronous list is the root for all the bulk and control transfers. Isochronous data streams are managed using Isochronous Transaction Descriptors. Isochronous split-transaction data streams are managed with Split-transaction Isochronous Transfer Descriptors. All Interrupt, Control, and Bulk data streams are managed via queue heads and Queue Element Transfer Descriptors. These data structures are optimized to reduce the total memory footprint of the schedule and to reduce (on average) the number of memory accesses needed to execute a USB transaction.

Note that software must ensure that no interface data structure reachable by the EHCI host controller spans a 4 K-page boundary.

The data structures defined in this section are (from the host controller's perspective) a mix of read-only and read/writeable fields. The host controller must preserve the read-only fields on all data structure writes.

42.5.2.1 Periodic Frame List

This schedule is for all periodic transfers (isochronous and interrupt). The periodic schedule is referenced from the operational registers space using the USB_PERIODICLISTBASE address register and the USB_FRINDEX register.

The periodic schedule is based on an array of pointers called the Periodic Frame List.

The USB_PERIODICLISTBASE address register is combined with the USB_FRINDEX register to produce a memory pointer into the frame list. The Periodic Frame List implements a sliding window of work over time.

The following figure shows the organization of periodic schedule.

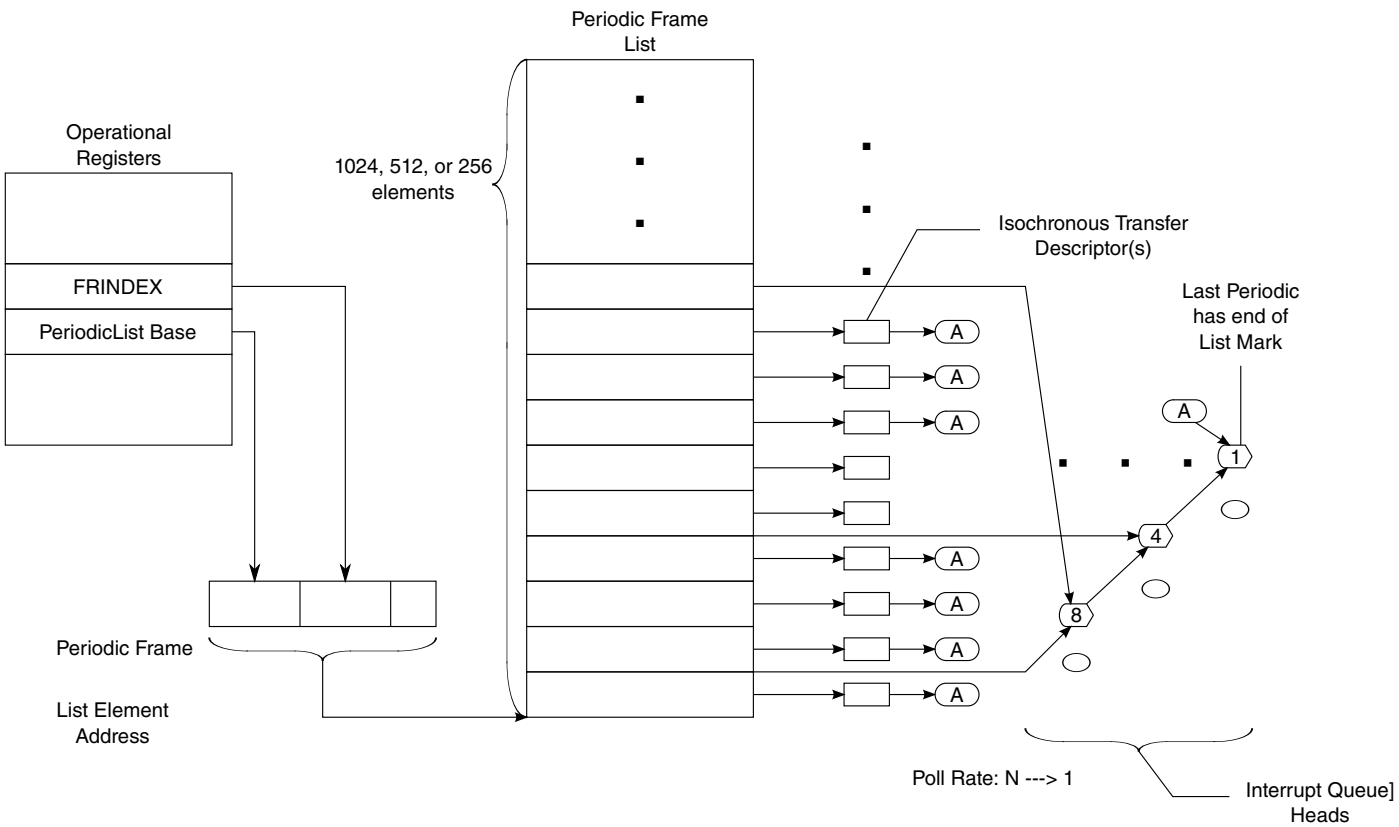


Figure 42-2. Periodic Schedule Organization

Split transaction Interrupt, Bulk and Control are also managed using queue heads and queue element transfer descriptors.

The periodic frame list is a 4 K-page aligned array of Frame List Link pointers. The length of the frame list may be programmable. The programmability of the periodic frame list is exported to system software via the USB_HCCPARAMS register. If non-programmable, the length is 1024 elements. If programmable, the length can be selected by system software as one of 256, 512, or 1024 elements. An implementation must support all three sizes. Programming the size (that is, the number of elements) is accomplished by system software writing the appropriate value into Frame List Size field in the USB_USBCMD register.

Frame List Link pointers direct the host controller to the first work item in the frame's periodic schedule for the current micro-frame. The link pointers are aligned on DWord boundaries within the Frame List.

The table below illustrates the format of the Frame list element pointer.

Table 42-5. Format of Frame List Element Pointer

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Frame List Link Pointer																											0		Typ			03-00H

Frame List Link pointers always reference memory objects that are 32-byte aligned. The referenced object may be an isochronous transfer descriptor for high-speed devices, a split-transaction isochronous transfer descriptor (for full-speed isochronous endpoints), or a queue head (used to support high-, full- and low-speed interrupt). System software should not place non-periodic schedule items into the periodic schedule. The least significant bits in a frame list pointer are used to key the host controller as to the type of object the pointer is referencing.

The least significant bit is the T-Bit (bit 0). When this bit is set to a one, the host controller never uses the value of the frame list pointer as a physical memory pointer. The Typ field is used to indicate the exact type of data structure being referenced by this pointer. The value encodings are.

Table 42-6. Typ Field Value Definitions

Value	Meaning
00b	Isochronous Transfer Descriptor
01b	Queue Head
10b	Split Transaction Isochronous Transfer Descriptor.
11b	Frame Span Traversal Node.

42.5.2.2 Asynchronous List Queue Head Pointer

The Asynchronous Transfer List (based at the USB_ASYNCLISTADDR register) is where all of the control and bulk transfers are managed.

Host controllers use this list only when it reaches the end of the periodic list, the periodic list is disabled, or the periodic list is empty.

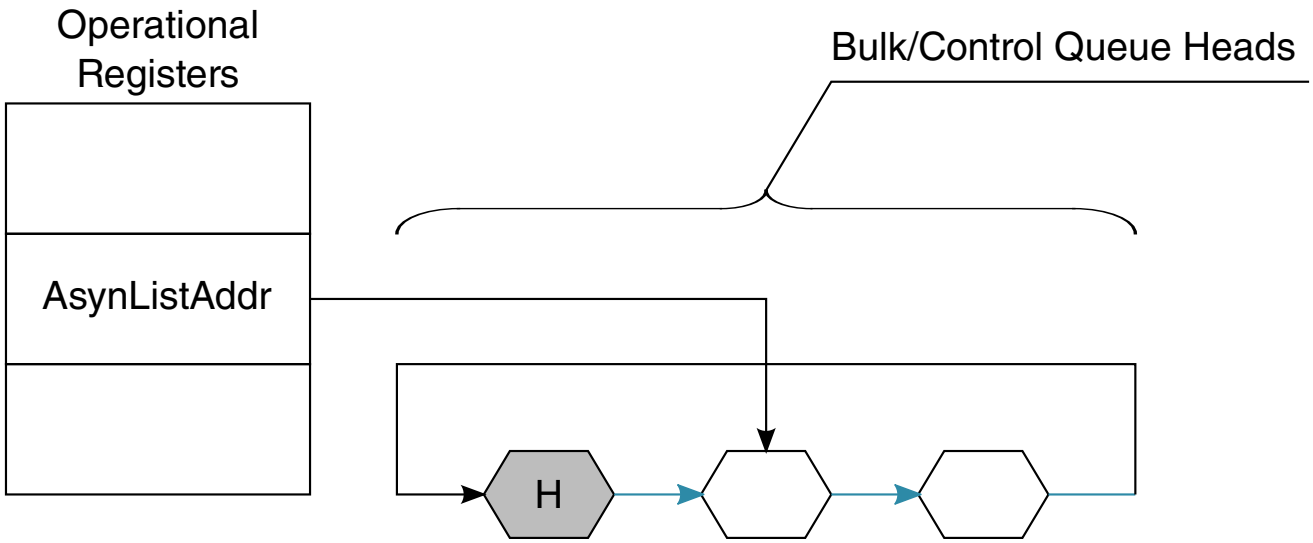


Figure 42-3. Asynchronous Schedule Organization

The Asynchronous list is a simple circular list of queue heads. The USB_ASYNCLISTADDR register is simply a pointer to the next queue head. This implements a pure round-robin service for all queue heads linked into the asynchronous list.

42.5.2.3 Isochronous (High-Speed) Transfer Descriptor (iTD)

The format of an isochronous transfer descriptor is shown in the table below.

This structure is used only for high-speed isochronous endpoints. All other transfer types should use queue structures. Isochronous TDs must be aligned on a 32-byte boundary.

Table 42-7. Isochronous Transaction Descriptor (iTD)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Next Link Pointer																											0		Typ		T	03-00 H
Status				Transaction 0 Length												IOC	PG*			Transaction 0 Offset*												07-04 H
Status				Transaction 1 Length												IOC	PG*			Transaction 1 Offset*												0B-08H
Status				Transaction 2 Length												IOC	PG*			Transaction 2 Offset*												0F-0CH
Status				Transaction 3 Length												IOC	PG*			Transaction 3 Offset*												13-10 H

Table continues on the next page...

Table 42-7. Isochronous Transaction Descriptor (iTD) (continued)

Status	Transaction 4 Length			IOC	PG*	Transaction 4 Offset*					17-14H
Status	Transaction 5 Length			IOC	PG*	Transaction 5 Offset*					1B-18H
Status	Transaction 6 Length			IOC	PG*	Transaction 6 Offset*					1F-1CH
Status	Transaction 7 Length			IOC	PG*	Transaction 7 Offset*					23-20H
Buffer Pointer (Page 0)						EndPt	R	Device Address			27-24H
Buffer Pointer (Page 1)						I/O	Maximum Packet Size				2B-28H
Buffer Pointer (Page 2)						-				Mult	2F-2CH
Buffer Pointer (Page 3)						-					33-30H
Buffer Pointer (Page 4)						-					37-34H
Buffer Pointer (Page 5)						-					3B-38H
Buffer Pointer (Page 6)						-					3F-3CH

Host Controller Read/Write

Host Controller Read Only

These fields may be modified by the host controller if the I/O field indicates an OUT.

42.5.2.3.1 Next Link Pointer

The first DWord of an iTD is a pointer to the next schedule data structure.

The following table describes the Next Schedule Element pointer field.

Table 42-8. Next Schedule Element Pointer

Bit	Description
31-5 Link Pointer (LP)	These bits correspond to memory address signals [31:5], respectively. This field points to another Isochronous Transaction Descriptor (iTD/siTD) or Queue Head (QH).
4-3	These bits are reserved and their value has no effect on operation. Software should initialize this field to zero.

Table continues on the next page...

Table 42-8. Next Schedule Element Pointer (continued)

Reserved	
2-1 QH/(s)iTD Select (Typ)	This field indicates to the Host Controller whether the item referenced is an iTD, siTD or a QH. This allows the Host Controller to perform the proper type of processing on the item after it is fetched. Value encodings are: Value Meaning 00b iTD (isochronous transfer descriptor) 01b QH (queue head) 10b siTD (split transaction isochronous transfer descriptor 11b FSTN (frame span traversal node)
0 Terminate (T)	1= Link Pointer field is not valid. 0= Link Pointer field is valid.

42.5.2.3.2 iTD Transaction Status and Control List

DWords 1 through 8 are eight slots of transaction control and status.

Each transaction description includes:

- Status results field
- Transaction length (bytes to send for OUT transactions and bytes received for IN transactions).
- Buffer offset. The PG and Transaction X Offset fields are used with the buffer pointer list to construct the starting buffer address for the transaction.

The host controller uses the information in each transaction description plus the endpoint information contained in the first three DWords of the Buffer Page Pointer list, to execute a transaction on the USB.

The following table describes iTD Transaction Status and Control fields.

Table 42-9. iTD Transaction Status and Control

Bit	Description	
31-28 Status	This field records the status of the transaction executed by the host controller for this slot. This field is a bit vector with the following encoding:	
	Bit	Definition
	31	Active. Set to one by software to enable the execution of an isochronous transaction by the Host Controller. When the transaction associated with this descriptor is completed, the Host Controller sets this bit to zero indicating that a transaction for this element should not be executed when it is next encountered in the schedule.
	30	Data Buffer Error. Set to a one by the Host Controller during status update to indicate that the Host Controller is unable to keep up with the reception of incoming data (overrun) or is unable to supply data fast enough during transmission (under run). If an overrun condition occurs, no action is necessary.
	29	Babble Detected. Set to one by the Host Controller during status update when" babble" is detected during the transaction generated by this descriptor.

Table continues on the next page...

Table 42-9. iTD Transaction Status and Control (continued)

Bit	Description	
	28	Transaction Error (XactErr). Set to one by the Host Controller during status update in the case where the host did not receive a valid response from the device (Timeout, CRC, Bad PID, etc.). This bit may only be set for isochronous IN transactions.
27-16 Transaction X Length	For an OUT, this field is the number of data bytes the host controller sends during the transaction. The host controller is not required to update this field to reflect the actual number of bytes transferred during the transfer. For an IN, the initial value of the endpoint to deliver. During the status update, the host controller writes back the field is the number of bytes the host expects the number of bytes successfully received. The value in this register is the actual byte count (0‡zero length data, 1‡one byte, 2‡two bytes, etc.). The maximum value this field may contain is 0xC00 (3072).	
15 Interrupt On Complete (IOC)	If this bit is set to one, it specifies that when this transaction completes, the Host Controller should issue an interrupt at the next interrupt threshold.	
14-12 Page Select (PG)	These bits are set by software to indicate which of the buffer page pointers the offset field in this slot should be concatenated to produce the starting memory address for this transaction. The valid range of values for this field is 0 to 6.	
11-0 Transaction X Offset	This field is a value that is an offset, expressed in bytes, from the beginning of a buffer. This field is concatenated onto the buffer page pointer indicated in the adjacent *PG* field to produce the starting buffer address for this transaction.	

42.5.2.3.3 iTD Buffer Page Pointer List (Plus)

DWords 9-15 of an isochronous transaction descriptor are nominally page pointers (4 K aligned) to the data buffer for this transfer descriptor. This data structure requires the associated data buffer to be contiguous (relative to virtual memory), but allows the physical memory pages to be non-contiguous.

Seven page pointers are provided to support the expression of eight isochronous transfers. The seven pointers allow for 3 (transactions) * 1024 (maximum packet size) * 8 (transaction records) (24576 bytes) to be moved with this data structure, regardless of the alignment offset of the first page.

Because each pointer is a 4 K aligned page pointer, the least significant 12 bits in several of the page pointers are used for other purposes.

The tables below illustrate the field descriptions.

Table 42-10. iTD Buffer Pointer Page 0 (Plus)

Bit	Description
31-12 Buffer Pointer (Page 0)	This is a 4 K aligned pointer to physical memory. Corresponds to memory address bits [31:12].
11-8 Endpoint Number (Endpt)	This 4-bit field selects the particular endpoint number on the device serving as the data source or sink.

Table continues on the next page...

Table 42-10. iTD Buffer Pointer Page 0 (Plus) (continued)

Bit	Description
7 Reserved	Bit reserved for future use and should be initialized by software to zero.
6-0 Device Address	This field selects the specific device serving as the data source or sink.

Table 42-11. iTD Buffer Pointer Page 1 (Plus)

Bit	Description
31-12 Buffer Pointer (Page 1)	This is a 4K aligned pointer to physical memory. Corresponds to memory address bits [31:12].
11 Direction (I/O)	0 = OUT; 1 = IN. This field encodes whether the high-speed transaction should use an IN or OUT PID.
10-0 Maximum Packet Size	This directly corresponds to the maximum packet size of the associated endpoint *(wMaxPacketSize)*. This field is used for high-bandwidth endpoints where more than one transaction is issued per transaction description (per micro-frame). This field is used with the *Multi* field to support high-bandwidth pipes. This field is also used for all IN transfers to detect packet babble. Software should not set a value larger than 1024 (400h). Any value larger yields undefined results.

Table 42-12. iTD Buffer Pointer Page 2 (Plus)

Bit	Description
31-12 Buffer Pointer	This is a 4K aligned pointer to physical memory. Corresponds to memory address bits [31:12].
11-2 Reserved	This bit reserved for future use and should be set to zero.
1-0 Multi	This field is used to indicate to the host controller the number of transactions that should be executed per transaction description (per micro-frame). The valid values are: Value Meaning 00b Reserved. A zero in this field yields undefined results. 01b One transaction to be issued for this endpoint per micro- frame. 10b Two transactions to be issued for this endpoint per micro- frame. 11b Three transactions to be issued for this endpoint per micro- frame.

Table 42-13. iTD Buffer Pointer Page 3-6

Bit	Description
31-12 Buffer Pointer	This is a 4 K aligned pointer to physical memory. Corresponds to memory address bits [31:12].
11-0 Reserved	These bits reserved for future use and should be set to zero.

42.5.2.4 Split Transaction Isochronous Transfer Descriptor (siTD)

All Full-speed isochronous transfers through the internal transaction translator are managed using the siTD data structure. This data structure satisfies the operational requirements for managing the split transaction protocol.

The following table shows the Split Transaction Isochronous Transfer Descriptor (siTD).

Table 42-14. Split Transaction Isochronous Transfer Descriptor

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Addr
Next Link Pointer																											0		Typ		T	03-00
I/O	Port Number							-	Hub Addr							Reserved				EndPt				-	Device Address							07-04[1]
Reserved																µFrame C-mask								µFrame S-mask								0B-08[1]
ioc	P	Reserved				Total Bytes to Transfer										µFrame C-prog-mask								Status								0F-0C[2]
Buffer Pointer (Page 0)																				Current Offset												13-10[2]
Buffer Pointer (Page 1)																				Reserved							TP		T-count			17-14[2]
Back Pointer																											0				T	1B-18

1. 04-0B: Static Endpoint State
2. 0C-13: Transfer results

Host Controller Read/Write

Host Controller Read Only

42.5.2.4.1 Next Link Pointer

DWord0 of a siTD is a pointer to the next schedule data structure.

The following table describes the Next Link Pointer fields.

Table 42-15. Next Link Pointer

Bit	Description
31-5	Next Link Pointer (LP). This field contains the address of the next data object to be processed in the periodic list and corresponds to memory address signals [31:5], respectively.

Table continues on the next page...

Table 42-15. Next Link Pointer (continued)

Bit	Description
4-3	Reserved. These bits must be written as zeros.
2-1	QH/(s)iTD Select (Typ). This field indicates to the Host Controller whether the item referenced is an iTD/siTD or a QH. This allows the Host Controller to perform the proper type of processing on the item after it is fetched. Value encodings are: Value Meaning 00b iTD (isochronous transfer descriptor) 01b QH (queue head) 10b siTD (split transaction isochronous transfer descriptor 11b FSTN (frame span traversal node)
0	Terminate (T). 1 = Link Pointer field is not valid. 0 = Link Pointer is valid.

42.5.2.4.2 siTD Endpoint Capabilities/Characteristics

DWords 1 and 2 specify static information about the full-speed endpoint, the addressing of the parent Companion Controller, and micro-frame scheduling control.

The tables below describe the Endpoint and transaction translator characteristics and micro-frame schedule control fields.

Table 42-16. Endpoint and Transaction Translator Characteristics

Bit	Description
31	Direction (I/O).0 = OUT; 1 = IN. This field encodes whether the full-speed transaction should be an IN or OUT.
30-24	Port Number. This field is the port number of the recipient Transaction Translator.
23	Reserved. Bit reserved and should be set to zero.
22-16	Hub Address. This field holds the device address of the Companion Controllers' hub.
15-12	Reserved. Field reserved and should be set to zero.
11-8	Endpoint Number (Endpt). This 4-bit field selects the particular endpoint number on the device serving as the data source or sink.
7	Reserved. Bit is reserved for future use. It should be set to zero.
6-0	Device Address. This field selects the specific device serving as the data source or sink.

Table 42-17. Micro-frame Schedule Control

Bit	Description
31-16	Reserved. This field reserved for future use. It should be set to zero.
15-8	Split Completion Mask (mFrame C-Mask). This field (along with the *Active* and *SplitX- state* fields in the *Status* byte) is used to determine during which micro-frames the host controller should execute complete-split transactions. When the criteria for using this field is met, an all zeros value has undefined behavior. The host

Table continues on the next page...

Table 42-17. Micro-frame Schedule Control (continued)

Bit	Description
	controller uses the value of the three low-order bits of the FRINDEX register to index into this bit field. If the FRINDEX register value indexes to a position where the m*Frame C-Mask* field is a one, then this siTD is a candidate for transaction execution. There may be more than one bit in this mask set.
7-0	Split Start Mask (mFrame S-mask). This field (along with the *Active* and *SplitX-state* fields in the *Status* byte) is used to determine during which micro-frames the host controller should execute start-split transactions. The host controller uses the value of the three low-order bits of the FRINDEX register to index into this bit field. If the FRINDEX register value indexes to a position where the m*Frame S-mask* field is a one, then this siTD is a candidate for transaction execution. An all zeros value in this field, in combination with existing periodic frame list has undefined results.

42.5.2.4.3 siTD Transfer State

DWords 3-6 are used to manage the state of the transfer.

The following table describes siTD transfer state fields.

Table 42-18. siTD Transfer Status and Control

Bit	Description
31	Interrupt On Complete (ioc). 0 = Do not interrupt when transaction is complete. 1 = Do interrupt when transaction is complete. When the host controller determines that the split transaction has completed it asserts a hardware interrupt at the next interrupt threshold.
30	Page Select (P). Used to indicate which data page pointer should be concatenated with the *CurrentOffset* field to construct a data buffer pointer (0 selects *Page 0* pointer and 1 selects *Page 1*). The host controller is not required to write this field back when the siTD is retired (*Active* bit transitioned from a one to a zero).
29-26	Reserved. This field reserved for future use and should be set to zero.
25-16	Total Bytes To Transfer. This field is initialized by software to the total number of bytes expected in this transfer. Maximum value is 1023 (3FFh)
15-8	µFrame Complete-split Progress Mask (C-prog-Mask). This field is used by the host controller to record which split-completes has been executed.
7-0: Status—This field records the status of the transaction executed by the host controller for this slot. It is a bit vector with the encoding shown in the following rows.	
7	Active. Set to one by software to enable the execution of an isochronous split transaction by the Host Controller.
6	ERR. Set to a one by the Host Controller when an ERR response is received from the Companion Controller.
5	Data Buffer Error. Set to a one by the Host Controller during status update to indicate that the Host Controller is unable to keep up with the reception of incoming data (overrun) or is unable to supply data fast enough during transmission (under run). In the case of an under run, the Host Controller transmits an incorrect CRC (thus invalidating the data at the endpoint). If an overrun condition occurs, no action is necessary.
4	Babble Detected. Set to a one by the Host Controller during status update when" babble" is detected during the transaction generated by this descriptor.
3	Transaction Error (XactErr). Set to a one by the Host Controller during status update in the case where the host did not receive a valid response from the device (Timeout, CRC, Bad PID, etc.). This bit is set only for IN transactions.
2	Missed Micro-Frame. The host controller detected that a host-induced hold- off caused the host controller to miss a required complete-split transaction.

Table continues on the next page...

Table 42-18. siTD Transfer Status and Control (continued)

Bit	Description
1	Split Transaction State (SplitXstate). The bit encodings are: Value Meaning 00b Do Start Split. This value directs the host controller to issue a Start split transaction to the endpoint when a match is encountered in the S-mask. 01b Do Complete Split. This value directs the host controller to issue a Complete split transaction to the endpoint when a match is encountered in the C-mask.
0	Reserved. Bit reserved for future use and should be set to zero.

42.5.2.4.4 siTD Buffer Pointer List (plus)

DWords 4 and 5 are the data buffer page pointers for the transfer. This structure supports one physical page cross. The most significant 20 bits of each DWord in this section are the 4 K (page) aligned buffer pointers.

The least significant 12 bits of each DWord are used as additional transfer state. The following table describes the siTD buffer pointer fields.

Table 42-19. Buffer Page Pointer List (plus)

Bit	Description
31-12	Buffer Pointer List. Bits [31:12] of DWords 4 and 5 are 4 K page aligned physical memory addresses. These bits correspond to physical address bits [31:12] respectively. The lower 12 bits in each pointer are defined and used as specified below. The field *P* (see siTD Transfer State) specifies the *current* active pointer.
Bits 11-0 (Page 0)	Current Offset—The 12 least significant bits of the Page 0 pointer are the current byte offset for the current page pointer (as selected with the page indicator bit (*P* field). The host controller is not required to write this field back when the siTD is retired (*Active* bit transitioned from a one to a zero).
Bits 11-0 (Page 1)—The least significant bits of the Page 1 pointer are split into three subfields as shown in the following rows.	
11-5 (Page 1)	Reserved
4-3 (Page 1)	Transaction position (TP). This field is used with T-count to determine whether to send *all*, *first*, *middle*, or *last* with each outbound transaction payload. System software must initialize this field with the appropriate starting value. The host controller must correctly manage this state during the lifetime of the transfer. The bit encodings are: Value Meaning 00b All. The entire full-speed transaction data payload is in this transaction (that is, less than or equal to 188 bytes). 01b Begin. This is the first data payload for a full- speed that is greater than 188 bytes. 10B Mid. This is the *middle* payload for a full-speed OUT transaction that is larger than 188 bytes.

Table continues on the next page...

Table 42-19. Buffer Page Pointer List (plus) (continued)

Bit	Description
	11b End. This is the *last* payload for a full-speed OUT transaction that was larger than 188 bytes.
2-0 (Page 1)	Transaction count (T-Count). Software initializes this field with the number of OUT start-splits this transfer requires. Any value larger than 6 is undefined.

42.5.2.4.5 siTD Back Link Pointer

DWord 6 of a siTD is simply another schedule link pointer. This pointer is always zero, or references a siTD, and it cannot reference any other schedule data structure.

The following table describes the siTD back link pointer fields.

Table 42-20. siTD Back Link Pointer

Bit	Description
31-5	siTD Back Pointer. This field is a physical memory pointer to a siTD.
4-1	Reserved. This field is reserved for future use. It should be set to zero.
0	Terminate (T). 1 = siTD Back Pointer field is not valid. 0 = siTD Back Pointer field is valid.

42.5.2.5 Queue element transfer descriptor (qTD)

This data structure is only used with a queue head. It describes one or more USB transactions to transfer up to 20480 (5*4096) bytes.

The structure contains two structure pointers used for queue advancement, a DWord of transfer state, and a five-element array of data buffer pointers.

It is 32 bytes and must be physically contiguous.

The buffer associated with this transfer must be virtually contiguous. The buffer may start on any byte boundary; however, for optimal utilization of on-chip busses it is recommended to align the buffers on a 32-byte boundary. A separate buffer pointer list element must be used for each physical page in the buffer, regardless of whether the buffer is physically contiguous.

Host controller updates (host controller writes) to stand-alone qTDs only occur during transfer retirement. References in the following bit field definitions of updates to the qTD are to the qTD portion of a queue head.

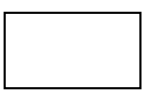
The following table shows the queue element transfer descriptor data structure.

Table 42-21. Queue element transfer descriptor data structure

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Addr
Next qTD Pointer																											0				T	03-00
Alternate Next qTD Pointer																											0				T	07-04
dt	Total Bytes to Transfer															ioc	C_Page			Cerr		PID Code		Status								0B-08[1]
Buffer Pointer (page 0)																				Current Offset												0F-0C[1]
Buffer Pointer (page 1)																				Reserved												13-10
Buffer Pointer (page 2)																				Reserved												17-14
Buffer Pointer (page 3)																				Reserved												1B-18
Buffer Pointer (page 4)																				Reserved												1F-1C

1. 08-0F: Transfer Results

Host Controller Read/Write



Host Controller Read Only

Queue Element Transfer Descriptors must be aligned on 32-byte boundaries.

42.5.2.5.1 Next qTD Pointer

The first DWord of an element transfer descriptor is a pointer to another transfer element descriptor.

The following table describes Next qTD pointer fields.

Table 42-22. qTD Next Element Transfer Pointer (DWord 0)

Bit	Description
31-5	Next Transfer Element Pointer. This field contains the physical memory address of the next qTD to be processed. The field corresponds to memory address signals[31:5], respectively.
4-1	Reserved

Table continues on the next page...

Table 42-22. qTD Next Element Transfer Pointer (DWord 0) (continued)

0	Terminate (T). 1= pointer is invalid. 0=Pointer is valid (points to a valid Transfer Element Descriptor). This bit indicates to the Host Controller that there are no more valid entries in the queue.

42.5.2.5.2 Alternate Next qTD Pointer

The second DWord of a queue element transfer descriptor is used to support hardware-only advance of the data stream to the next transfer descriptor on short packet. To be more explicit the host controller always uses this pointer when the current qTD is retired due to short packet.

The following table describes the TD Alternate Next Element Transfer Pointer field descriptions.

Table 42-23. TD Alternate Next Element Transfer Pointer (DWord 1)

Bit	Description
31-5	Alternate Next Transfer Element Pointer. This field contains the physical memory address of the next qTD to be processed in the event that the current qTD execution encounters a short packet (for an IN transaction). The field corresponds to memory address signals [31:5], respectively.
4-1	Reserved
0	Terminate (T). 1= pointer is invalid. 0=Pointer is valid (points to a valid Transfer Element Descriptor). This bit indicates to the Host Controller that there are no more valid entries in the queue.

42.5.2.5.3 qTD Token

The third DWord of a queue element transfer descriptor contains most of the information the host controller requires to execute a USB transaction (the remaining endpoint-addressing information is specified in the queue head).

NOTE

The field descriptions forward reference fields defined in the queue head. Where necessary, these forward references are preceded with a QH notation.

The following table describes the TD Token fields.

Table 42-24. TD Token (DWord 2)

Bit	Description
31 Data Toggle	This is the data toggle sequence bit. The use of this bit depends on the setting of the *Data Toggle Control* bit in the queue head.

Table continues on the next page...

Table 42-24. TD Token (DWord 2) (continued)

<table>
<tr><th>Bit</th><th>Description</th></tr>
<tr><td>30-16
Total Bytes to Transfer</td><td>This field specifies the total number of bytes to be moved with this transfer descriptor. This field is decremented by the number of bytes actually moved during the transaction, only on the successful completion of the transaction. The maximum value software may store in this field is 5 * 4K (5000H). This is the maximum number of bytes 5 page pointers can access. If the value of this field is zero when the host controller fetches this transfer descriptor (and the active bit is set), the host controller executes a zero-length transaction and retires the transfer descriptor. It is not a requirement for OUT transfers that Total Bytes To Transfer be an even multiple of QHD.Maximum Packet Length. If software builds such a transfer descriptor for an OUT transfer, the last transaction is always less than QHD.Maximum Packet Length.

Although it is possible to create a transfer up to 20K this assumes the 1st offset into the first page is 0. When the offset cannot be predetermined, crossing past the 5th page can be guaranteed by limiting the total bytes to 16K**. Therefore, the maximum recommended transfer is 16 K(4000H).</td></tr>
<tr><td>15
Interrupt On Complete (IOC)</td><td>If this bit is set to a one, it specifies that when this qTD is completed, the Host Controller should issue an interrupt at the next interrupt threshold.</td></tr>
<tr><td>14-12
Current Page (C_Page)</td><td>This field is used as an index into the qTD buffer pointer list. Valid values are in the range 0H to 4H. The host controller is not required to write this field back when the qTD is retired.</td></tr>
<tr><td>11-10
Error Counter (CERR)</td><td>This field is a 2-bit down counter that keeps track of the number of consecutive Errors detected while executing this qTD. If this field is programmed with a non-zero value during set-up, the Host Controller decrements the count and writes it back to the qTD if the transaction fails. If the counter counts from one to zero, the Host Controller marks the qTD inactive, sets the Halted bit to a one, and error status bit for the error that caused CERR to decrement to zero. An interrupt is generated if the USB Error Interrupt Enable bit in the USBINTR register is set to a one. If HCD programs this field to zero during set-up, the Host Controller does not count errors for this qTD and there is no limit on the retries of this qTD. Note that write-backs of intermediate execution state are to the queue head overlay area, not the qTD.

Transaction Error - Decrement

Data Buffer Error - No Decrement[3]

Stalled - No Decrement[1]

Babble Detected - No Decrement[1]

No Error - No Decrement[2]
<table>
<tr><th>Error</th><th>Decrement Counter</th></tr>
<tr><td>1</td><td>Detection of Babble or Stall automatically halts the queue head. Thus, count is not decremented</td></tr>
<tr><td>2</td><td>If the EPS field indicates a HS device or the queue head is in the Asynchronous Schedule (and PIDCode indicates an IN or OUT) and a bus transaction completes and the host controller does not detect a transaction error, then the host controller should reset CERR to extend the total number of errors for this transaction. For example, CERR should be reset with maximum value (3) on each successful completion of a transaction. The host controller must never reset this field if the value at the start of the transaction is 00b.

See Split Transaction Interrupt for CERR adjustment rules when the EPS field indicates a FS or LS device and the queue head is in the Periodic Schedule. See Asynchronous - Do Complete Split for CERR adjustment rules when the EPS field indicates a FS or LS device, the queue head is in the Asynchronous schedule and the PIDCode indicates a SETUP.</td></tr>
</table></td></tr>
</table>

Table continues on the next page...

Table 42-24. TD Token (DWord 2) (continued)

Bit	Description	
	3	Data buffer errors are host problems. They don't count against the device's retries.
	NOTE: Software must not program CERR to a value of zero when the EPS field is programmed with a value indicating a Full- or Low-speed device. This combination could result in undefined behavior.	
9-8 PID Code	This field is an encoding of the token, which should be used for transactions associated with this transfer descriptor. Encodings are:	
	00b	OUT Token generates token (E1H)
	01b	IN Token generates token (69H)
	10b	SETUP Token generates token (2DH) (undefined if endpoint is an interrupt, the queue head is non-zero) transfer type, for example, µ*Frame S-mask* field in.
	11b	Reserved
7-0 Status	This field is used by the Host Controller to communicate individual command execution states back to HCD. This field contains the status of the last transaction performed on this qTD. The bit encodings are:	
	Bit	Status Field Description
	7	Active. Set to one by software to enable the execution of transactions by the Host Controller.
	6	Halted. Set to one by the Host Controller during status updates to indicate that a serious error has occurred at the device/endpoint addressed by this qTD. This can be caused by babble, the error counter counting down to zero, or reception of the STALL handshake from the device during a transaction. Any time that a transaction results in the Halted bit being set to a one, the Active bit is also set to zero.
	5	Data Buffer Error. Set to a one by the Host Controller during status update to indicate that the Host Controller is unable to keep up with the reception of incoming data (overrun) or is unable to supply data fast enough during transmission (under run). If an overrun condition occurs, the Host Controller forces a timeout condition on the USB, invalidating the transaction at the source. If the host controller sets this bit to a one, then it remains a one for the duration of the transfer.
	4	Babble Detected. Set to a one by the Host Controller during status update when" babble" is detected during the transaction. In addition to setting this bit, the Host Controller also sets the *Halted* bit to a one. Because "babble" is considered a fatal error for the transfer, setting the Halted bit to a one insures that no more transactions occur because of this descriptor.
	3	Transaction Error (XactErr). Set to a one by the Host Controller during status update in the case where the host did not receive a valid response from the device (Timeout, CRC, Bad PID, etc.). If the host controller sets this bit to a one, then it remains a one for the duration of the transfer.
	2	Missed Micro-Frame. This bit is ignored unless the *QH.EPS* field indicates a full- or low-speed endpoint and the queue head is in the periodic list. This bit is set when the host controller detected that a host-induced hold-off caused the host controller to miss a required complete-split transaction. If the host controller sets this bit to a one, then it remains a one for the duration of the transfer.
	1	Split Transaction State (SplitXstate). This bit is ignored by the host controller unless the *QH.EPS* field indicates a full- or low-speed endpoint. When a Full- or Low-speed device, the host controller uses this bit to track the state of the split-

Table continues on the next page...

Table 42-24. TD Token (DWord 2) (continued)

Bit	Description	
		transaction. The functional requirements of the host controller for managing this state bit and the split transaction protocol depends on whether the endpoint is in the periodic or asynchronous schedule. The bit encodings are: Value Meaning 0b Do Start Split. This value directs the host controller to issue a Start split transaction to the endpoint. 1b Do Complete Split. This value directs the host controller to issue a Complete split transaction to the endpoint.
	0	Ping State (P)/ERR. If the *QH.EPS* field indicates a High-speed device and the *PID_Code* indicates an OUT endpoint, then this is the state bit for the Ping protocol. The bit encodings are: Value Meaning 0b Do OUT. This value directs the host controller to issue an OUT PID to the endpoint. 1b Do Ping. This value directs the host controller to issue a PING PID to the endpoint. If the *QH.EPS* field does not indicate a High-speed device, then this field is used as an error indicator bit. It is set to a one by the host controller whenever a periodic split-transaction receives an ERR handshake.

42.5.2.5.4 qTD Buffer Page Pointer List

The last five DWords of a queue element transfer descriptor is an array of physical memory address pointers. These pointers reference the individual pages of a data buffer.

System software initializes Current Offset field to the starting offset into the current page, where current page is selected through the value in the C_Page field.

The following table describes the qTD Buffer Pointer(s) (DWords 3-7) fields.

Table 42-25. qTD Buffer Pointer(s) (DWords 3-7)

Bit	Description
31-12	Buffer Pointer List. Each element in the list is a 4 K page aligned physical memory address. The lower 12 bits in each pointer are reserved (except for the first one), as each memory pointer must reference the start of a 4 K page. The field C_Page specifies the current active pointer. When the transfer element descriptor is fetched, the starting buffer address is selected using C_Page (similar to an array index to select an array element). If a transaction spans a 4K buffer boundary, the host controller must detect the page-span boundary in the data stream, increment C_Page and advance to the next buffer pointer in the list, and conclude the transaction through the new buffer pointer.
11-0	Current Offset (Reserved). This field is reserved in all pointers except the first one (for example Page 0). The host controller should ignore all reserved bits. For the page 0 current offset interpretation, this field is the byte offset into the current page (as selected by C_Page). The host controller is not required to write this field back when the qTD is retired. Software should ensure the Reserved fields are initialized to zero.

42.5.2.6 Queue Head

The table located in this section shows the Queue Head structure layout.

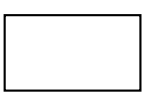
The following table shows the queue head structure layout.

Table 42-26. Queue Head Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Addr
Queue Head Horizontal Link Pointer																											0		Typ		T	03-00
RL				C	Maximum Packet Length											H	dtc	EP		EndPt				I	Device Address							07-04[1]
Mult		Port Number[2]							Hub Addr[2]							μFrame C-mask[2]								μFrame S-mask								0B-08[1]
Current qTD Pointer																											0					0F-0C
Next qTD Pointer																											0				T	13-10[3]
Alternate Next qTD pointer																											NakCnt				T	17-14[4]
dt	Total Bytes to Transfer															ioc	C_Page			Cerr		PID Code		Status								1B-18
Buffer Pointer (Page 0)																				Current Offset												1F-1C
Buffer Pointer (Page 1)																				Reserved				C-prog-mask[2]								23-20
Buffer Pointer (Page 2)																				S-bytes[2]									FrameTag[2]			27-24[4]
Buffer Pointer (Page 3)																				Reserved												2B-28
Buffer Pointer (Page 4)																				Reserved												2F-2C[3]

1. 04-0B: Static endpoint state.
2. These fields are used exclusively to support split transactions to USB 2.0 hubs
3. 10-2F: Transfer overlay.
4. 14-27: Transfer results.

Host Controller Read/Write



Host Controller Read Only

42.5.2.6.1 Queue Head Horizontal Link Pointer

The first DWord of a Queue Head contains a link pointer to the next data object to be processed after any required processing in this queue has been completed, as well as the control bits defined below.

This pointer may reference a queue head or one of the isochronous transfer descriptors. It must not reference a queue element transfer descriptor.

The following table describes the Queue head DWord 0 fields.

Table 42-27. Queue Head DWord 0

Bit	Description
31-5	Queue Head Horizontal Link Pointer (QHLP). This field contains the address of the next data object to be processed in the horizontal list and corresponds to memory address signals [31:5], respectively.
4-3	Reserved
2-1	QH/(s)iTD Select (Typ). This field indicates to the hardware whether the item referenced by the link pointer is an iTD, siTD or a QH. This allows the Host Controller to perform the proper type of processing on the item after it is fetched. Value encodings are: Value Meaning 00b iTD (isochronous transfer descriptor) 01b QH (queue head) 10b siTD (split transaction isochronous transfer descriptor) 11b FSTN (frame span traversal node)
0	Terminate (T). 1=Last QH (pointer is invalid). 0=Pointer is valid. If the queue head is in the context of the periodic list, a one bit in this field indicates to the host controller that this is the end of the periodic list. This bit is ignored by the host controller when the queue head is in the Asynchronous schedule. Software must ensure that queue heads reachable by the host controller always have valid horizontal link pointers.

42.5.2.6.2 Queue Head Endpoint Capabilities/Characteristics

The second and third DWords of a Queue Head specifies static information about the endpoint. This information does not change over the lifetime of the endpoint.

There are three types of information in this region:

- Endpoint Characteristics. These are the USB endpoint characteristics including addressing, maximum packet size, and endpoint speed.
- Endpoint Capabilities. These are adjustable parameters of the endpoint. They effect how the endpoint data stream is managed by the host controller.
- Split Transaction Characteristics. This data structure is used to manage full- and low-speed data streams for bulk, control, and interrupt via split transactions to USB2.0 Hub Transaction Translator. There are additional fields used for addressing the hub and scheduling the protocol transactions (for periodic).

The host controller must not modify the bits in this region.

The following table describes the Endpoint characteristics: Queue head DWord 1 fields.

Table 42-28. Endpoint Characteristics: Queue Head DWord 1

Bit	Description	
31-28	Nak Count Reload (RL). This field contains a value, which is used by the host controller to reload the Nak Counter field.	
27	Control Endpoint Flag (C). If the *QH.EPS* field indicates the endpoint is not a high-speed device, and the endpoint is a control endpoint, then software must set this bit to a one. Otherwise, it should always set this bit to zero.	
26-16	Maximum Packet Length. This directly corresponds to the maximum packet size of the associated endpoint (*wMaxPacketSize*). The maximum value this field may contain is 0x400 (1024).	
15	Head of Reclamation List Flag (H). This bit is set by System Software to mark a queue head as being the head of the reclamation list.	
14	Data Toggle Control (DTC). This bit specifies where the host controller should get the initial data toggle on an overlay transition. 0b Ignore DT bit from incoming qTD. Host controller preserves DT bit in the queue head. 1b Initial data toggle comes from incoming qTD DT bit. Host controller replaces DT bit in the queue head from the DT bit in the qTD.	
13-12	Endpoint Speed (EPS). This is the speed of the associated endpoint. Bit combinations are:	
	Value	Meaning
	00b	Full-Speed (12 Mbits/sec)
	01b	Low-Speed (1.5 Mbits/sec)
	10b	High-Speed (480 Mbits/sec)
	11b	Reserved
	This field must not be modified by the host controller.	
11-8	Endpoint Number (Endpt). This 4-bit field selects the particular endpoint number on the device serving as the data source or sink.	
7	Inactivate on Next Transaction (I). This bit is used by system software to request that the host controller set the Active bit to zero. See Rebalancing the periodic schedule , for full operational details. This field is only valid when the queue head is in the Periodic Schedule and the *EPS* field indicates a Full or Low-speed endpoint. Setting this bit to one when the queue head is in the Asynchronous Schedule or the *EPS* field indicates a high-speed device yields undefined results.	
6-0	Device Address. This field selects the specific device serving as the data source or sink.	

The table below describes the Endpoint capabilities: Queue head DWord 2 field descriptions.

Table 42-29. Endpoint Capabilities: Queue Head DWord 2

Bit	Description
31-30	High-Bandwidth Pipe Multiplier (Mult). This field is a multiplier used to key the host controller as the number of successive packets the host controller may submit to the endpoint in the current execution. The host controller makes the simplifying assumption that software properly initializes this field (regardless of location of queue head in the schedules or other run time parameters). The valid values are: Value Meaning

Table continues on the next page...

Table 42-29. Endpoint Capabilities: Queue Head DWord 2 (continued)

	00b Reserved. A zero in this field yields undefined results. 01b One transaction to be issued for this endpoint per micro-frame. 10b Two transactions to be issued for this endpoint per micro-frame. 11b Three transactions to be issued for this endpoint per micro-frame.
29-23	Port Number. This field is ignored by the host controller unless the *EPS* field indicates a full- or low-speed device. The value is the port number identifier on the USB 2.0 Hub (for hub at device address *Hub Addr* below), below which the full- or low-speed device associated with this endpoint is attached. This information is used in the split-transaction protocol.
22-16	Hub Addr. This field is ignored by the host controller unless the *EPS* field indicates a full-or low-speed device. The value is the USB device address of the USB 2.0 Hub below which the full- or low-speed device associated with this endpoint is attached. This field is used in the split-transaction protocol.
15-8	Split Completion Mask (µFrame C-Mask). This field is ignored by the host controller unless the *EPS* field indicates this device is a low- or full-speed device and this queue head is in the periodic list. This field (along with the *Active* and *SplitX-state* fields) is used to determine during which micro-frames the host controller should execute a complete-split transaction. When the criteria for using this field are met, a zero value in this field has undefined behavior. This field is used by the host controller to match against the three low-order bits of the FRINDEX register. If the FRINDEX register bits decode to a position where the µ*Frame C- Mask* field is a one, then this queue head is a candidate for transaction execution. There may be more than one bit in this mask set.
7-0	Interrupt Schedule Mask (µFrame S-mask). This field is used for all endpoint speeds. Software should set this field to a zero when the queue head is on the asynchronous schedule. A non-zero value in this field indicates an interrupt endpoint. The host controller uses the value of the three low-order bits of the FRINDEX register as an index into a bit position in this bit vector. If the µ*Frame S-mask* field has a one at the indexed bit position then this queue head is a candidate for transaction execution. If the *EPS* field indicates the endpoint is a high-speed endpoint, then the transaction executed is determined by the *PID_Code* field contained in the execution area. This field is also used to support split transaction types: Interrupt (IN/OUT). This condition is true when this field is non-zero and the *EPS* field indicates this is either a full- or low-speed device. A zero value in this field, in combination with existing in the periodic frame list has undefined results.

42.5.2.6.3 Transfer Overlay-Queue Head

The nine DWords in this area represent a transaction working space for the host controller. The general operational model is that the host controller can detect whether the overlay area contains a description of an active transfer. If it does not contain an active transfer, then it follows the Queue Head Horizontal Link Pointer to the next queue head. The host controller will never follow the Next Transfer Queue Element or Alternate Queue Element pointers unless it is actively attempting to advance the queue. For the duration of the transfer, the host controller keeps the incremental status of the transfer in the overlay area. When the transfer is complete, the results are written back to the original queue element.

The DWord3 of a Queue Head contains a pointer to the source qTD currently associated with the overlay. The host controller uses this pointer to write back the overlay area into the source qTD after the transfer is complete.

The following table describes the current qTD link pointer field descriptions.

Table 42-30. Current qTD Link Pointer

Bit	Description
31-5	Current Element Transaction Descriptor Link Pointer. This field contains the address Of the current transaction being processed in this queue and corresponds to memory address signals [31:5], respectively.
4-0	Reserved (R). These bits are ignored by the host controller when using the value as an address to write data. The actual value may vary depending on the usage.

The DWords 4-11 of a queue head are the transaction overlay area. This area has the same base structure as a Queue Element Transfer Descriptor. The queue head utilizes the reserved fields of the page pointers to implement tracking the state of split transactions.

This area is characterized as an overlay because when the queue is advanced to the next queue element, the source queue element is merged onto this area. This area serves as execution cache for the transfer.

The table below describes the Host-controller rules for bits in overlay.

Table 42-31. Host-Controller Rules for Bits in Overlay (DWords 5, 6, 8 and 9)

DWord	Bit	Description
5	4-1	Nak Counter (NakCnt)µRW. This field is a counter the host controller decrements whenever a transaction for the endpoint associated with this queue head results in a Nak or Nyet response. This counter is reloaded from *RL* before a transaction is executed during the first pass of the reclamation list (relative to an Asynchronous List Restart condition). It is also loaded from RL during an overlay.
6	31	Data Toggle. The *Data Toggle Control* controls whether the host controller preserves this bit when an overlay operation is performed.
6	15	Interrupt On Complete (IOC). The IOC control bit is always inherited from the source qTD when the overlay operation is performed.
6	11-10	Error Counter (C_ERR). This two-bit field is copied from the qTD during the overlay and written back during queue advancement.
6	0	Ping State (P)/ERR. If the *EPS* field indicates a high-speed endpoint, then this field should be preserved during the overlay operation.
8	7-0	Split-transaction Complete-split Progress (C-prog-mask). This field is initialized to zero during any overlay. This field is used to track the progress of an interrupt split-transaction.
9	4-0	Split-transaction Frame Tag (Frame Tag). This field is initialized to zero during any overlay. This field is used to track the progress of an interrupt split-transaction.
9	11-5	S-bytes. Software must ensure that the *S-bytes* field in a *qTD* is zero before activating the *qTD*. This field is used to keep track of the number of bytes sent or received during an IN or OUT split transaction.

42.5.2.7 Periodic Frame Span Traversal Node (FSTN)

This data structure is to be used only for managing Full- and Low-speed transactions that span a Host-frame boundary.

See Host Controller Operational Model for FSTNs for full operational details. Software must not use an FSTN in the Asynchronous Schedule. An FSTN in the Asynchronous schedule results in undefined behavior. Software must not use the FSTN feature with a host controller whose USB_HCIVERSION register indicates a revision implementation below 0096h. FSTNs are not defined for implementations before 0.96 and their use yields undefined results.

Table 42-32. Frame Span Traversal Node Structure Layout

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Addr
Normal Path Link Pointer																											0		Typ		T	03-00
Back Path Link Pointer																											0		Typ[1]		T	07-04

1. Must be set to indicate a queue head

Host Controller Read/Write

Host Controller Read Only

42.5.2.7.1 FSTN Normal Path Pointer

The first DWord of an FSTN contains a link pointer to the next schedule object. This object can be of any valid periodic schedule data type.

The following table describes the FSTN normal path pointer fields.

Table 42-33. FSTN Normal Path Pointer Field Descriptions

Bit	Description
31-5	Normal Path Link Pointer (NPLP). This field contains the address of the next data object to be processed in the periodic list and corresponds to memory address signals [31:5], respectively.
4-3	Reserved
2-1	QH/(s)iTD/FSTN Select (Typ). This field indicates to the Host Controller whether the item referenced is a iTD/siTD, a QH or an FSTN. This allows the Host Controller to perform the proper type of processing on the item after it is fetched. Value encodings are: Value Meaning 00b iTD (isochronous transfer descriptor) 01b QH (queue head) 10b siTD (split transaction isochronous transfer descriptor) 11b FSTN (Frame Span Traversal Node)
0	Terminate (T). 1 = Link Pointer field is not valid. 0 = Link Pointer is valid.

42.5.2.7.2 FSTN Back Path Link Pointer

The second DWord of an FTSN node contains a link pointer to a queue head.

If the T-bit in this pointer is zero, then this FSTN is a Save-Place indicator. Its Typ field must be set by software to indicate the target data structure is a queue head. If the T-bit in this pointer is set to one, then this FSTN is the Restore indicator. When the T-bit is one, the host controller ignores the Typ field.

The following table describes the FSTN back path link pointer fields.

Table 42-34. FSTN Back Path Link Pointer Field Descriptions

Bit	Description
31-5	Back Path Link Pointer (BPLP). This field contains the address of a Queue Head. This field corresponds to memory address signals [31:5], respectively.
4-3	Reserved
2-1	Typ. Software must ensure this field is set to indicate the target data structure is a Queue Head. Any other value in this field yields undefined results.
0	Terminate (T). 1=Link Pointer field is not valid (that is the host controller must not use bits [31:5] as a valid memory address). This value also indicates that this FSTN is a Restore indicator. 0=Link Pointer is valid (that is the host controller may use bits [31:5] (in combination with the CTRLDSSEGMENT register if applicable) as a valid memory address). This value also indicates that this FSTN is a Save-Place indicator.

42.5.3 Host Operational Model

The general operational model is for the enhanced interface host controller hardware and enhanced interface host controller driver (generally referred to as system software).

Each significant operational feature of the EHCI host controller is discussed in a separate section. Each section presents the operational model requirements for the host controller hardware. Where appropriate, recommended system software operational models for features are also presented.

42.5.3.1 Host Controller Initialization

After initial power-on or HCReset (hardware or through HCReset bit in the USB_USBCMD register), all of the operational registers are at their default values. After a hardware reset, only the operational registers not contained in the Auxiliary power well are at their default values.

The following table describes the default values of operational registers.

.

Table 42-35. Default Values of Operational Register Space

Operational Register	Default Value (after Reset)
USB_USBCMD	00080000h (00080B00h, if *Asynchronous Schedule Park Capability is one*)
USB_USBSTS	00001000h
USB_USBINTR	00000000h
USB_FRINDEX	00000000h
USB_CTRLDSSEGMENT	00000000h
USB_PERIODICLISTBASE	Undefined
USB_ASYNCLISTADDR	Undefined
USB_CONFIGFLAG	00000000h
USB_PORTSC1	00002000h (w/*PPC* set to one); 00003000h (w/*PPC* set to zero)

To initialize the host controller, software should perform the following steps:

- Write the appropriate value to the USB_USBINTR register to enable the appropriate interrupts.
- Write the base address of the Periodic Frame List to the USB_PERIODICLIST BASE register. If no work items are in the periodic schedule, all elements of the Periodic Frame List should have their T-Bits set to one.
- Write the USB_USBCMD register to set the desired interrupt threshold, frame list size (if applicable) and turn the host controller ON through setting the Run/Stop bit.

At this point, the host controller is up and running and the port registers begin reporting device connects, and so on. System software can enumerate a port through the reset process (where the port is in the enabled state). At this point, the port is active with SOFs occurring down the enabled ports, but the schedules have not enabled. To communicate with devices through the asynchronous schedule, system software must write the USB_ASYNCLISTADDR register with the address of a control or bulk queue head. Software must then enable the asynchronous schedule by writing one to the Asynchronous Schedule Enable bit in the USB_USBCMD register. To communicate with devices through the periodic schedule, system software must enable the periodic schedule by writing one to the Periodic Schedule Enable bit in the USB_USBCMD register.

NOTE

The schedules can be turned on before the first port is reset (and enabled).

When the USB_USBCMD register is written, system software must ensure the appropriate bits are preserved, depending on the intended operation.

42.5.3.2 Port Routing and Control

The EHCI specification defines that a USB 2.0 Host controller is comprised of one high-speed host controller, which implements the EHCI programming interface and 0 to N USB 1.1 companion host controllers.

Companion host controllers (cHCs) may be implementations of either Universal or Open host controller specifications. This configuration is used to deliver the required full USB 2.0-defined port capability; for example, Low-, Full-, and High-speed capability for every port.

NOTE

The USB controllers on do not require nor support companion controllers to support Full and Low Speed device. Full and Low Speed devices are supported within the USB controller by emulating the functionality of a high-speed HUB. Therefore, no port routing is present in the controller. Please refer to Embedded Transaction Translator Function for details.

The following figure illustrates a simple block diagram of the port routing logic and its relationship to the high-speed and companion host controllers within a USB 2.0 host controller.

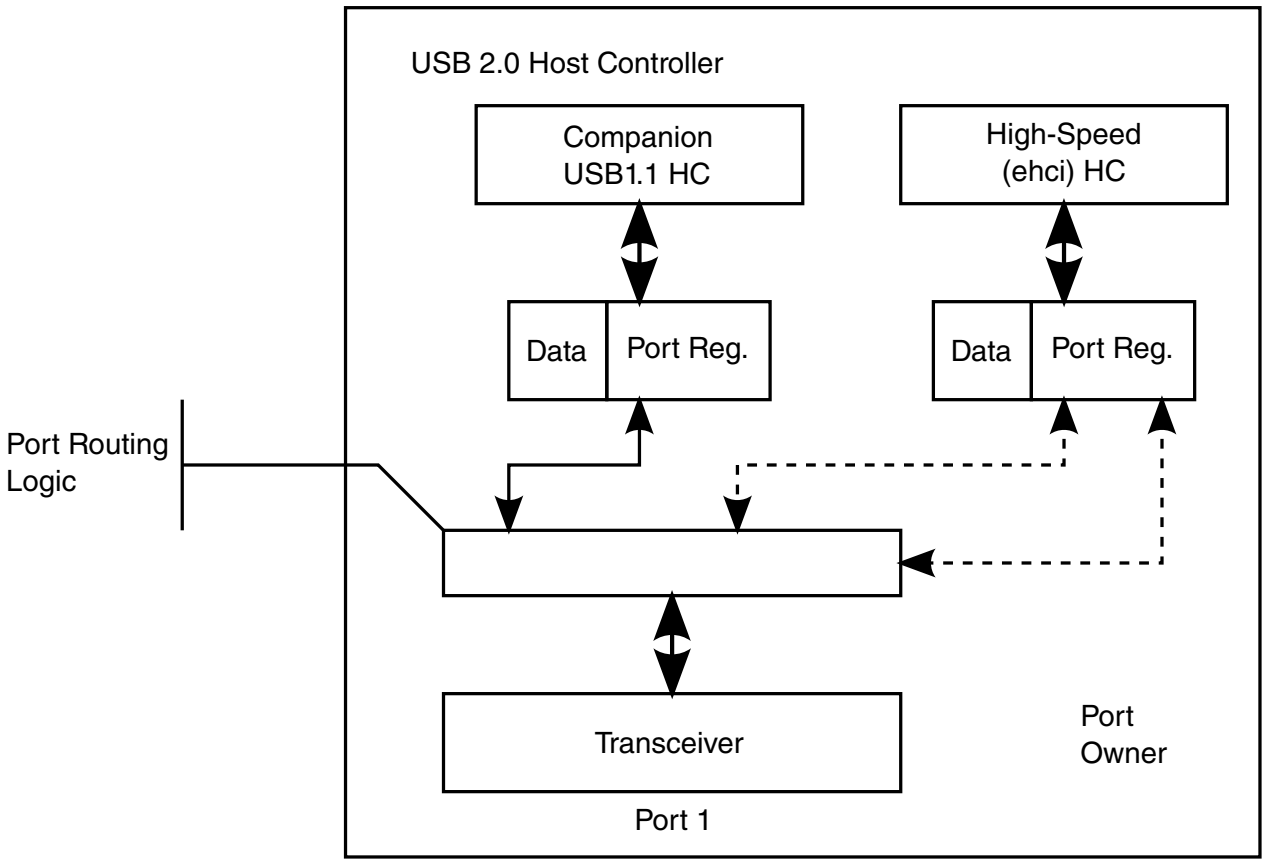


Figure 42-4. Example USB 2.0 Host Controller Port Routing Block Diagram

There exists one transceiver per physical port and each host controller block has its own port status and control registers. The EHCI controller has port status and control registers for every port. Each companion host controller has only the port control and status registers it is required to operate. Either the EHCI host controller or one companion host controller controls each transceiver. Routing logic lies between the transceiver, the port status and control registers.[1]

The port routing logic is controlled from signals originating in the EHCI host controller. The EHCI host controller has a global routing policy control field and per-port ownership control fields. The Configured Flag (CF) bit is the global routing policy control. At power-on or reset, the default routing policy is to the companion controllers (if they exist). If the system does not include a driver for the EHCI host controller and the host controller includes Companion Controllers, then the ports still work in Full- and Low-speed mode (assuming the system includes a driver for the companion controllers). In general, when the EHCI owns the ports, the companion host controllers' port registers do not see a connect indication from the transceiver. Similarly, when a companion host controller owns a port, the EHCI controller's port registers do not see a connect indication

1. The routing logic should not be implemented in the 480 MHz clock domain of the transceiver.

from the transceiver. The details on the rules for the port routing logic are described in the following sections. The USB 2.0 host controller must be implemented as a multi-function PCI device if the implementation includes companion controllers. The companion host controllers' function numbers must be less than the EHCI host controller function number. The EHCI host controller must be a larger function number with respect to the companion host controllers associated with this EHCI host controller. If a PCI device implementation contains only an EHCI controller (that is no companion controllers or other PCI functions), then the EHCI host controller must be function zero, in accordance with the PCI Specification. The N_CC field in the Structural Parameter register (HCSPARAMS) indicates whether the controller implementation includes companion host controllers. When N_CC has a non-zero value there exists companion host controllers. If N_CC has a value of zero, then the host controller implementation does not include companion host controllers. If the host controller root ports are exposed to attachment of full- or low-speed devices, the ports always fails the high-speed chirp during reset and the ports are not enabled. System software can notify the user of the illegal condition. This type of implementation requires a USB 2.0 hub be connected to a root port to provide full and low-speed device connectivity.

System software uses information in the host controller capability registers to determine how the ports are routed to the companion host controllers. See Host Controller Structural Parameters (USB_*n*HCSPARAMS)

42.5.3.2.1 Port Routing Control through EHCI Configured (CF) Bit

Each port in the USB 2.0 host controller are routed either to a single companion host controller or to the EHCI host controller.

The port routing logic is controlled by two mechanisms in the EHCI HC: a host controller global flag and per-port control. The Configured Flag (CF) bit, is used to globally set the policy of the routing logic. Each port register has a Port Owner control bit which allows the EHCI Driver to explicitly control the routing of individual ports. Whenever the CF bit transitions from zero to one (this transition is only available under program control) the port routing unconditionally routes all of the port registers to the EHCI HC (all Port Owner bits go to zero). While the CF-bit is one, the EHCI Driver controls individual ports' routing through the Port Owner control bit. Likewise, whenever the CF bit transitions from one to zero (as a result of Aux power application, HCRESET, or software writing zero to CF-bit), the port routing unconditionally routes all of the port registers to the appropriate companion HC. The default value for the EHCI HC's CF bit (after Aux power application or HCRESET) is zero.

The *view* of the port depends on the current owner. A Universal or Open companion host controller will see port register bits consistent with the appropriate specification. Port bit definitions that are required for EHCI host controllers are not visible to companion host controllers.

The following table summarizes the default routing for all the ports, based on the value of the EHCI HC's CF bit.

Table 42-36. Default Port Routing Depending on EHCI HC CF Bit

HS CF Bit	Default Port Ownership	Explanation
0B	Companion HCs	The companion host controllers own the ports and only Full- and Low-speed devices are supported in the system. The exact port assignments are implementation dependent. The ports behave only as Full- and Low-speed ports in this configuration
1B	EHCI HC	The EHCI host controller has default ownership over all of the ports. The routing logic inhibits device connect events from reaching the companion HCs' port status and control registers when the port owner is the EHCI HC.The EHCI HC has access to the additional port status and control bits defined in this specification (see Port Status & Control (USB_*n*PORTSC1)). The EHCI HC can temporarily release control of the port to a companion HC by setting the *PortOwner* bit in the PORTSC1 register to one.

42.5.3.2.2 Port Routing Control through PortOwner and Disconnect Event

Manipulating the port routing through the CF-bit is an extreme process and not intended to be used during normal operation.

The normal mode of port ownership transferal is on the granularity of individual ports using the Port Owner bit in the EHCI HC's USB_PORTSC1 register (for hand-offs from EHCI to companion host controllers). Individual port ownership is returned to the EHCI controller when the port registers a device disconnect. When the disconnect is detected, the port routing logic immediately returns the port ownership to the EHCI controller. The companion host controller port register detects the device disconnect and operates normally.

Under normal operating conditions (assuming all HC drivers loaded and operational and the EHCI *CF-bit* is set to one), the typical port enumeration sequence proceeds as illustrated below:

- Initial condition is that EHCI is port owner. A device is connected causing the port to detect a connect, set the port connect change bit and issue a port-change interrupt (if enabled).
- EHCI Driver identifies the port with the new connect change bit asserted and sends a change report to the hub driver. Hub driver issues a GetPortStatus() request and

identifies the connect change. It then issues a request to clear the connect change, followed by a request to reset and enable the port.

- When the EHCI Driver receives the request to reset and enable the port, it first checks the value reported by the LineStatus bits in the USB_PORTSC1 register. If they indicate the attached device is a full-speed device (for example, D+ is asserted), then the EHCI Driver sets the PortReset control bit to one (and sets the PortEnable bit to zero) which begins the reset-process. Software times the duration of the reset, then terminates reset signaling by writing zero to the port reset bit. The reset process is actually complete when software reads zero in the PortReset bit. The EHCI Driver checks the PortOwner bit in the USB_PORTSC1 register. If set to one, the connected device is a high-speed device and EHCI Driver (root hub emulator) issues a change report to the hub driver and the hub driver continues to enumerate the attached device.
- At the time the EHCI Driver receives the port reset and enable request the LineStatus bits might indicate a low-speed device. Additionally, when the port reset process is complete, the PortEnable field may indicate that a full-speed device is attached. In either case the EHCI driver sets the PortOwner bit in the USB_PORTSC1 register to one to release port ownership to a companion host controller.
- When the EHCI Driver sets PortOwner bit to one, the port routing logic makes the connection state of the transceiver available to the companion host controller port register and removes the connection state from the EHCI HC port. The EHCI USB_PORTSC1 register observes and reports a disconnect event through the disconnect change bit. The EHCI Driver detects the connection status change (either by polling or by port change interrupt) and then sends a change report to the hub driver. When the hub driver requests that port-state, the EHCI Driver responds with a reset complete change set to one, a connect change set to one and a connect status set to zero. This information is derived directly from the EHCI port register. This allows the hub driver to assume the device was disconnected during reset. It acknowledges the change bits and wait for the next change event. While the EHCI controller does not own the port, it simply remains in a state where the port reports no device connected. The device-connect evaluation circuitry of the companion HC activates and detects the device, the companion Driver detects the connection and enumerates the port.

When a port is routed to a companion HC, it remains under the control of the companion HC until the device is disconnected from the root port (ignoring for now the scenario where EHCI's CF-bit transitions from1b to 0b). When a disconnect occurs, the disconnect event is detected by both the companion HC port control and the EHCI port ownership control. On the event, the port ownership is returned immediately to the EHCI controller. The companion HC stack detects the disconnect and acknowledges as it would in an ordinary standalone implementation. Subsequent connects is detected by the EHCI port register and the process repeats.

42.5.3.2.3 Example Port Routing State Machine

The following figure illustrates an example of how the port ownership should be managed. The following sections describe the entry conditions to each state.

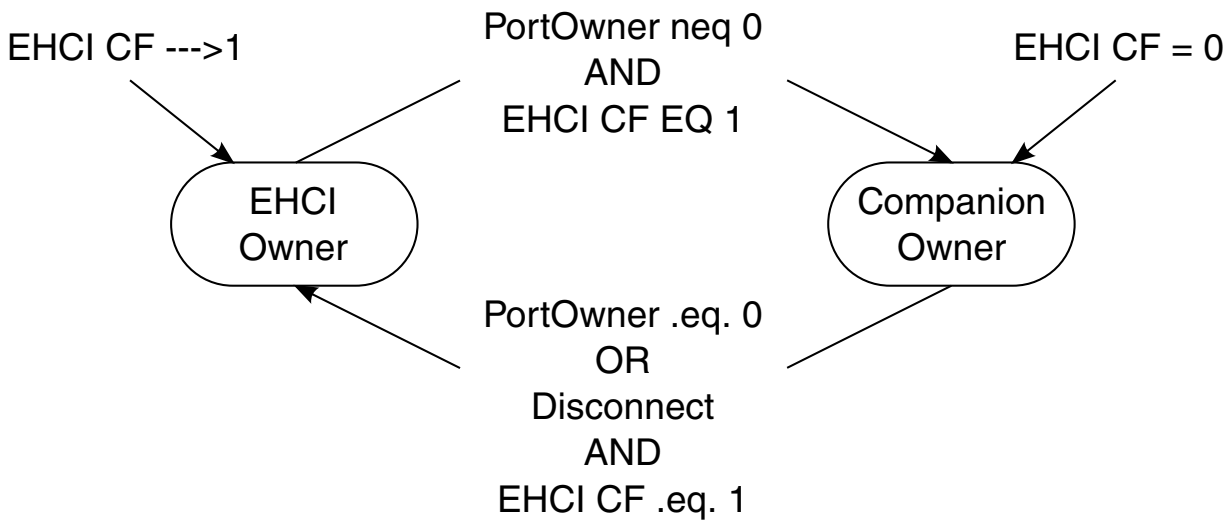


Figure 42-5. Port Owner Handoff State Machine

42.5.3.2.3.1 EHCI HC Owner

Entry to this state occurs when one of the following events occur:

- When the EHCI HC's Configure Flag (CF) bit in the USB_CONFIGFLAG register transitions from zero to one. This signals the fact that the system has a host controller driver for the EHCI HC and that all ports in the USB 2.0 host controller must default route to the EHCI controller.
- When the port is owned by a companion HC and the device is disconnected from the port. The EHCI port routing control logic is notified of the disconnect, and returns port routing to the EHCI controller. The connection state of the companion HC goes immediately to the disconnected state (with appropriate side effect to connect change, enable and enable change). The companion HC driver acknowledges the disconnect by setting the connect status change bit to zero. This allows the companion HC's driver to interact with the port completely through the disconnect process.
- When system software writes zero to the PortOwner bit in the USB_PORTSC1 register. This allows software to take ownership of a port from a companion host controller. When this occurs, the routing logic to the companion HC effectively signals a disconnect to the companion HC's port status and control register.

42.5.3.2.3.2 Companion HC Owner

Entry to this state occurs whenever one of the following events occur:

- When the PortOwner field transitions from zero to one.
- When the HS-mode HC's Configure Flag (CF) is equal to zero.

On entry to this state, the routing logic allows the companion HC port register to detect a device connect. Normal port enumeration proceeds.

42.5.3.2.4 Port Power

The Port Power Control (PPC) bit in the USB_HCSPARAMS register indicates whether the USB 2.0 host controller has port power control (see Host Controller Structural Parameters (USB_*n*HCSPARAMS)).

When this bit is zero, then the host controller does not support software control of port power switches. When in this configuration, the port power is always available and the companion host controllers must implement functionality consistent with port power always on. When the *PPC* bit is one, then the host controller implementation includes port power switches. Each available switch has an output enable, which is referred to in this discussion as PortPowerOutputEnable (PPE). PPE is controlled based on the state of the combination bits PPC bit, EHCI Configured (CF)-bit and individual Port Power (PP) bits.

The following table describes the summary behavioral model.

Table 42-37. Port Power Enable Control Rules

CF	CHC[1] (PP)	EHC[2] (PP)	Owner	PPE[3]	Description
0	0	X	CHC	0	When the EHCI controller is not configured, the port is owned by the companion host controller. When the companion HC's port power select is off, then the port power is off.
0	1	X	CHC	1	Similar to previous entry. When the companion HC's port power select is on, then the port power is on.
1	0	0	CHC	0	Port owner has port power turned off, the power to port is off.
1	0	0	EHC	0	Port owner has port power turned off, the power to port is off.
1	0	1	EHC	1	Port owner has port power on, so power to port is on.
1	0	1	CHC	1	If either HC has port power turned on, the power to the port is on.

Table continues on the next page...

Table 42-37. Port Power Enable Control Rules (continued)

1	1	0	EHC	1	If either HC has port power turned on, the power to the port is on.
1	1	0	CHC	1	Port owner has port power on, so power to port is on.
1	1	1	CHC	1	Port owner has port power on, so power to port is on.
1	1	1	EHC	1	Port owner has port power on, so power to port is on.

1. CHC (Companion Host Controller).
2. EHC (EHCI Host Controller).
3. PPE (Port Power Enable). This bit actually turns on the port power switch (if one exists).

42.5.3.2.5 Port Reporting Over-Current

Host controllers are by definition power providers on USB. Whether the ports are considered high- or low-powered is a platform implementation issue. Each EHCI USB_PORTSC1 register has an over-current status and over-current change bit.

The functionality of these bits are specified in the USB Specification Revision 2.0.

The over current detection and limiting logic usually resides outside the host controller logic. This logic may be associated with one or more ports. When this logic detects an over-current condition it is made available to both the companion and EHCI ports. The effect of an over-current status on a companion host controller port is beyond the scope of this document.

The over-current condition effects the following bits in the USB_PORTSC1 register on the EHCI port:

- Over-current Active bits are set to one. When the over-current condition goes away, the Over-current Active bit transitions from one to zero.
- Over-current Change bits are set to one. On every transition of the Over-current Active bit the host controller sets the Over-current Change bit to one. Software sets the Over-current Change bit to zero by writing one to this bit.
- Port Enabled/Disabled bit is set to zero. When this change bit gets set to one, then the Port Change Detect bit in the USB_USBSTS register is set to one.
- Port Power (PP) bits may optionally be set to zero. There is no requirement in USB that a power provider shut off power in an over current condition. It is sufficient to limit the current and leave power applied. When the Over-current Change bit transitions from zero to one, the host controller also sets the Port Change Detect bit in the USB_USBSTS register to one. In addition, if the Port Change Interrupt Enable bit in the USB_USBINTR register is one, then the host controller issues an interrupt to the system. Refer to Table 42-38 for summary behavior for over-current detection

when the host controller is halted (suspended from a device component point of view).

42.5.3.3 Suspend/Resume-Host Operational Model

The EHCI host controller provides an equivalent suspend and resume model as that defined for individual ports in a USB 2.0 Hub.

Control mechanisms are provided to allow system software to suspend and resume individual ports. The mechanisms allow the individual ports to be resumed completely through software initiation. Other control mechanisms are provided to parameterize the host controller's response (or sensitivity) to external resume events. In this discussion, host-initiated, or software initiated resumes are called Resume Events/Actions. Bus-initiated resume events are called wake-up events. The classes of wake-up events are:

- Remote-wake-up enabled device asserts resume signaling. In similar kind to USB 2.0 Hubs, EHCI controllers must always respond to explicit device resume signaling and wake-up the system (if necessary).
- Port connect and disconnect and over-current events. Sensitivity to these events can be turned on or off by using the per-port control bits in the USB_PORTSC1 registers.

Selective suspend is a feature supported by every USB_PORTSC1 register. It is used to place specific ports into a suspend mode. This feature is used as a functional component for implementing the appropriate power management policy implemented in a particular operating system. When system software intends to suspend the entire bus, it should selectively suspend all enabled ports, then shut off the host controller by setting the Run/Stop bit in the USB_USBCMD register to zero. The EHCI sub-block can then be placed into a lower device state through the PCI power management interface (see Appendix A, Enhanced Host Controller Interface Specification for Universal Serial Bus, Revision 0.95, November 2000, Intel Corporation. http://www.intel.com).

When a wake event occurs, the system resumes operation and system software eventually set the Run/Stop bit to one and resume the suspended ports. Software must not set the Run/Stop bit to one until it is confirmed that the clock to the host controller is stable. This is usually confirmed in a system implementation in that all of the clocks in the system are stable before the Arm platform is restarted. So, by definition, if software is running, clocks in the system are stable and the Run/Stop bit in the USB_USBCMD register can be set to one. Minimum system software delays are also defined in the PCI Power Management Specification. Refer to PCI Power Management Specification for more information.

42.5.3.3.1 Port Suspend/Resume

System software places individual ports into suspend mode by writing one into the appropriate USB_PORTSC1 Suspend bit. Software must only set the Suspend bit when the port is in the enabled state (Port Enabled bit is one) and the EHCI is the port owner (PortOwner bit is zero).

The host controller may evaluate the Suspend bit immediately or wait until a micro-frame or frame boundary occurs. If evaluated immediately, the port is not suspended until the current transaction (if one is executing) completes. Therefore, there may be several micro-frames of activity on the port until the host controller evaluates the Suspend bit. The host controller must evaluate the Suspend bit at least every frame boundary.

System software can initiate a resume on a selectively suspended port by writing one to the Force Port Resume bit. Software should not attempt to resume a port unless the port reports that it is in the suspended state (see Port Status & Control (USB_*n*PORTSC1)). If system software sets Force Port Resume bit to one when the port is not in the suspended state, the resulting behavior is undefined. In order to assure proper USB device operation, software must wait for at least 10 ms after a port indicates that it is suspended (Suspend bit is one) before initiating a port resume through the Force Port Resume bit. When Force Port Resume bit is one, the host controller sends resume signaling down the port. System software times the duration of the resume (nominally 20 ms) then sets the Force Port Resume bit to zero. When the host controller receives the write to transition Force Port Resume to zero, it completes the resume sequence as defined in the USB specification, and sets both the Force Port Resume and Suspend bits to zero. Software-initiated port resumes do not affect the Port Change Detect bit in the USB_USBSTS register nor do they cause an interrupt if the Port Change Interrupt Enable bit in the USB_USBINTR register is one. An external USB event may also initiate a resume. The wake events are defined above. When a wake event occurs on a suspended port, the resume signaling is detected by the port and the resume is reflected downstream within 100 µsec. The port's Force Port Resume bit is set to one and the Port Change Detect bit in the USB_USBSTS register is set to one. If the Port Change Interrupt Enable bit in the USB_USBINTR register is one the host controller issues a hardware interrupt.

System software observes the resume event on the port, delays a port resume time (nominally 20 ms), then terminates the resume sequence by writing zero to the Force Port Resume bit in the port. The host controller receives the write of zero to Force Port Resume, terminates the resume sequence and sets Force Port Resume and Suspend port bits to zero. Software can determine that the port is enabled (not suspended) by sampling the USB_PORTSC1 register and observing that the Suspend and Force Port Resume bits are zero. Software must ensure that the host controller is running (that is HCHalted bit in the USB_USBSTS register is zero), before terminating a resume by writing zero to a

port's Force Port Resume bit. If HCHalted is one when Force Port Resume is set to zero, then SOFs do not occur down the enabled port and the device returns to suspend mode in a maximum of 10 msec.

The table below summarizes the wake-up events. Whenever a resume event is detected, the Port Change Detect bit in the USB_USBSTS register is set to one. If the Port Change Interrupt Enable bit is one in the USB_USBINTR register, the host controller generates an interrupt on the resume event. Software acknowledges the resume event interrupt by clearing the Port Change Detect status bit in the USB_USBSTS register.

Table 42-38. Behavior During Wake-up Events

Port Status and Signaling Type	Signaled Port Response	Device State	
		D0	Not D0
Port disabled, resume K-State received	No Effect	N/A	N/A
Port suspended, resume K-State received	Resume reflected downstream on signaled port. Force Port Resume status bit in USB_PORTSC1 register is set to one. Port Change Detect bit in USB_USBSTS register set to one.	[1], [2]	[2]
Port is enabled, disabled or suspended, and the port's WKDSCNNT_E bit is one. A disconnect is detected.	Depending in the initial port state, theUSB_ PORTSC1 Connected Enable status bits are set to zero, and the Connect Change status bit is set to one. Port Change Detect bit in the USB_USBSTS register is set to one.	[1], [2]	[2]
Port is enabled, disabled or suspended, and the port's WKDSCNNT_E bit is zero. A disconnect is detected.	Depending on the initial port state, the USB_PORTSC1 Connect and Enable status bits are set to zero, and the Connect Change status bit is set to one. Port Change Detect bit in the USB_USBSTS register is set to one.	[1], [3]	[3]
Port is not connected and the port's WKCNNT_E bit is one. A connect is detected.	USB_PORTSC1 Connect Status and Connect Status Change bits are set to one. Port Change Detect bit in the USB_USBSTS register is set to one.	[1], [2]	[2]
Port is not connected and the port's WKCNNT_E bit is zero. A connect is detected.	USB_PORTSC1 Connect Status and Connect Status Change bits are set to one. Port Change Detect bit in the USB_USBSTS register is set to one.	[1], [3]	[3]
Port is connected and the port's WKOC_E bit is one. An over-current condition occurs.	USB_PORTSC1 Over-current Active, Over-current Change bits are set to one. If Port Enable/Disable bit is one, it is set to zero. Port Change Detect bit in the USB_USBSTS register is set to one	[1], [2]	[2]
Port is connected and the port's WKOC_E bit is zero. An over-current condition occurs.	USB_PORTSC1 Over-current Active, Over-current Change bits are set to one. If Port Enable/Disable bit is one, it is set to zero. Port Change Detect bit in the USB_USBSTS register is set to one.	[1], [3]	[3]

[1] Hardware interrupt issued if Port Change Interrupt Enable bit in the USB_USBINTR register is one.

[2] PME# asserted if enabled (Note: PME Status must always be set to one).

[3] PME# not asserted.

42.5.3.4 Schedule Traversal Rules

The host controller executes transactions for devices using a simple, shared-memory schedule.

The schedule is comprised of a few data structures, organized into two distinct lists. The data structures are designed to provide the maximum flexibility required by USB, minimize memory traffic and hardware / software complexity.

System software maintains two schedules for the host controller: a periodic schedule and an asynchronous schedule. The root of the periodic schedule is the USB_PERIODICLISTBASE register (see Frame List Base Address (USB_*n*PERIODICLISTBASE))/ Device Address (USB_*n*DEVICEADDR). The USB_PERIODICLISTBASE register is the physical memory base address of the periodic frame list. The periodic frame list is an array of physical memory pointers. The objects referenced from the frame list must be valid schedule data structures as defined in Host Data Structures. In each micro-frame, if the periodic schedule is enabled (see Periodic scheduling threshold) then the host controller must execute from the periodic schedule before executing from the asynchronous schedule. It only executes from the asynchronous schedule after it encounters the end of the periodic schedule. The host controller traverses the periodic schedule by constructing an array offset reference from the USB_PERIODICLISTBASE and the USB_FRINDEX registers (see the following figure). It fetches the element and begins traversing the graph of linked schedule data structures.

The end of the periodic schedule is identified by a next link pointer of a schedule data structure having its T-bit set to one. When the host controller encounters a T-Bit set to one during a horizontal traversal of the periodic list, it interprets this as an End-Of-Periodic-List mark. This causes the host controller to cease working on the periodic schedule and transitions immediately to traversing the asynchronous schedule. After the transition, the host controller executes from the asynchronous schedule until the end of the micro-frame.

The following figure illustrates the derivation of pointer into frame list array.

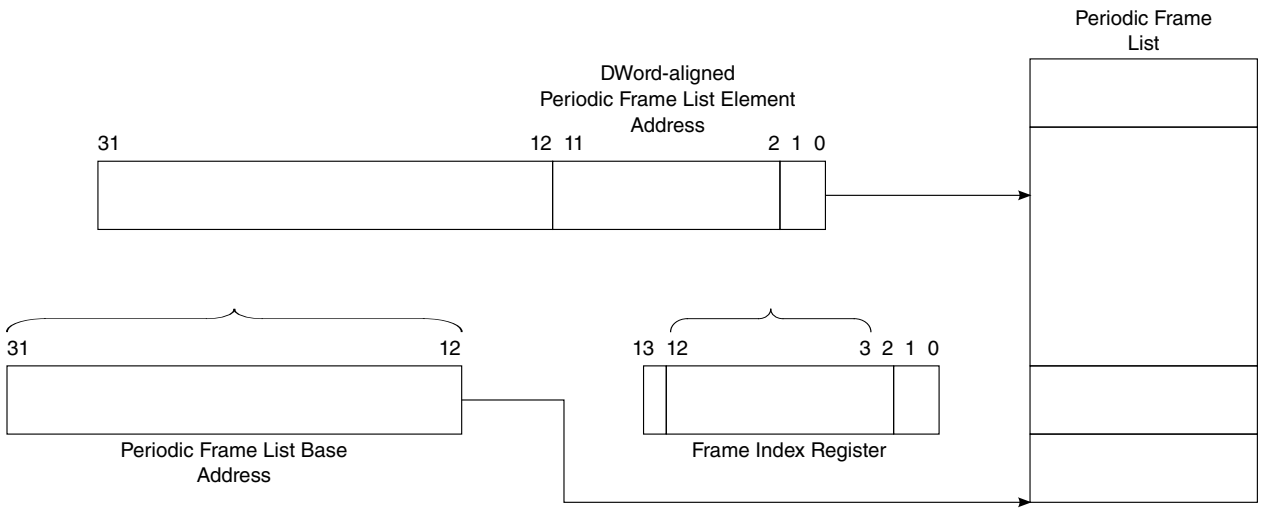


Figure 42-6. Derivation of Pointer into Frame List Array

When the host controller determines that it is the time to execute from the asynchronous list, it uses the operational register USB_ASYNCLISTADDR to access the asynchronous schedule, see the figure below.

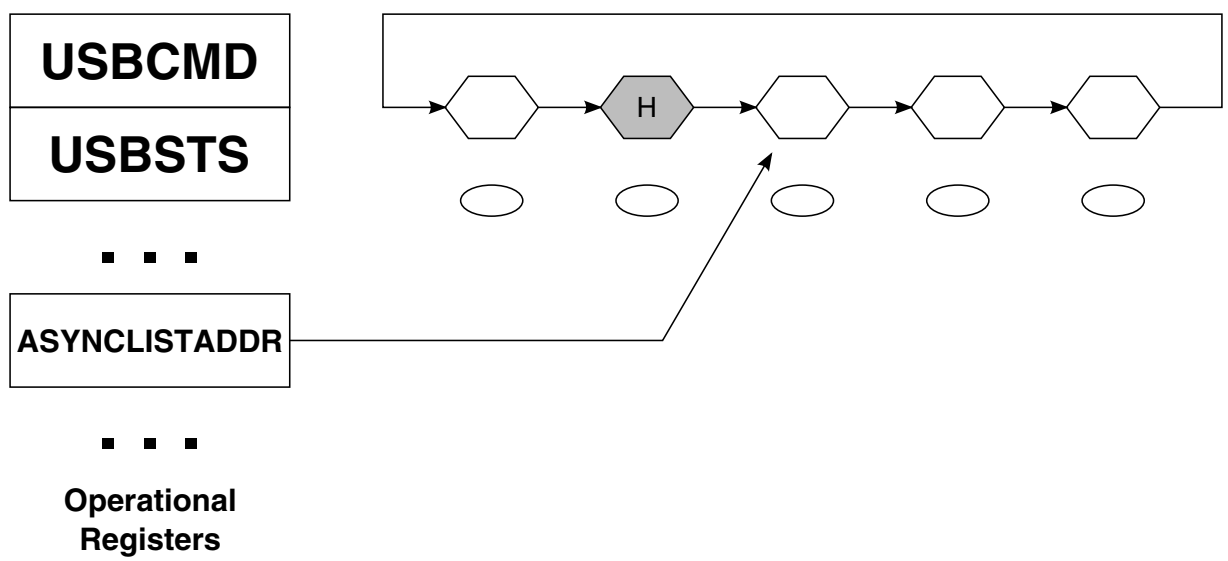


Figure 42-7. General Format of Asynchronous Schedule List

The USB_ASYNCLISTADDR register contains a physical memory pointer to the next queue head. When the host controller makes a transition to executing the asynchronous schedule, it begins by reading the queue head referenced by the USB_ASYNCLISTADDR register. Software must set queue head horizontal pointer T-bits to zero for queue heads in the asynchronous schedule. See Asynchronous Schedule for complete operational details.

42.5.3.4.1 Example - Preserving Micro-Frame Integrity

One of the requirements of a USB host controller is to maintain Frame Integrity. This means that the HC must preserve the micro-frame boundaries.

For example, SOF packets must be generated on time (within the specified allowable jitter), and High-speed EOF1,2 thresholds must be enforced. The end of micro-frame timing points EOF1 and EOF2 are clearly defined in the USB Specification Revision 2.0. One implication of this responsibility is that the HC must ensure that it does not start transactions that do not complete before the end of the micro-frame. More precisely, no transactions should be started by the host controller, which do not complete in their entirety before the EOF1 point. In order to enforce this rule, the host controller must check each transaction before it starts to ensure that it completes before the end of the micro-frame.

So, what exactly needs to be involved in this check? Fundamentally, the transaction data payload, plus bit stuffing, plus transaction overhead must be taken into consideration. It is possible to be extremely accurate on how much time the next transaction takes. Take OUTs for an example. The host controller must fetch all of the OUT data from memory in order to send it onto the USB bus. A host controller implementation could pre-fetch all of the OUT data, and pre-compute the actual number of bits in the token and data packets. In addition, the system knows the depth of the target endpoint, so it could closely estimate turnaround time for handshake. In addition, the host controller knows the size of a handshake packet. Pre-computing effects of bit stuffing and summing up the other overhead numbers can allow the host controller to know exactly whether there is enough bus time, before EOF1 to complete the OUT transaction. To accomplish this particular approach takes an inordinate amount of time and hardware complexity.

The alternative is to make a reasonable guess whether the next transaction can be started. An example approximation algorithm is described below. This example algorithm relies on the EHCI policy that periodic transactions are scheduled first in the micro-frame. It is a reasonable assumption that software never over-commits the micro-frame to periodic transactions greater than the specification allowable 80%. In the available remaining 20% bandwidth, the host controller has some ability (in this example) to decide whether or not to execute a transaction. The result of this algorithm is that sometimes, under some circumstances a transaction is not executed that could have been executed. However, under all circumstances, a transaction is never started unless there is enough time in the frame to complete the transaction.

42.5.3.4.1.1 Transaction Fit - A Best-Fit Approximation Algorithm

A curve is calculated which represents the latest start time for every packet size, at which software schedules the start of a periodic transaction.

This curve is the 80% bandwidth curve. Another curve is calculated which is the absolute, latest permitted start time for every packet size. This curve represents the absolute latest time, that a transaction of each packet size can be started and completed, in the micro-frame. A plot of these two curves are illustrated in Figure 42-8. The plot Y-axis represents the number of byte-times left in a frame.

The space between the 80% and the Last Start plots is bandwidth reclamation area. In this algorithm the host controller may skip transactions during this time if it is prudent.

The Best-Fit Approximation method plots a function ($f(x)$) between the 80% and Last Start curves. The function $f(x)$ adds a constant to every transaction's maximum packet size and the result compared with the number of bytes left in the frame. The constant represents an approximation of the effects of bit stuffing and protocol overhead. The host controller starts transactions whose results land above the function curve. The host controller will not start transactions whose results land below the function curve.

The following figure illustrates the Best-Fit Approximation.

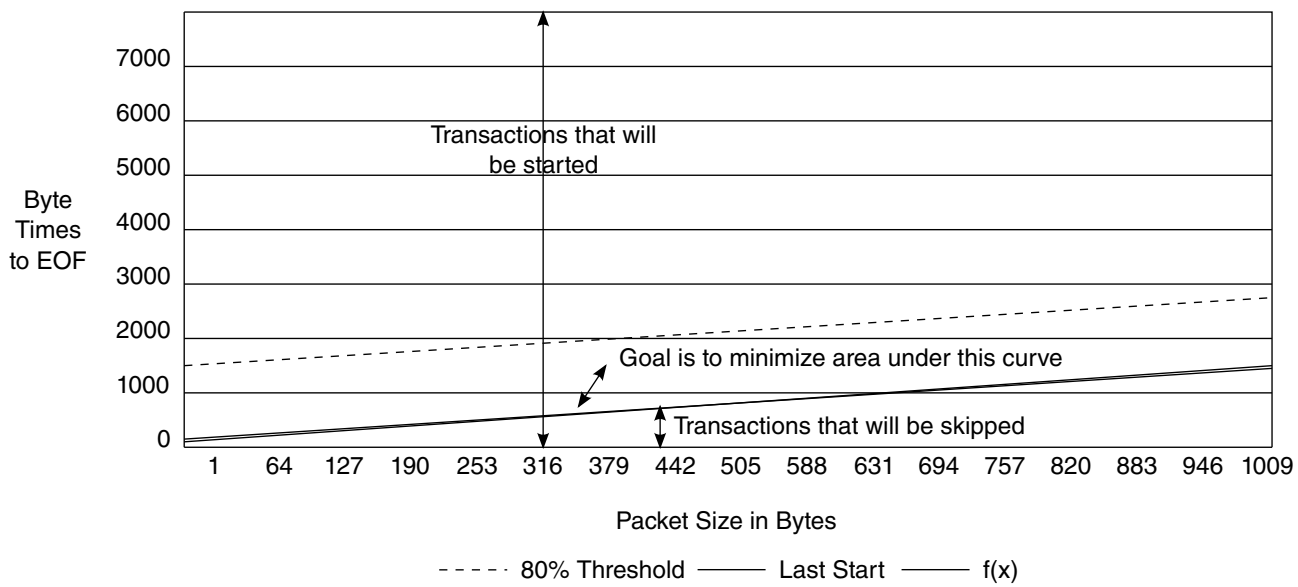


Figure 42-8. Best Fit Approximation

The LastStart line was calculated in this example to assume the absolute worst-case bus overhead per transaction. The particular transaction used is a start-split, zero-length OUT transaction with a handshake. Summaries of the component parts are listed in the table below. The component times were derived from the protocol timings defined in the USB Specification Revision 2.0.

Table 42-39. Example Worse-case Transaction Timing Components

Component	Bit time	Byte Time	Explanation

Table continues on the next page...

Table 42-39. Example Worse-case Transaction Timing Components (continued)

Split Token	76	9.5	Split token as defined in USB core specification. Includes sync, token, eop, and so on.
Host 2 Host IPG	88	11	Number of bit times required between consecutive host packets.
Token	67	8.375	Token as defined in USB core specification. Includes sync, token, eop, and so on.
Host 2 Host IPG	88	11	Token as defined in USB core specification. Includessync, token, eop, and so on.
Data Packet (0 data bytes)	66.7	8.34	Zero-length data packet. Includes sync, PID, crc16, eop, and so on.
Turnaround time	721	90.125	Time for packet initiator (Host) to see the beginning of a response to a transmitted packet.
Handshake packet	48	6	Handshake packet as defined in USB core specification. Includes sync, PID, eop, and so on.
		144	Total

The exact details of the function (f(x)) are up to the particular implementation. However, it should be obvious that the goal is to minimize the area under the curve between the approximation function and the Last Start curve, without dipping below the LastStart line, while at the same time keeping the check as simple as possible for hardware implementation. The f(x) in Figure 42-8 was constructed using the following pseudo-code test on each transaction size data point. This algorithm assumes that the host controller keeps track of the remaining bits in the frame.

```
Alorithm CheckTransactionWillFit (MaximumPacketSize, HC_BytesLeftInFrame)
Begin
Local Temp = MaximumPacketSize + 192
Local rvalue = TRUE
If MaximumPacketSize >= 128 then
     Temp += 128
End If
If Temp > HC_BytesLeftInFrame then
     Rvalue = FALSE
End If
Return rvalue
End
```

This algorithm takes two inputs, the current maximum packet size of the transaction and the hardware counter of the number of bytes left in the current micro-frame. It unconditionally adds a simple constant of 192 to the maximum packet size to account for a first-order effect of transaction overhead and bit stuffing. If the transaction size is greater than or equal to 128 bytes, then an additional constant of 128 is added to the running sum to account for the additional worst-case bit stuffing of payloads larger than 128. An inflection point was inserted at 128 because the $f(x)$ plot was getting close to the LastStart line.

42.5.3.5 Periodic Schedule Frame Boundaries vs Bus Frame Boundaries

The USB Specification Revision 2.0 requires that the frame boundaries (SOF frame number changes) of the high-speed bus and the full- and low-speed bus(s) below USB 2.0 Hubs be strictly aligned.

Super-imposed on this requirement is that USB 2.0 Hubs manage full- and low-speed transactions through a micro-frame pipeline (see start- (SS) and complete- (CS) splits illustrated in the following figure). A simple, direct projection of the frame boundary model into the host controller interface schedule architecture creates tension (complexity for both hardware and software) between the frame boundaries and the scheduling mechanisms required to service the full- and low-speed transaction translator periodic pipelines.

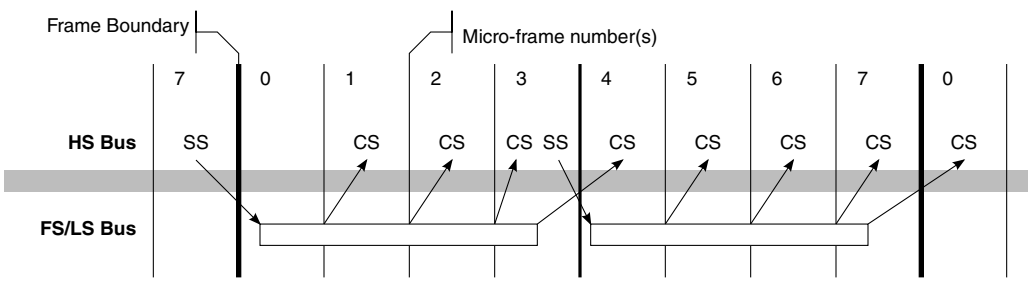


Figure 42-9. Frame Boundary Relationship between HS bus and FS/LS Bus

The simple projection, as the above figure illustrates, introduces frame-boundary wrap conditions for scheduling on both the beginning and end of a frame. In order to reduce the complexity for hardware and software, the host controller is required to implement one micro-frame phase shift for its view of frame boundaries. The phase shift eliminates the beginning of frame and frame-wrap scheduling boundary conditions.

The implementation of this phase shift requires that the host controller use one register value for accessing the periodic frame list and another value for the frame number value included in the SOF token. These two values are separate, but tightly coupled. The periodic frame list is accessed through the Frame List Index Register (USB_FRINDEX) documented in USB Frame Index (USB_*n*FRINDEX) and initially illustrated in Schedule Traversal Rules. Bits FRINDEX[2:0], represent the micro-frame number. The SOF value is coupled to the value of FRINDEX[13:3]. Both FRINDEX[13:3] and the SOF value are increment based on FRINDEX[2:0]. It is required that the SOF value be delayed from the FRINDEX value by one micro-frame. The one micro-frame delay yields host controller periodic schedule and bus frame boundary relationship as illustrated in the following figure. This adjustment allows software to trivially schedule the periodic start and

complete-split transactions for full-and low-speed periodic endpoints, using the natural alignment of the periodic schedule interface. The reasons for selecting this phase-shift are beyond the scope of this specification.

The following figure illustrates how periodic schedule data structures relate to schedule frame boundaries and bus frame boundaries. To aid the presentation, two terms are defined: The host controller's view of the 1 msec boundaries is called H-Frames. The high-speed bus's view of the 1 msec boundaries is called B-Frames.

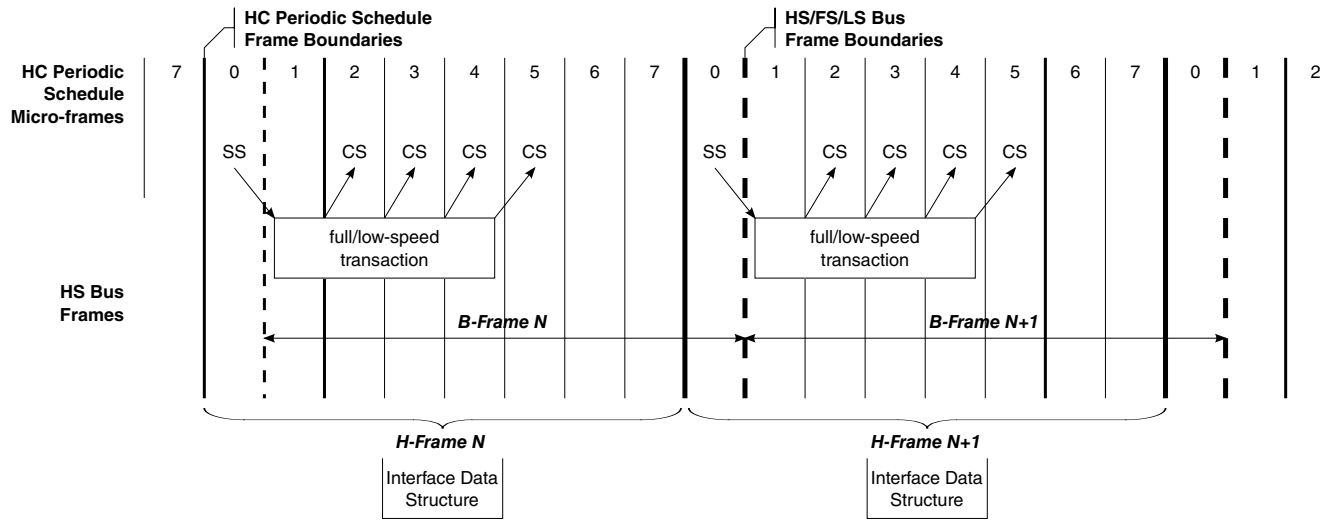


Figure 42-10. Relationship of Periodic Schedule Frame Boundaries to Bus Frame Boundaries

H-Frame boundaries for the host controller correspond to increments of FRINDEX[13:3]. Micro-frame numbers for the H-Frame are tracked by FRINDEX[2:0]. B-Frame boundaries are visible on the high-speed bus through changes in the SOF token's frame number. Micro-frame numbers on the high-speed bus are only derived from the SOF token's frame number (that is the high-speed bus sees eight SOFs with the same frame number value). H-Frames and B-Frames have the fixed relationship (that is B-Frames lag H-Frames by one micro-frame time) illustrated in the figure above. The host controller's periodic schedule is naturally aligned to H-Frames. Software schedules transactions for full- and low-speed periodic endpoints relative the H-Frames. The result is these transactions execute on the high-speed bus at exactly the right time for the USB 2.0 Hub periodic pipeline. As described in USB Frame Index (USB_*n*FRINDEX), the SOF Value can be implemented as a shadow register (in this example, called SOFV), which lags the FRINDEX register bits [13:3] by one micro-frame count. This lag behavior can be accomplished by incrementing FRINDEX[13:3] based on carry-out on the 7 to 0 increment of FRINDEX[2:0] and incrementing SOFV based on the transition of 0 to 1 of FRINDEX[2:0].

Software is allowed to write to FRINDEX. USB Frame Index (USB_*n*FRINDEX) provides the requirements that software should adhere when writing a new value in FRINDEX.

The table below illustrates the required relationship between the value of FRINDEX and the value of SOFV.

Table 42-40. Operation of FRINDEX and SOFV (SOF Value Register)

Current			Next		
FRINDEX[F]	SOFV	FRINDEX[mF]	FRINDEX[F]	SOFV	FRINDEX[mF]
N	N	111b	N+1	N	000b
N+1	N	000b	N+1	N+1	001b
N+1	N+1	001b	N+1	N+1	010b
N+1	N+1	010b	N+1	N+1	011b
N+1	N+1	011b	N+1	N+1	100b
N+1	N+1	100b	N+1	N+1	101b
N+1	N+1	101b	N+1	N+1	110b
N+1	N+1	110b	N+1	N+1	111b

NOTE

Where [F] = [13:3]; [μF] = [2:0]

42.5.3.6 Periodic Schedule

The periodic schedule traversal is enabled or disabled through the Periodic Schedule Enable bit in the USB_USBCMD register. If the Periodic Schedule Enable bit is set to zero, then the host controller simply does not try to access the periodic frame list through the USB_PERIODICLISTBASE register. Likewise, when the Periodic Schedule Enable bit is one, then the host controller does use the USB_PERIODICLISTBASE register to traverse the periodic schedule.

The host controller will not react to modifications to the Periodic Schedule Enable immediately. In order to eliminate conflicts with split transactions, the host controller evaluates the Periodic Schedule Enable bit only when FRINDEX[2:0] is zero. System software must not disable the periodic schedule if the schedule contains an active split transaction work item that spans the 000b micro-frame. These work items must be removed from the schedule before the Periodic Schedule Enable bit is written to zero.

The Periodic Schedule Status bit in the USB_USBSTS register indicates status of the periodic schedule. System software enables (or disables) the periodic schedule by writing one (or zero) to the Periodic Schedule Enable bit in the USB_USBCMD register. Software then can poll the Periodic Schedule Status bit to determine when the periodic

schedule has made the desired transition. Software must not modify the Periodic Schedule Enable bit unless the value of the Periodic Schedule Enable bit equals that of the Periodic Schedule Status bit.

The periodic schedule is used to manage all isochronous and interrupt transfer streams. The base of the periodic schedule is the periodic frame list. Software links schedule data structures to the periodic frame list to produce a graph of scheduled data structures. The graph represents an appropriate sequence of transactions.

The following figure illustrates isochronous transfers (using iTDs and siTDs) with a period of one are linked directly to the periodic frame list. Interrupt transfers (are managed with queue heads) and isochronous streams with periods other than one are linked following the period-one iTD/siTDs. Interrupt queue heads are linked into the frame list ordered by poll rate. Longer poll rates are linked first (for example, closest to the periodic frame list), followed by shorter poll rates, with queue heads with a poll rate of one, on the very end.

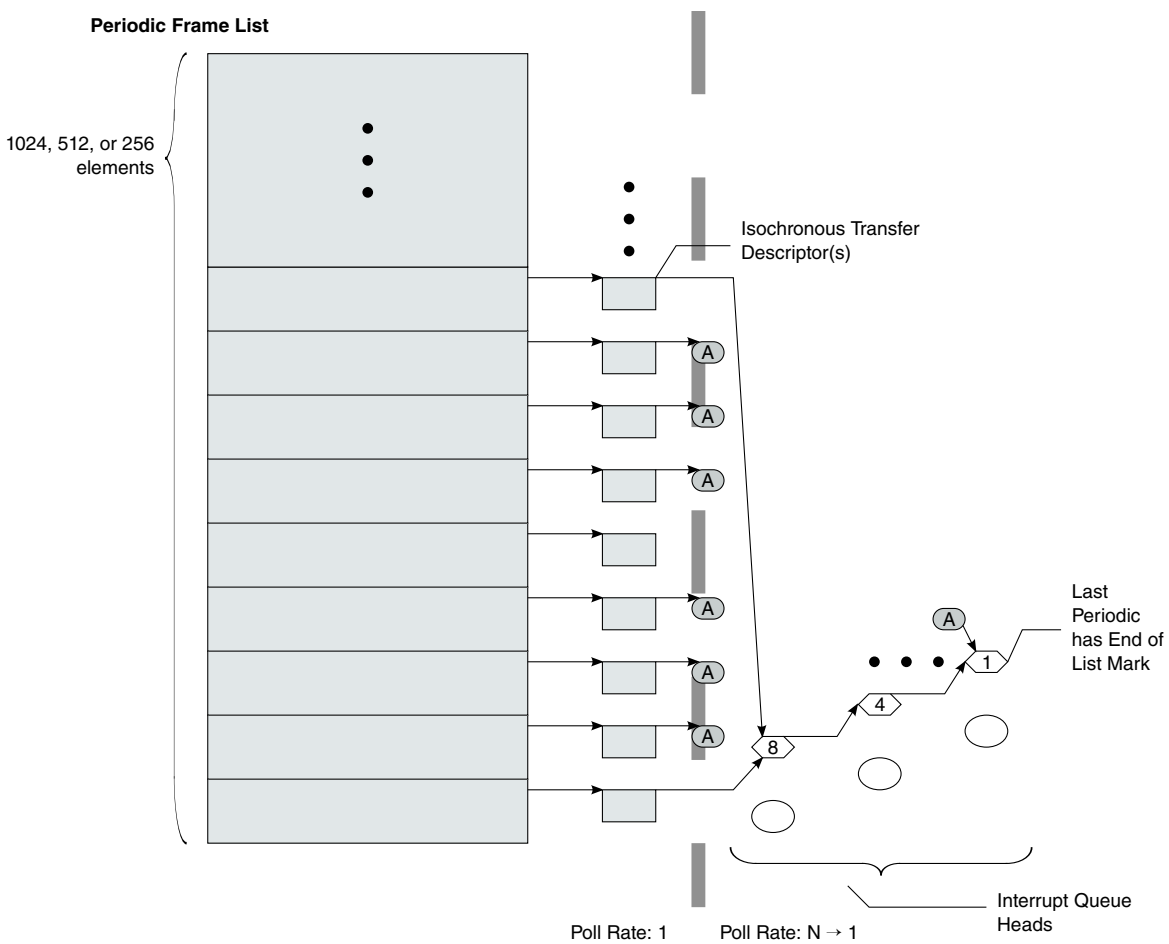


Figure 42-11. Example Periodic Schedule

42.5.3.7 Managing Isochronous Transfers Using iTDs

The structure of an iTD is presented in Isochronous (High-Speed) Transfer Descriptor (iTD). The four distinct sections to an iTD:

- The first field is the Next Link Pointer. This field is for schedule linkage purposes only.
- Transaction description array. This area is an eight-element array. Each element represents control and status information for one micro-frame's worth of transactions for a single high-speed isochronous endpoint.
- The buffer page pointer array is a 7-element array of physical memory pointers to data buffers. These are 4 K aligned pointers to physical memory.
- Endpoint capabilities. This area utilizes the unused low-order 12 bits of the buffer page pointer array. The fields in this area are used across all transactions executed for this iTD, including endpoint addressing, transfer direction, maximum packet size and high-bandwidth multiplier.

42.5.3.7.1 Host Controller Operational Model for iTDs

The host controller uses FRINDEX register bits [12:3] to index into the periodic frame list. This means that the host controller visits each frame list element eight consecutive times before incrementing to the next periodic frame list element. Each iTD contains eight transaction descriptions, which map directly to FRINDEX register bits [2:0]. Therefore, each transaction descriptor corresponds to one micro-frame. Each iTD can span 8 micro-frames worth of transactions.

When the host controller fetches an iTD, it uses FRINDEX register bits [2:0] to index into the transaction description array.

If the active bit in the Status field of the indexed transaction description is set to zero, the host controller ignores the iTD and follows the Next pointer to the next schedule data structure.

When the indexed active bit is one, the host controller continues to parse the iTD. It stores the indexed transaction description and the general endpoint information (device address, endpoint number, maximum packet size, and so on.). It also uses the Page Select (PG) field to index the buffer pointer array, storing the selected buffer pointer and the next sequential buffer pointer. For example, if PG field is 0, then the host controller stores Page 0 and Page 1.

The host controller constructs a physical data buffer address by concatenating the current buffer pointer (as selected using the current transaction description's PG field) and the transaction description's Transaction Offset field. The host controller uses the endpoint addressing information and I/O-bit to execute a transaction to the appropriate endpoint. When the transaction is complete, the host controller clears the active bit and writes back any additional status information to the Status field in the currently selected transaction description.

The data buffer associated with the iTD must be virtually contiguous memory. Seven page pointers are provided to support eight high-bandwidth transactions regardless of the starting packet's offset alignment into the first page. A starting buffer pointer (physical memory address) is constructed by concatenating the page pointer (for example, page 0 pointer) selected by the active transaction descriptions' PG (for example, value: 00B) field with the transaction offset field. As the transaction moves data, the host controller must detect when an increment of the current buffer pointer crosses a page boundary. When this occurs the host controller simply replaces the current buffer pointer's page portion with the next page pointer (for example, page 1 pointer) and continues to move data. The size of each bus transaction is determined by the value in the Maximum Packet Size field. An iTD supports high-bandwidth pipes through the Mult (multiplier) field. When the Mult field is 1, 2, or 3, the host controller executes the specified number of Maximum Packet sized bus transactions for the endpoint in the current micro-frame. In other words, the Mult field represents a transaction count for the endpoint in the current micro-frame. If the Mult field is zero, the operation of the host controller is undefined. The transfer description is used to service all transactions indicated by the Mult field.

For OUT transfers, the value of the Transaction X Length field represents the total bytes to be sent during the micro-frame. The Mult field must be set by software to be consistent with Transaction X Length and Maximum Packet Sixe. The host controller sends the bytes in Maximum Packet Size'd portions. After each transaction, the host controller decrements its local copy of Transaction X Length by Maximum Packet Size. The number of bytes the host controller sends is always Maximum Packet Size or Transaction X Length, whichever is less. The host controller advances the transfer state in the transfer description, updates the appropriate record in the iTD and moves to the next schedule data structure. The maximum sized transaction supported is 3 x 1024 bytes.

For IN transfers, the host controller issues Mult transactions. It is assumed that software has properly initialized the iTD to accommodate all of the possible data. During each IN transaction, the host controller must use Maximum Packet Size to detect packet babble errors. The host controller keeps the sum of bytes received in the Transaction X Length field. After all transactions for the endpoint have completed for the micro-frame, Transaction X Length contains the total bytes received. If the final value of Transaction X Length is less than the value of Maximum Packet Size, then less data than was allowed for was received from the associated endpoint. This short packet condition does not set

the USBINT bit in the USB_USBSTS register to one. The host controller will not detect this condition. If the device sends more than Transaction X Length or Maximum Packet Size bytes (whichever is less), then the host controller sets the Babble Detected bit to one and set the Active bit to zero. Note, that the host controller is not required to update the iTD field Transaction X Length in this error scenario. If the Mult field is greater than one, then the host controller automatically executes the value of Mult transactions. The host controller will not execute all Mult transactions if:

- The endpoint is an OUT and Transaction X Length goes to zero before all the Mult transactions have executed (ran out of data), or
- The endpoint is an IN and the endpoint delivers a short packet, or an error occurs on a transaction before Mult transactions have been executed. The end of micro-frame may occur before all of the transaction opportunities have been executed. When this happens, the transfer state of the transfer description is advanced to reflect the progress that was made, the result written back to the iTD and the host controller proceeds to processing the next micro-frame. Refer to Appendix D for a table summary of the host controller required behavior for all the high-bandwidth transaction cases.

42.5.3.7.2 Software Operational Model for iTDs

A client buffer request to an isochronous endpoint may span 1 to N micro-frames. When N is larger than one, system software may have to use multiple iTDs to read or write data with the buffer (if N is larger than eight, it must use more than one iTD).

The following figure illustrates the simple model of how a client buffer is mapped by system software to the periodic schedule (that is the periodic frame list and a set of iTDs). On the right is the client description of its request. The description includes a buffer base address plus additional annotations to identify which portions of the buffer should be used with each bus transaction. In the middle is the iTD data structures used by the system software to service the client request. Each iTD can be initialized to service up to 24 transactions, organized into eight groups of up to three transactions each. Each group maps to one micro-frame's worth of transactions. The EHCI controller does not provide per-transaction results within a micro-frame. It treats the per-micro-frame transactions as a single logical transfer. On the left is the host controller's frame list. System software establishes references from the appropriate locations in the frame list to each of the appropriate iTDs. If the buffer is large, then system software can use a small set of iTDs to service the entire buffer. System software can activate the transaction description records (contained in each iTD) in any pattern required for the particular data stream.

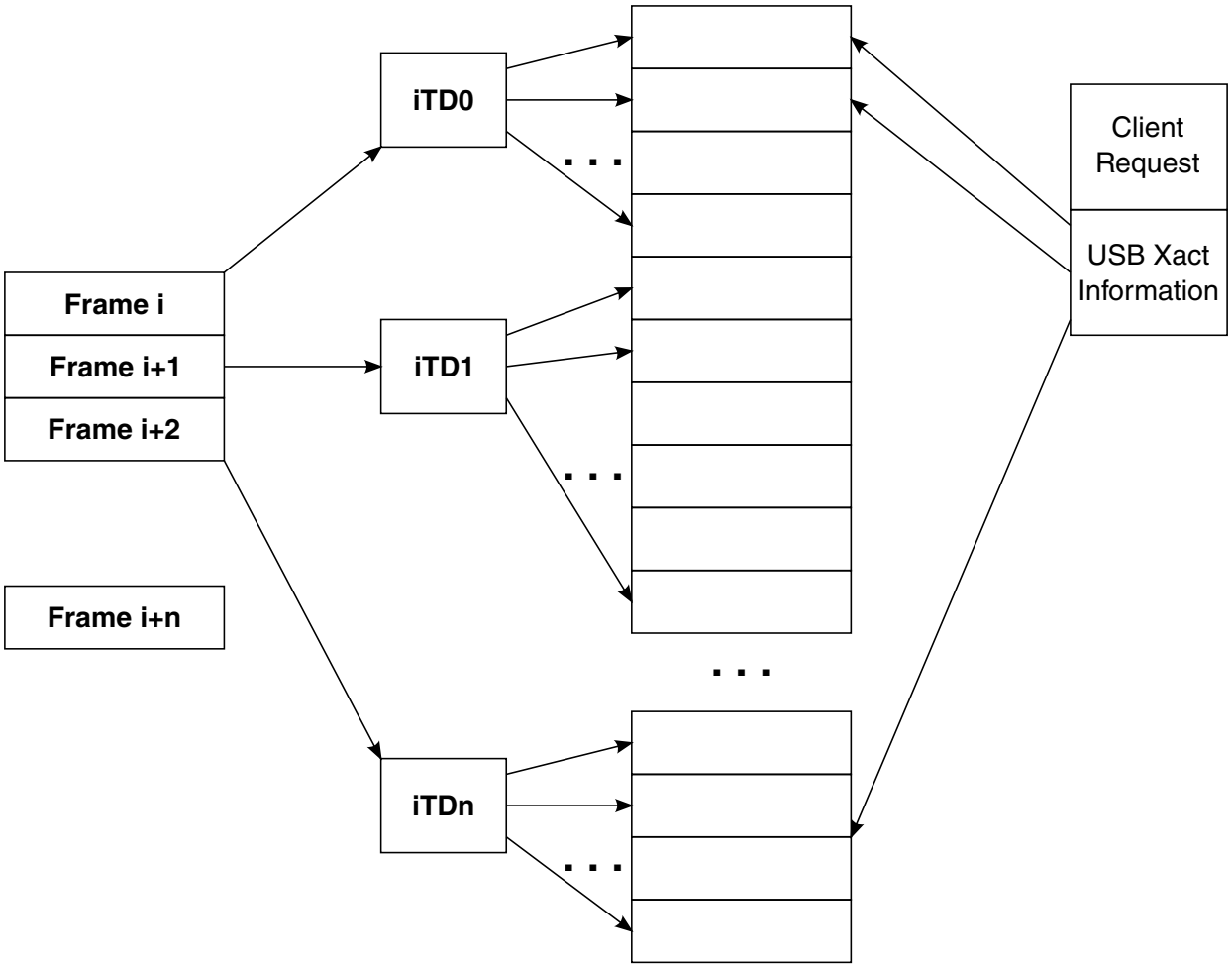


Figure 42-12. Example Association of iTDs to Client Request Buffer

As noted above, the client request includes a pointer to the base of the buffer and offsets into the buffer to annotate which buffer sections are to be used on each bus transaction that occurs on this endpoint. System software must initialize each transaction description in an iTD to ensure it uses the correct portion of the client buffer. For example, for each transaction description, the PG field is set to index the correct physical buffer page pointer and the Transaction Offset field is set relative to the correct buffer pointer page (for example, the same one referenced by the PG field). When the host controller executes a transaction it selects a transaction description record based on FRINDEX[2:0]. It then uses the current Page Buffer Pointer (as selected by the PG field) and concatenates to the transaction offset field. The result is a starting buffer address for the transaction. As the host controller moves data for the transaction, it must watch for a page wrap condition and properly advance to the next available Page Buffer Pointer. System software must not use the Page 6 buffer pointer in a transaction description where the length of the transfer wraps a page boundary. Doing so yields undefined behavior. The host controller hardware is not required to 'alias' the page selector to Page zero. USB 2.0 isochronous

endpoints can specify a period greater than one. Software can achieve the appropriate scheduling by linking iTDs into the appropriate frames (relative to the frame list) and by setting appropriate transaction description elements active bits to one.

42.5.3.7.2.1 Periodic scheduling threshold

The Isochronous Scheduling Threshold field in the USB_HCCPARAMS capability register is an indicator to system software as to how the host controller pre-fetches and effectively caches schedule data structures.

It is used by system software when adding isochronous work items to the periodic schedule. The value of this field indicates to system software the minimum distance it can update isochronous data (relative to the current location of the host controller execution in the periodic list) and still have the host controller process them.

The iTD and siTD data structures each describe 8 micro-frames worth of transactions. The host controller is allowed to cache one (or more) of these data structures in order to reduce memory traffic. Three basic caching models that account for the fact the isochronous data structures span 8 micro-frames. The three caching models are: no caching, micro-frame caching and frame caching.

When software is adding new isochronous transactions to the schedule, it always performs a read of the USB_FRINDEX register to determine the current frame and micro-frame the host controller is currently executing. Of course, there is no information about where in the micro-frame the host controller is, so a constant uncertainty-factor of one micro-frame has to be assumed. Combining the knowledge of where the host controller is executing with the knowledge of the caching model allows the definition of simple algorithms for how closely software can reliably work to the executing host controller.

No caching is indicated with a value of zero in the Isochronous Scheduling Threshold field. The host controller may pre-fetch data structures during a periodic schedule traversal (per micro-frame) but always dumps any accumulated schedule state at the end of the micro-frame. At the appropriate time relative to the beginning of every micro-frame, the host controller always begins schedule traversal from the frame list. Software can use the value of the USB_FRINDEX register (plus the constant 1 uncertainty-factor) to determine the approximate position of the executing host controller. When no caching is selected, software can add an isochronous transaction as near as 2 micro-frames in front of the current executing position of the host controller.

Frame caching is indicated with a non-zero value in bit [7] of the Isochronous Scheduling Threshold field. In the frame-caching model, system software assumes that the host controller caches one (or more) isochronous data structures for an entire frame (8 micro-frames). Software uses the value of the USB_FRINDEX register (plus the constant 1

uncertainty) to determine the current micro-frame/frame (assume modulo 8 arithmetic in adding the constant 1 to the micro-frame number). For any current frame N, if the current micro-frame is 0 to 6, then software can safely add isochronous transactions to Frame N + 1. If the current micro-frame is 7, then software can add isochronous transactions to Frame N + 2.

Micro-frame caching is indicated with a non-zero value in the least-significant 3 bits of the Isochronous Scheduling Threshold field. System software assumes the host controller caches one or more periodic data structures for the number of micro-frames indicated in the Isochronous Scheduling Threshold field. For example, if the count value were 2, then the host controller keeps a window of 2 micro-frames worth of state (current micro-frame, plus the next) on-chip. On each micro-frame boundary, the host controller releases the current micro-frame state and begins accumulating the next micro-frame state.

42.5.3.8 Asynchronous Schedule

The Asynchronous schedule traversal is enabled or disabled through the Asynchronous Schedule Enable bit in the USB_USBCMD register.

If the Asynchronous Schedule Enable bit is set to zero, then the host controller simply does not try to access the asynchronous schedule through the USB_ASYNCLISTADDR register. Likewise, when the Asynchronous Schedule Enable bit is one, then the host controller does use the USB_ASYNCLISTADDR register to traverse the asynchronous schedule. Modifications to the Asynchronous Schedule Enable bit are not necessarily immediate. Rather the new value of the bit is taken into consideration the next time the host controller needs to use the value of the USB_ASYNCLISTADDR register to get the next queue head.

The Asynchronous Schedule Status bit in the USB_USBSTS register indicates status of the asynchronous schedule. System software enables (or disables) the asynchronous schedule by writing one (or zero) to the Asynchronous Schedule Enable bit in the USB_USBCMD register. Software then can poll the Asynchronous Schedule Status bit to determine when the asynchronous schedule has made the desired transition. Software must not modify the Asynchronous Schedule Enable bit unless the value of the Asynchronous Schedule Enable bit equals that of the Asynchronous Schedule Status bit.

The asynchronous schedule is used to manage all Control and Bulk transfers. Control and Bulk transfers are managed using queue head data structures. The asynchronous schedule is based at the USB_ASYNCLISTADDR register. The default value of the USB_ASYNCLISTADDR register after reset is undefined and the schedule is disabled when the Asynchronous Schedule Enable bit is zero.

Software may only write this register with defined results when the schedule is disabled. For example, Asynchronous Schedule Enable bit in the USB_USBCMD and the Asynchronous Schedule Status bit in the USB_USBSTS register are zero. System software enables execution from the asynchronous schedule by writing a valid memory address (of a queue head) into this register. Then software enables the asynchronous schedule by setting the Asynchronous Schedule Enable bit to one. The asynchronous schedule is actually enabled when the Asynchronous Schedule Status bit is one.

When the host controller begins servicing the asynchronous schedule, it begins by using the value of the USB_ASYNCLISTADDR register. It reads the first referenced data structure and begins executing transactions and traversing the linked list as appropriate. When the host controller completes processing the asynchronous schedule, it retains the value of the last accessed queue head's horizontal pointer in the USB_ASYNCLISTADDR register. Next time the asynchronous schedule is accessed, this is the first data structure that is serviced. This provides round-robin fairness for processing the asynchronous schedule.

A host controller completes processing the asynchronous schedule when one of the following events occur:

- The end of a micro-frame occurs.
- The host controller detects an empty list condition (see Empty Asynchronous Schedule Detection)
- The schedule has been disabled through the Asynchronous Schedule Enable bit in the USB_USBCMD register.

The queue heads in the asynchronous list are linked into a simple circular list as shown in Figure 42-7. Queue head data structures are the only valid data structures that may be linked into the asynchronous schedule. An isochronous transfer descriptor (iTD or siTD) in the asynchronous schedule yields undefined results.

The maximum packet size field in a queue head is sized to accommodate the use of this data structure for all non-isochronous transfer types. The USB Specification, Revision 2.0 specifies the maximum packet sizes for all transfer types and transfer speeds. System software should always parameterize the queue head data structures according to the core specification requirements.

42.5.3.8.1 Adding Queue Heads to Asynchronous Schedule

This is a software requirement section.

There are two independent events for adding queue heads to the asynchronous schedule. The first is the initial activation of the asynchronous list. The second is inserting a new queue head into an activated asynchronous list.

Activation of the list is simple. System software writes the physical memory address of a queue head into the USB_ASYNCLISTADDR register, then enables the list by setting the Asynchronous Schedule Enable bit in the USB_USBCMD register to one.

When inserting a queue head into an active list, software must ensure that the schedule is always coherent from the host controllers' point of view. This means that the system software must ensure that all queue head pointer fields are valid. For example, qTD pointers have T-Bits set to one or reference valid qTDs and the Horizontal Pointer references a valid queue head data structure. The following algorithm represents the functional requirements:

```
InsertQueueHead (pQHeadCurrent, pQueueHeadNew)
          --
          -- Requirement: all inputs must be properly initialized.
          --
          -- pQHeadCurrent is a pointer to a queue head that is
          -- already in the active list
          -- pQHeadNew is a pointer to the queue head to be added
          --
          -- This algorithm links a new queue head into a existing
          -- list
          --
          pQueueHeadNew.HorizontalPointer  =  pQueueHeadCurrent.HorizontalPointer
          pQueueHeadCurrent.HorizontalPointer =  physicalAddressOf(pQueueHeadNew)
End InsertQueueHead
```

42.5.3.8.2 Removing Queue Heads from Asynchronous Schedule

This is a software requirement section.

There are two independent events for removing queue heads from the asynchronous schedule. The first is shutting down (deactivating) the asynchronous list. The second is extracting a single queue head from an activated list.

Software deactivates the asynchronous schedule by setting the Asynchronous Schedule Enable bit in the USB_USBCMD register to zero. Software can determine when the list is idle when the Asynchronous Schedule Status bit in the USB_USBSTS register is zero. The normal mode of operation is that software removes queue heads from the asynchronous schedule without shutting it down. Software must not remove an active queue head from the schedule. Software should first deactivate all active qTDs, wait for the queue head to go inactive, then remove the queue head from the asynchronous list. Software removes a queue head from the asynchronous list through the following algorithm. As illustrated, the unlinking is quite easy. Software merely must ensure all of the link pointers reachable by the host controller are kept consistent.

```
UnlinkQueueHead (pQHeadPrevious, pQueueHeadToUnlink, pQHeadNext)
          --
          -- Requirement: all inputs must be properly initialized.
          --
          -- pQHeadPrevious is a pointer to a queue head that
          -- references the queue head to remove
          -- pQHeadToUnlink is a pointer to the queue head to be
```

```
          -- removed
          -- pQheadNext is a pointer to a queue head still in the
          -- schedule. Software provides this pointer with the
          -- following strict rules:
          --          if the host software is one queue head, then
          --          pQHeadNext must be the same as
          --          QueueheadToUnlink.HorizontalPointer. If the host
          --          software is unlinking a consecutive series of
          --          queue heads, QHeadNext must be set by software to
          --          the  queue head remaining in the schedule.
          --
          -- This algorithm unlinks a queue head from a circular list
          --
          pQueueHeadPrevious.HorizontalPointer =  pQueueHeadToUnlink.HorizontalPointer
                                  pQueueHeadToUnlink.HorizontalPointer = pQHeadNext
End UnlinkQueueHead
```

If software removes the queue head with the H-bit set to one, it must select another queue head still linked into the schedule and set its H-bit to one. This should be completed before removing the queue head. The requirement is that software keep one queue head in the asynchronous schedule, with its H-bit set to one. At the point software has removed one or more queue heads from the asynchronous schedule, it is unknown whether the host controller has a cached pointer to them. Similarly, it is unknown how long the host controller might retain the cached information, as it is implementation dependent and may be affected by the actual dynamics of the schedule load. Therefore, once software has removed a queue head from the asynchronous list, it must retain the coherency of the queue head (link pointers, and so on). It cannot disturb the removed queue heads until it knows that the host controller does not have a local copy of a pointer to any of the removed data structures.

The method software uses to determine when it is safe to modify a removed queue head is to handshake with the host controller. The handshake mechanism allows software to remove items from the asynchronous schedule, then execute a simple, lightweight handshake that is used by software as a key that it can free (or reuse) the memory associated the data structures it has removed from the asynchronous schedule.

The handshake is implemented with three bits in the host controller. The first bit is a command bit (Interrupt on Async Advance Doorbell bit in the USB_USBCMD register) that allows software to inform the host controller that something has been removed from its asynchronous schedule. The second bit is a status bit (Interrupt on Async Advance bit in the USB_USBSTS register) that the host controller sets after it has released all on-chip state that may potentially reference one of the data structures just removed. When the host controller sets this status bit to one, it also sets the command bit to zero. The third bit is an interrupt enable (Interrupt on Async Advance bit in the USB_USBINTR register) that is matched with the status bit. If the status bit is one and the interrupt enable bit is one, then the host controller asserts a hardware interrupt.

The figure below illustrates a general example. In this example, consecutive queue heads (B and C) are unlinked from the schedule using the algorithm above. Before the unlink operation, the host controller has a copy of queue head A.

The unlink algorithm requires that as software unlinks each queue head, the unlinked queue head is loaded with the address of a queue head that remains in the asynchronous schedule.

When the host controller observes that doorbell bit being set to one, it makes a note of the local reachable schedule information. In this example, the local reachable schedule information includes both queue heads (A and B). It is sufficient that the host controller can set the status bit (and clear the doorbell bit) as soon as it has traversed beyond current reachable schedule information (that is traversed beyond queue head (B) in this example). The following figure illustrates the generic queue head unlink scenario.

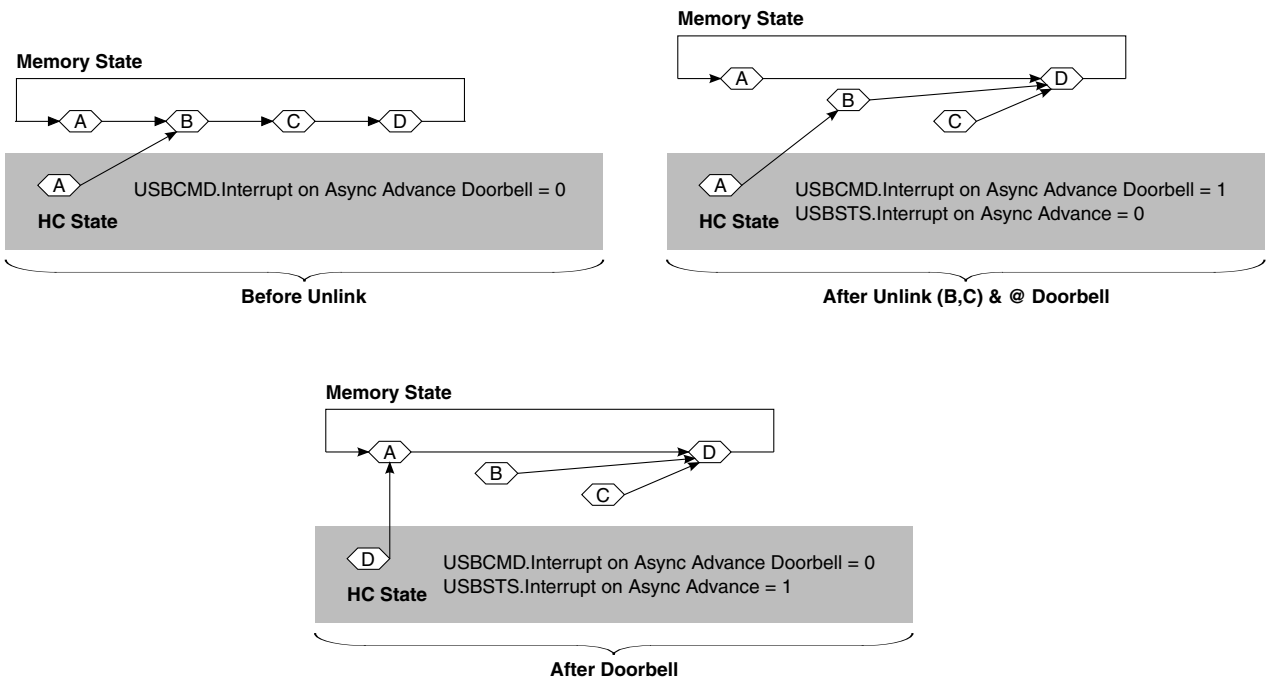


Figure 42-13. Generic Queue Head Unlink Scenario

Alternatively, a host controller implementation is allowed to traverse the entire asynchronous schedule list (for example, observed the head of the queue (twice)) before setting the Advance on Async status bit to one.

Software may re-use the memory associated with the removed queue heads after it observes the Interrupt on Async Advance status bit is set to one, following assertion of the doorbell. Software should acknowledge the Interrupt on Async Advance status as indicated in the USB_USBSTS register, before using the doorbell handshake again.

42.5.3.8.3 Empty Asynchronous Schedule Detection

The Enhanced Host Controller Interface uses two bits to detect when the asynchronous schedule is empty.

The queue head data structure (see Table 42-26) defines an *H-bit* in the queue head, which allows software to mark a queue head as being the *head* of the reclaim list. The Enhanced Host Controller Interface also keeps a 1-bit flag in the USB_USBSTS register (*Reclamation*) that is set to zero when the Enhanced Interface Host Controller observes a queue head with the H-bit set to one. The reclamation flag in the status register is set to one when any USB transaction from the asynchronous schedule is executed (or whenever the asynchronous schedule starts, see Asynchronous schedule traversal: *Start Event*).

If the Enhanced Host Controller Interface ever encounters an *H-bit* of one and a *Reclamation* bit of zero, the EHCI controller simply stops traversal of the asynchronous schedule.

An example illustrating the H-bit in a schedule is shown in the following figure.

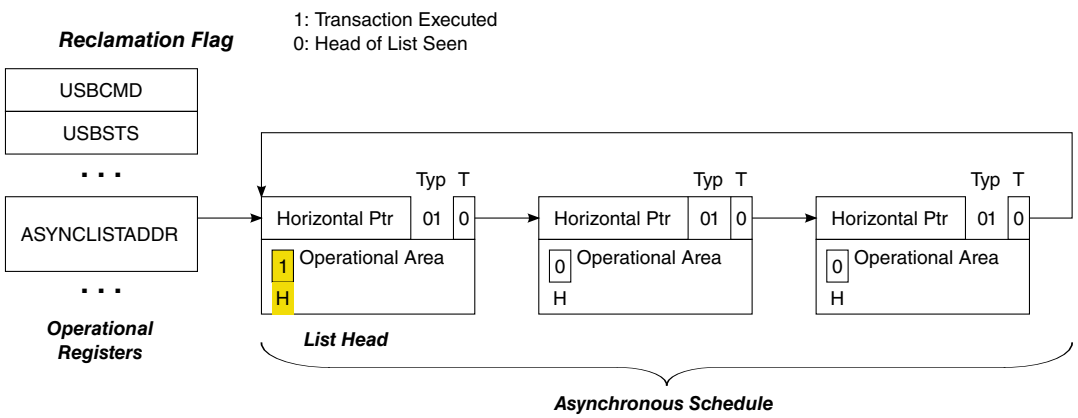


Figure 42-14. Asynchronous Schedule List w/Annotation to Mark Head of List

Software must ensure there is at most one queue head with the *H-bit* set to one, and that it is always coherent with respect to the schedule.

42.5.3.8.4 Restarting Asynchronous Schedule Before EOF

There are many situations where the host controller will detect an empty list *long* before the end of the micro-frame.

It is important to remember that under many circumstances the schedule traversal has stopped due to Nak/Nyet responses from all endpoints.

An example of particular interest is when a start-split for a bulk endpoint occurs early in the micro-frame. Given the EHCI simple traversal rules, the complete-split for that transaction may Nak/Nyet out very quickly. If it is the only item in the schedule, then the host controller ceases traversal of the Asynchronous schedule very early in the micro-frame. In order to provide reasonable service to this endpoint, the host controller should issue the complete-split before the end of the current micro-frame, instead of waiting

until the next micro-frame. When the reason for host controller idling asynchronous schedule traversal is because of empty list detection, it is mandatory the host controller implement a 'waking' method to resume traversal of the asynchronous schedule. An example method is described below.

42.5.3.8.4.1 Example Method for Restarting Asynchronous Schedule Traversal

The reason for idling the host controller when the list is empty is to keep the host controller from unnecessarily occupying too much memory bandwidth. The question is: *how long should the host controller stay idle before restarting*?

The answer in this example is based on deriving a manifest constant, which is the amount of time the host controller will stay idle before restarting traversal. In this example, the manifest constant is called *AsyncSchedSleepTime*, and has a value of 10 µsec. The value is derived based on the analysis in Example Derivation for *AsyncSchedSleepTime*, The traversal algorithm is simple:

- Traverse the Asynchronous schedule until the either an End-Of-micro-Frame event occurs, or an empty list is detected. If the event is an End-of-micro-Frame, go attempt to traverse the Periodic schedule. If the event is an empty list, then set a sleep timer and go to a *schedule sleep* state.
- When the sleep timer expires, set working context to the Asynchronous Schedule start condition and go to *schedule active* state. The start context allows the HC to reload *Nakcnt* fields, and so on. So the HC has a chance to run for more than one iteration through the schedule.

This process simply repeats itself each micro-frame. The figure below illustrates a sample state machine to manage the active and sleep states of the Asynchronous Schedule traversal policy. There are three states: Actively traversing the Asynchronous schedule, Sleeping, and Not Active. The last two are similar in terms of interaction with the Asynchronous schedule, but the Not Active state means that the host controller is busy with the Periodic schedule or the Asynchronous schedule is not enabled. The Sleeping state is specifically a special state where the host controller is just waiting for a period of time before resuming execution of the Asynchronous schedule.

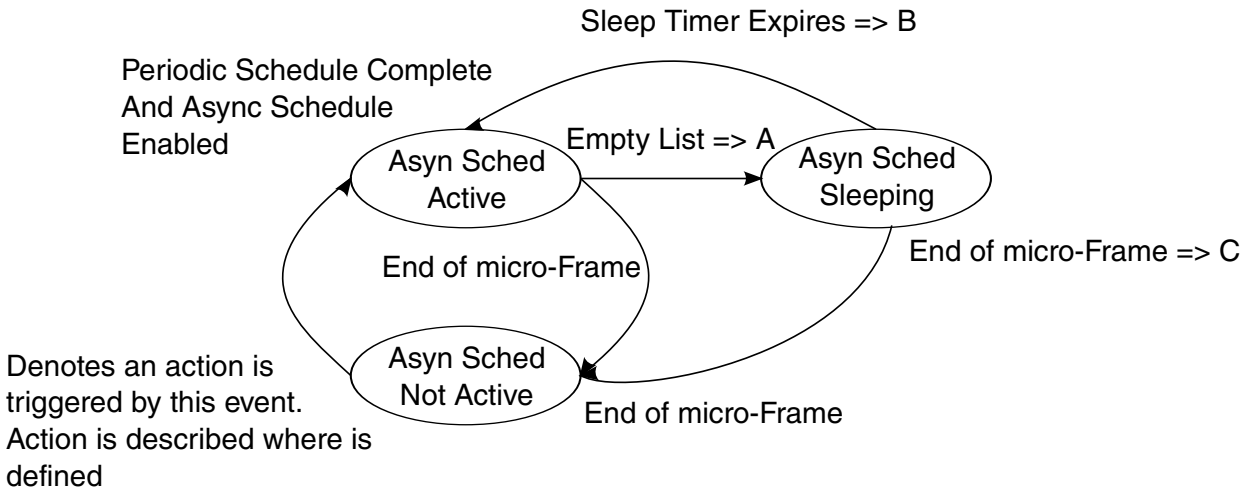


Figure 42-15. Example State Machine for Managing Asynchronous Schedule Traversal

The actions referred to in the figure above are defined in the following table.

Table 42-41. Asynchronous Schedule SM Transition Actions

Action	Action Description Label
A	On detection of the empty list, the host controller sets the *AsynchronousTraversalSleepTimer* to *AsyncSchedSleepTime*.
B	When the *AsynchronousTraversalSleepTimer* expires, the host controller sets the *Reclamation* bit in the USBSTS register to one and moves the Nak Counter reload state machine to WaitForListHead (see Nak Count Reload Control).
C	The host controller cancels the sleep timer (*AsynchronousTraversalSleepTimer*).

42.5.3.8.4.2 Async Sched Not Active

This is the initial state of the traversal state machine after a host controller reset. The traversal state machine does not leave this state when the *Asynchronous Schedule Enable* bit in the USB_USBCMD register is zero.

This state is entered from Async Sched Active or Async Sched Sleeping states when the end-of-micro-frame event is detected.

42.5.3.8.4.3 Async Sched Active

This state is entered from the Async Sched Not Active state when the periodic schedule is not active. It is also entered from the Async Sched Sleeping states when the *AsyncrhonousTraversalSleepTimer* expires. On every transition into this state, the host controller sets the *Reclamation* bit in the USB_USBSTS register to one.

While in this state, the host controller continually traverses the asynchronous schedule until either the end of micro-frame or an empty list condition is detected.

42.5.3.8.4.4 Async Sched Sleeping

The state is entered from the Async Sched Active state when a schedule empty condition is detected. On entry to this state, the host controller sets the *AsynchronousTraversalSleepTimer* to *AsyncSchedSleepTime*.

42.5.3.8.4.5 Example Derivation for *AsyncSchedSleepTime*

The derivation is based on analysis of what work the host controller could be doing next.

It assumes the host controller does not keep any state about what work is possibly pending in the asynchronous schedule. The schedule could contain any mix of the possible combinations of high- full- or low-speed control and bulk requests.

The table below summarizes some of the typical 'next transactions' that could be in the schedule, and the amount of time (for example *footprint*, or *wall clock*) the transaction takes to complete.

Table 42-42. Typical Low-/Full-speed Transaction Times

Transaction Attributes		Footprint (time)	Description
Speed	HS	11.9 ms	Maximum foot print for a worst-case, full-sized bulk data transaction.
Size	512	9.45 ms	Maximum footprint for an approximate best-case, full-sized bulk data transaction.
Type	Bulk		
Speed	FS	~50 ms	Approximate typical for full-sized bulk data. An 8-byte low-speed is about 2x, or between 90 and 100 ms.
Size	64		
Type	Bulk		
Speed	FS	~12 ms	Approximate typical for 8-byte bulk/control (that is setup)
Size	8		
Type	Cntrl		

A *AsyncSchedSleepTime* value of 10 µs provides a reasonable relaxation of the system memory load and still provides a good level of service for the various transfer types and payload sizes. For example, say we detect an empty list after issuing a start-split for a 64-byte full-speed bulk request. Assuming this is the only thing in the list, the host controller gets the results of the full-speed transaction from the hub during the fifth complete-split request. If the full-speed transaction was an IN and it nak'd, the 10 µs sleep period would allow the host controller to get the NAK results on the first complete-split.

42.5.3.8.5 Asynchronous schedule traversal: *Start Event*

Once the HC has *idled* itself through the empty schedule detection (Section 0), it will naturally *activate* and begin processing from the Periodic Schedule at the beginning of each micro-frame.

In addition, it may have idled itself early in a micro-frame. When this occurs (idles early in the micro-frame) the HC must occasionally *re-activate* during the micro-frame and traverse the asynchronous schedule to determine whether any progress can be made. The requirements and method for this restart are described in Restarting Asynchronous Schedule Before EOF . Asynchronous schedule *Start Event*s are defined to be:

- Whenever the host controller transitions from the periodic schedule to the asynchronous schedule. If the periodic schedule is disabled and the asynchronous schedule is enabled, then the beginning of the micro-frame is equivalent to the transition from the periodic schedule, or
- The asynchronous schedule traversal restarts from a sleeping state (see Restarting Asynchronous Schedule Before EOF).

42.5.3.8.6 Reclamation Status Bit (USBSTS Register)

The operation of the empty asynchronous schedule detection feature (see Empty Asynchronous Schedule Detection) depends on the proper management of the *Reclamation* bit in the USB_USBSTS register. The host controller tests for an empty schedule just after it fetches a new queue head while traversing the asynchronous schedule (see Fetch Queue Head).

The host controller is required to set the *Reclamation* bit to one whenever an asynchronous schedule traversal *Start Event*, as documented in Asynchronous schedule traversal: *Start Event*, occurs. The *Reclamation* bit is also set to one whenever the host controller executes a transaction while traversing the asynchronous schedule (see Execute Transaction). The host controller sets the *Reclamation* bit to zero whenever it finds a queue head with its *H-bit* set to one. Software should only set a queue head's *H-bit* if the queue head is in the asynchronous schedule. If software sets the *H-bit* in an interrupt queue head to one, the resulting behavior is undefined. The host controller may set the *Reclamation* bit to zero when executing from the periodic schedule.

42.5.3.9 Operational Model for Nak Counter

This section describes the operational model for the *NakCnt* field defined in a queue head.

See Queue Head Initialization for more information. Software should not use this feature for interrupt queue heads. This rule is not required to be enforced by the host controller.

USB protocol has built-in flow control through the Nak response by a device. There are several scenarios, beyond the Ping feature, where an endpoint may naturally Nak or Nyet the majority of the time. An example is the host controller management of the split transaction protocol for control and bulk endpoints. All bulk endpoints (High- or Full-speed) are serviced through the same asynchronous schedule. The time between the *Start-split* transaction and the first *Complete-split* transaction could be very short (that is like when the endpoint is the only one in the asynchronous schedule). The hub NYETs (effectively Naks) the *Complete-split* transaction until the classic transaction is complete. This could result in the host controller thrashing memory, repeatedly fetching the queue head and executing the transaction to the Hub, which does not complete until after the transaction on the classic bus completes.

The two component fields in a queue head to support the throttling feature: a counter field (*NakCnt*), and a counter reload field (*RL*). *NakCnt* is used by the host controller as one of the criteria to determine whether or not to execute a transaction to the endpoint. The two operational modes associated with this counter:

- Not Used- This mode is set when the *RL* field is zero. The host controller ignores the *NakCnt* field for any execution of transactions through a queue head with an *RL* field of zero. Software must use this selection for interrupt endpoints.
- Nak Throttle Mode- This mode is selected when the *RL* field is non-zero. In this mode, the value in the *NakCnt* field represents the maximum number of Nak or Nyet responses the host controller tolerates on each endpoint. In this mode, the HC decrements the *NakCnt* field based on the token/handshake criteria listed in the table below. The host controller must reload *NakCnt* when the endpoint successfully moves data (for example, policy to reward device for moving data).

The following table describes the NakCnt field adjustment rules.

Table 42-43. NakCnt Field Adjustment Rules

Token	Handshake	
	Handshake NAK	NYET
IN/PING	decrement *NakCnt*	N/A (protocol error)
OUT	decrement *NakCnt*	No Action[1] Start
Split	decrement *NakCnt*	N/A (protocol error)
Complete Split	No Action	Decrement *NakCnt*

1. Recommended behavior on this response is to reload *NakCnt*

In summary, system software enables the counter by setting the reload field (*RL*) to a non-zero value. The host controller may execute a transaction if *NakCnt* is non-zero. The host controller does not execute a transaction if *NakCnt* is zero. The reload mechanism is described in detail in Nak Count Reload Control .

NOTE

When all queue heads in the Asynchronous Schedule either exhausts all transfers or all NakCnt's go to zero, then the host controller detects an empty Asynchronous Schedule and idle schedule traversal (see Empty Asynchronous Schedule Detection).

Any time the host controller begins a new traversal of the Asynchronous Schedule, a *Start Event* is assumed, see Asynchronous schedule traversal: *Start Event*. Every time a Start-Event occurs, the Nak Count reload procedure is enabled.

42.5.3.9.1 Nak Count Reload Control

When the host controller reaches the *Execute Transaction* state for a queue head (meaning that it has an active operational state), it checks to determine whether the *NakCnt* field should be reloaded from *RL* (see Execute Transaction). If the answer is yes, then *RL* is copied into *NakCnt*. After the reload or if the reload is not active, the host controller evaluates whether to execute the transaction.

The host controller must reload nak counters (*NakCnt* see Table 42-26) in queue heads during the first pass through the reclamation list after an asynchronous schedule Start Event (see Asynchronous schedule traversal: *Start Event* for the definition of the Start Event). The Asynchronous Schedule should have at most one queue head marked as the head (see Figure 42-14).

The following figure illustrates an example state machine that satisfies the operational requirements of the host controller detecting the first pass through the Asynchronous Schedule. This state machine is maintained internal to the host controller and is only used to gate reloading of the nak counter during the queue head traversal state: Execute Transaction (see the figure below). The host controller does not perform the nak counter reload operation if the RL field (see Table 42-26) is set to zero.

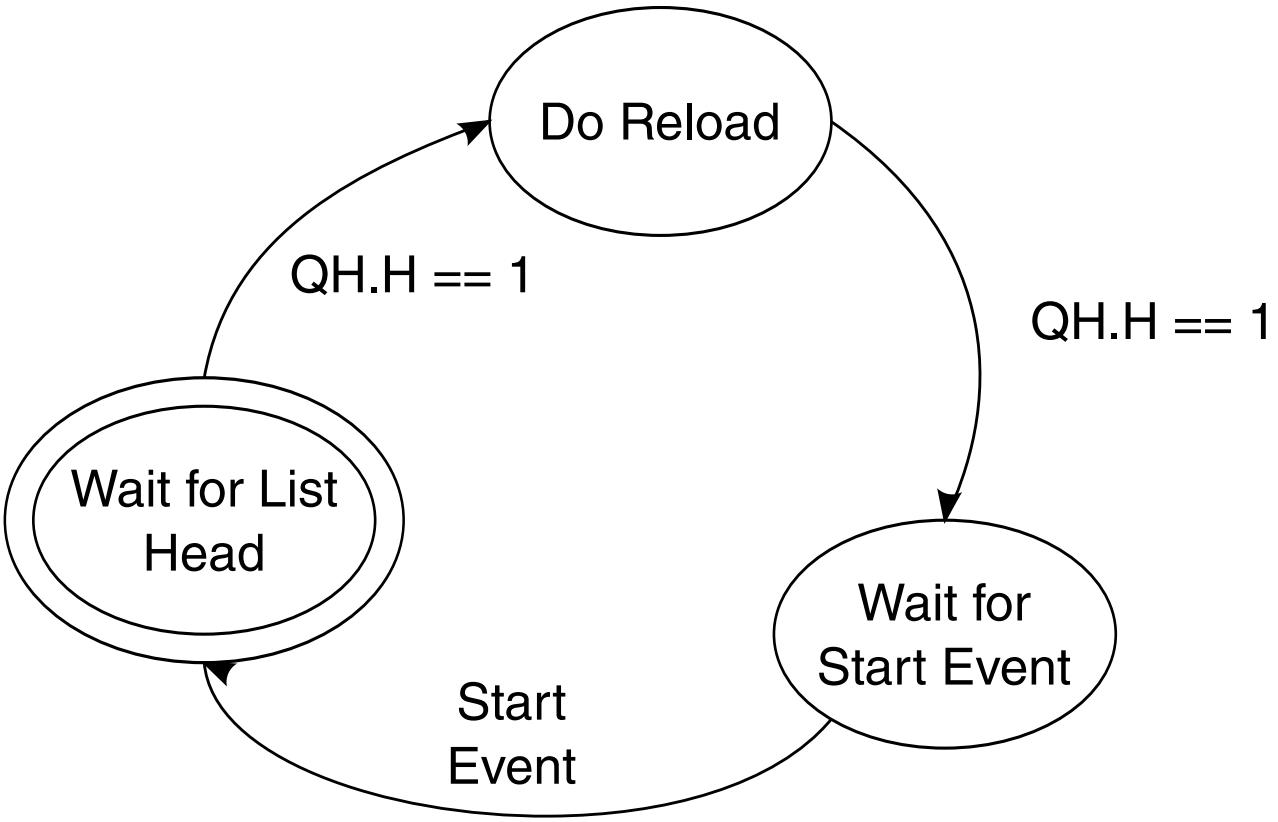


Figure 42-16. Example HC State Machine for Controlling Nak Counter Reloads

42.5.3.9.1.1 Wait for List Head

This is the initial state.

The state machine enters this state from Wait for Start Event when a start event as defined in Asynchronous schedule traversal: *Start Event* occurs.

The purpose of this state is to wait for the first observation of the head of the Asynchronous Schedule.

This occurs when the host controller fetches a queue head whose *H-bit* is set to one.

42.5.3.9.1.2 Do Reload

This state is entered from the Wait for List Head state when the host controller fetches a queue head with the *H-bit* set to one. While in this state, the host controller performs nak counter reloads for every queue head visited that has a non-zero nak reload value (*RL*) field.

42.5.3.9.1.3 Wait for Start Event

This state is entered from the *Do Reload* state when a queue head with the *H-bit* set to one is fetched. While in this state, the host controller does not perform nak counter reloads.

42.5.3.10 Managing Control/Bulk/Interrupt Transfers through Queue Heads

This section presents an overview of how the host controller interacts with queuing data structures.

Queue heads use the Queue Element Transfer Descriptor (qTD) structure. One queue head is used to manage the data stream for one endpoint. The queue head structure contains static endpoint characteristics and capabilities. It also contains a working area from where individual bus transactions for an endpoint are executed (see Overlay area defined in Table 42-26). Each qTD represents one or more bus transactions, which is defined in the context of this specification as a *transfer*.

The general processing model for the host controller's use of a queue head is simple:

- read a queue head,
- execute a transaction from the overlay area,
- write back the results of the transaction to the overlay area,
- move to the next queue head.

If the host controller encounters errors during a transaction, the host controller sets one (or more) of the error reporting bits in the queue head's *Status* field. The *Status* field accumulates all errors encountered during the execution of a qTD (for example, the error bits in the queue head *Status* field are 'sticky' until the transfer (qTD) has completed). This state is always written back to the source qTD when the transfer is complete. On transfer (for example, buffer or halt conditions) boundaries, the host controller must auto-advance (without software intervention) to the next qTD. Additionally, the hardware must be able to halt the queue so no additional bus transactions occurs for the endpoint and the host controller does not advance the queue.

An example host controller operational state machine of a queue head traversal is illustrated in the following figure. This state machine is a model for how a host controller should traverse a queue head. The host controller must be able to advance the queue from the *Fetch QH* state in order to avoid all hardware/software race conditions. This simple mechanism allows software to simply link qTDs to the queue head and *activate* them, then the host controller always *find* them if/when they are reachable. The figure below illustrates the Host Controller Queue Head Traversal State Machine.

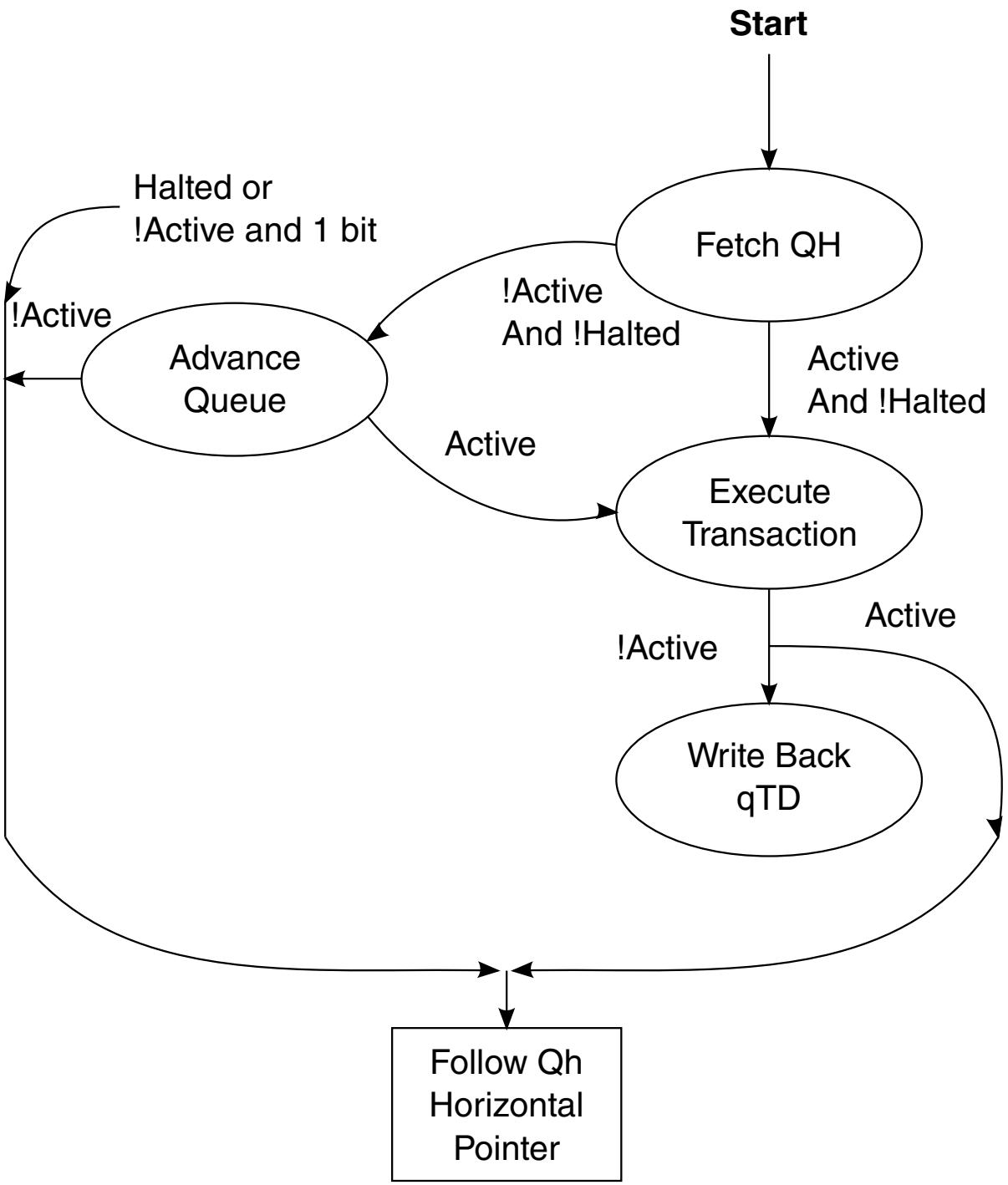


Figure 42-17. Host Controller Queue Head Traversal State Machine

This traversal state machine applies to all queue heads, regardless of transfer type or whether split transactions are required. The following sections describe each state. Each state description describes the entry criteria. The Execute Transaction state (see Execute

Transaction) describes the basic requirements for all endpoints. Split Transactions for Asynchronous Transfers and Split Transaction Interrupt describe details of the required extensions to the Execute Transaction state for endpoints requiring split transactions.

NOTE

Prior to software placing a queue head into either the periodic or asynchronous list, software must ensure the queue head is properly initialized. Minimally, the queue head should be initialized to the following (see Section Queue Head for layout of a queue head):

Valid static endpoint state.

- For the very first use of a queue head, software may zero-out the queue head transfer overlay, then set the *Next qTD Pointer* field value to reference a valid qTD.

42.5.3.10.1 Fetch Queue Head

A queue head can be referenced from the physical address stored in the ASYNCLISTADDR Register (see Next Asynch. Address (USB_*n*ASYNCLISTADDR))/ Endpoint List Address (USB_*n*ENDPTLISTADDR) Additionally, it may be referenced from the *Next LinkPointer* field of an iTD, siTD, FSTN or another Queue Head. If the referencing link pointer has the *Typ* field set to indicate a queue head, it is assumed to reference a queue head structure as defined in Table 42-26.

While in this state, the host controller performs operations to implement empty schedule detection (see Empty Asynchronous Schedule Detection) and Nak Counter reloads (see Operational Model for Nak Counter). After the queue head has been fetched, the host controller conducts the following queries for empty schedule detection:

- If queue head is not an interrupt queue head (that is *S-mask* is zero), and
- The *H-bit* is one, and
- The *Reclamation* bit in the USBSTS register is zero.

When these criteria are met, the host controller stops traversing the asynchronous list (as described in Empty Asynchronous Schedule Detection). When the criteria are not met, the host controller continues schedule traversal. If the queue head is not an interrupt and the *H-bit* is one and the *Reclamation* bit is one, then the host controller sets the *Reclamation* bit in the USBSTS register to zero before completing this state. The operations for reloading of the Nak Counter are described in detail in Operational Model for Nak Counter.

This state is complete when the queue head has been read on-chip.

42.5.3.10.2 Advance Queue

To advance the queue, the host controller must find the next qTD, adjust pointers, perform the overlay and write back the results to the queue head.

This state is entered from the FetchQHD state if the overlay *Active* and *Halt* bits are set to zero. On entry to this state, the host controller determines which next pointer to use to fetch a qTD, fetches a qTD and determines whether or not to perform an overlay.

NOTE

If the *I-bit* is one and the *Active* bit is zero, the host controller immediately skips processing of this queue head, exits this state and uses the horizontal pointer to the next schedule data structure. If the field *Bytes to Transfer* is not zero and the *T-bit* in the *Alternate Next qTD Pointer* is set to zero, then the host controller uses the *Alternate Next qTD Pointer*. Otherwise, the host controller uses the *NextqTD Pointer*. If *NextqTD Pointer's T-bit* is set to one, then the host controller exits this state and uses the horizontal pointer to the next schedule data structure.

Using the selected pointer the host controller fetches the referenced qTD. If the fetched qTD has its *Active* bit set to one, the host controller moves the pointer value used to reach the qTD (*Next* or *Alternate Next*) to the *Current qTD Pointer* field, then performs the overlay. If the fetched qTD has its *Active* bit set to zero, the host controller aborts the queue advance and follows the queue head's horizontal pointer to the next schedule data structure.

The host controller performs the overlay based on the following rules:

- The value of the data toggle (*dt*) field in the overlay area depends on the value of the *data toggle control* (*dtc*) bit (see Table 42-28).
- If the *EPS* field indicates the endpoint is a high-speed endpoint, the *Ping* state field is preserved by the host controller. The value of this field is not changed as a result of the overlay.
- *C-prog-mask* field is set to zero (field from incoming qTD is ignored, as is the current contents of the overlay area).
- *Frame Tag* field is set to zero (field from incoming qTD is ignored, as is the current contents of the overlay area).
- *NakCnt* field in the overlay area is loaded from the *RL* field in the queue head's Static Endpoint State.
- All other areas of the overlay are set by the incoming qTD.

The host controller exits this state when it has committed the write to the queue head.

42.5.3.10.3 Execute Transaction

The host controller enters this state from the Fetch Queue Head state only if the *Active* bit in *Status* field of the queue head is set to one.

On entry to this state, the host controller executes a few pre-operations, then checks some pre-condition criteria before committing to executing a transaction for the queue head.

The pre-operations performed and the pre-condition criteria depend on whether the queue head is an interrupt endpoint. The host controller can determine that a queue head is an interrupt queue head when the queue head's *S-mask* field contains a non-zero value. It is the responsibility of software to ensure the *S-mask* field is appropriately initialized based on the transfer type. There are other criteria that must be met if the *EPS* field indicates that the endpoint is a low- or full-speed endpoint, see Split Transactions for Asynchronous Transfers and Split Transaction Interrupt .

42.5.3.10.3.1 Interrupt Transfer Pre-condition Criteria

If the queue head is for an interrupt endpoint (for example, non-zero *S-mask* field), then the FRINDEX[2:0] field must identify a bit in the *S-mask* field that has one in it.

For example, an *S-mask* value of 00100000b would evaluate to true only when FRINDEX[2:0] is equal to 101b. If this condition is met then the host controller considers this queue head for a transaction.

42.5.3.10.3.2 Asynchronous Transfer Pre-operations and Pre-condition Criteria

If the queue head is not for an interrupt endpoint (for example, zero *S-mask* field), then the host controller performs one pre-operation and then evaluates one pre-condition criteria.

The pre-operation is:

Checks the Nak counter reload state (Operational Model for Nak Counter). It may be necessary for the host controller to reload the Nak Counter field. The reload is performed at this time.

The pre-condition evaluated is:

- Whether or not the *NakCnt* field has been reloaded, the host controller checks the value of the *NakCnt* field in the queue head. If *NakCnt* is non-zero, or if the *Reload Nak Counter* field is zero, then the host controller considers this queue head for a transaction.

42.5.3.10.3.3 Transfer Type Independent Pre-operations

Regardless of the transfer type, the host controller always performs at least one pre-operation and evaluates one pre-condition. The pre-operation is:

- A host controller internal transaction (down) counter *qHTransactionCounter* is loaded from the queue head's *Mult* field. A host controller implementation is allowed to ignore this for queue heads on the asynchronous list. It is mandatory for interrupt queue heads. Software should ensure that the *Mult* field is set appropriately for the transfer type.

The pre-conditions evaluated are:

- The host controller determines whether there is enough time in the micro-frame to complete this transaction (see Transaction Fit - A Best-Fit Approximation Algorithm for an example evaluation method). If there is not enough time to complete the transaction, the host controller exits this state.
- If the value of *qHTransactionCounter* for an interrupt endpoint is zero, then the host controller exits this state.

When the pre-operations are complete and pre-conditions are met, the host controller sets the *Reclamation* bit in the USBSTS register to one and then begins executing one or more transactions using the endpoint information in the queue head. The host controller iterates *qHTransactionCounter* times in this state executing transactions. After each transaction is executed, *qHTransactionCounter* is decremented by one. The host controller exits this state when one of the following events occurs:

- The *qHTransactionCounter* decrements to zero, or
- The endpoint responds to the transaction with any handshake other than an ACK,[4] or
- The transaction experiences a transaction error, or
- The *Active* bit in the queue head goes to zero, or
- There is not enough time in the micro-frame left to execute the next transaction(see Transaction Fit - A Best-Fit Approximation Algorithm) for example method for implementing the frame boundary test).

NOTE

For a high-bandwidth interrupt OUT endpoint, the host controller may optionally immediately retry the transaction if it fails.

The results of each transaction is recorded in the on-chip overlay area. If data was successfully moved during the transaction, the transfer state in the overlay area is advanced. To advance queue head's transfer state, the *Total Bytes to Transfer* field is decremented by the number of bytes moved in the transaction, the data toggle bit (*dt*) is toggled, the current page offset is advanced to the next appropriate value (for example,

advanced by the number of bytes successfully moved), and the *C_Page* field is updated to the appropriate value (if necessary). See Buffer Pointer List Use for Data Streaming with qTDs .

NOTE

The *Total Bytes To Transfer* field may be zero when all the other criteria for executing a transaction are met. When this occurs, the host controller executes zero-length transaction to the endpoint. If the *PID_Code* field indicates an IN transaction and the device delivers data, the host controller detects a packet babble condition, set the *babble* and *halted* bits in the *Status* field, set the *Active* bit to zero, write back the results to the source qTD, then exit this state.

In the event an IN token receives a data PID mismatch response, the host controller must ignore the received data (for example not advance the transfer state for the bytes received). Additionally, if the endpoint is an interrupt IN, then the host controller must record that the transaction occurred (for example, decrement *qHTransactionCounter*). It is recommended (but not required) the host controller continue executing transactions for this endpoint if the resultant value of *qHTransactionCounter* is greater than one.

If the response to the IN bus transaction is a Nak (or Nyet) and *RL* is non-zero, *NakCnt* is decremented by one. If *RL* is zero, then no write-back by the host controller is required (for a transaction receiving a Nak or Nyet response and the value of *CErr* did not change). Software should set the *RL* field to zero if the queue head is an interrupt endpoint. Host controller hardware is not required to enforce this rule or operation.

After the transaction has finished and the host controller has completed the post processing of the results (advancing the transfer state and possibly *NakCnt*, the host controller writes back the results of the transaction to the queue head's overlay area in main memory).

The number of bytes moved during an IN transaction depends on how much data the device endpoint delivers. The maximum number of bytes a device can send is *MaximumPacket Size*. The number of bytes moved during an OUT transaction is either *Maximum Packet Length* bytes or *Total Bytes to Transfer*, whichever is less.

If there was a transaction error during the transaction, the transfer state (as defined above) is not advanced by the host controller. The *CErr* field is decremented by one and the status field is updated to reflect the type of error observed. Transaction errors are summarized in Transaction Error .

The following events causes the host controller to clear the *Active* bit in the queue head's overlay status field. When the *Active* bit transitions from one to zero, the transfer in the overlay is considered complete. The reason for the transfer completion (clearing the *Active* bit) determines the next state.

- *CErr* field decrements to zero. When this occurs the *Halted* bit is set to one and *Active* is set to zero. This results in the hardware not advancing the queue and the pipe halts. Software must intercede to recover.
- The device responds to the transaction with a STALL PID. When this occurs, the *Halted* bit is set to one and the *Active* bit is set to zero. This results in the hardware not advancing the queue and the pipe halts. Software must intercede to recover.
- The *Total Bytes to Transfer* field is zero after the transaction completes.
  - For a zero length transaction, it was zero before the transaction was started. When this condition occurs, the *Active* bit is set to zero.
- The PID code is an IN, and the number of bytes moved during the transaction is less than the *Maximum Packet Length*. When this occurs, the *Active* bit is set to zero and a short packet condition exists. The short-packet condition is detected during the Advance Queue state. Refer to Split Transactions for additional rules for managing low- and full-speed transactions.

With the exception of a NAK response (when *RL* field is zero), the host controller always writes the results of the transaction back to the overlay area in main memory. This includes when the transfer completes. For a high-speed endpoint, the queue head information written back includes minimally the following fields: The *PID Code* field indicates an IN and the device sends more than the expected number of bytes (for example *Maximum Packet Length* or *Total Bytes to Transfer* bytes, whichever is less) (for example a packet babble). This results in the host controller setting the *Halted* bit to one.

- NakCnt, dt, Total Bytes to Transfer, C_Page, Status, CERR, and Current Offset

For a low- or full-speed device the queue head information written back also includes the fields:

- C-prog-mask, FrameTag and S-bytes.

The duration of this state depends on the time it takes to complete the transaction(s) and the status write to the overlay is committed.

42.5.3.10.3.4 Halting a Queue Head

A halted endpoint is defined only for the transfer types that are managed through queue heads (control, bulk and interrupt).

The following events indicate that the endpoint has reached a condition where no more activity can occur without intervention from the driver:

- An endpoint may return a STALL handshake during a transaction,
- A transaction had three consecutive error conditions, or
- A Packet Babble error occurs on the endpoint.

When any of these events occur (for a queue head) the Host Controller halts the queue head and set the USBERRINT status bit in the USB_n_USBSTS register to one. To halt the queue head, the *Active* bit is set to zero and the *Halted* bit is set to one. There may be other error status bits that are set when a queue is halted. The host controller always writes back the overlay area to the source qTD when the transfer is complete, regardless of the reason (normal completion, short packet or halt). The host controller does not advance the transfer state on a transaction that results in a *Halt* condition (for example no updates necessary for *Total Bytes to Transfer*, *C_Page*, *Current Offset,* and *dt*). The host controller must update *CErr* as appropriate. When a queue head is halted, the *USB Error Interrupt* bit in the USB_n_USBSTS register is set to one. If the *USB Error Interrupt Enable* bit in the USB_n_USBINTR register is set to one, a hardware interrupt is generated at the next interrupt threshold.

42.5.3.10.3.5 Asynchronous Schedule Park Mode

Asynchronous Schedule Park mode is a special execution mode that can be enabled by system software, where the host controller is permitted to execute more than one bus transaction from a high-speed queue head in the Asynchronous schedule before continuing horizontal traversal of the Asynchronous schedule.

This feature has no effect on queue heads or other data structures in the Periodic schedule. This feature is similar in intent as the *Mult* feature that is used in the Periodic schedule. Where-as the *Mult* feature is a characteristic that is tunable for each endpoint; park-mode is a policy that is applied to all high-speed queue heads in the asynchronous schedule. It is essentially the specification of an iterator for consecutive bus transactions to the same endpoint. All of the rules for managing bus transactions and the results of those as defined in Execute Transaction apply. This feature merely specifies how many consecutive times the host controller is permitted to execute from the same queue head before moving to the next queue head in the Asynchronous List. This feature should allow the host controller to attain better bus utilization for those devices that are capable of moving data at maximum rate, while at the same time providing a fair service to all endpoints.

A host controller exports its capability to support this feature to system software by setting the *Asynchronous Schedule Park Capability* bit in the USB_n_HCCPARAMs register to one. This information keys system software that the *Asynchronous Schedule Park Mode Enable* and *Asynchronous Schedule Park Mode Count* fields in the USB_n_USBCMD register are modifiable. System software enables the feature by writing a one to the *Asynchronous Schedule Park Mode Enable* bit.

When park-mode is not enabled (for example *Asynchronous Schedule Park Mode Enable* bit in the USB_n_USBCMD register is zero), the host controller must not execute more than one bus transaction per high-speed queue head, per traversal of the asynchronous schedule. When park-mode is enabled, the host controller must not apply the feature to a queue head whose *EPS* field indicates a Low/Full-speed device (for example only one bus transaction is allowed from each Low/Full-speed queue head per traversal of the asynchronous schedule). Park-mode may only be applied to queue heads in the Asynchronous schedule whose *EPS* field indicates that it is a high-speed device.

The host controller must apply park mode to queue heads whose *EPS* field indicates a high-speed endpoint. The maximum number of consecutive bus transactions a host controller may execute on a high-speed queue head is determined by the value in the *Asynchronous Schedule Park Mode Count* field in the USB_n_USBCMD register. Software must not set *Asynchronous Schedule Park Mode Enable* bit to one and also set *Asynchronous Schedule Park Mode Count* field to zero. The resulting behavior is not defined. An example behavioral example describes the operational requirements for the host controller implementing park-mode. This feature does not affect how the host controller handles the bus transaction as defined in Execute Transaction . It only effects how many consecutive bus transactions for the current queue head can be executed. All boundary conditions, error detection and reporting applies as usual. This feature is similar in concept to the use of the *Mult* field for high-bandwidth Interrupt for queue heads in the Periodic Schedule.

The host controller effectively loads an internal down-counter *PM-Count* from *Asynchronous Schedule Park Mode Count* when *Asyncrhonous Schedule Park Mode Enable* bit is one, and a high-speed queue head is first fetched and meets all the criteria for executing a bus transaction. After the bus transaction, *PM-Count* is decremented. The host controller may continue to execute bus transactions from the current queue head until *PM-Count* goes to zero, an error is detected, the buffer for the current transfer is exhausted or the endpoint responds with a flow-control or STALL handshake.

The following table summarizes the responses that effect whether the host controller continues with another bus transaction for the current queue head.

Table 42-44. Actions for Park Mode, based on Endpoint Response and Residual Transfer State

PID	Endpoint Response	Transfer State after Transaction		Action
		PM-Count	Bytes to Transfer	
IN	DATA[0,1] w/Maximum Packet sized data	Not zero	Not Zero	Allowed to perform another bus transaction.[1, 2]
		Not zero	Zero	Retire qTD and move to next QH
		Zero	Don't care	Move to next QH.

Table continues on the next page...

Table 42-44. Actions for Park Mode, based on Endpoint Response and Residual Transfer State (continued)

	DATA[0,1] w/short packet	Don't care	Don't care	Retire qTD and move to next QH.
	NAK	Don't care	Don't care	Move to next QH.
	STALL, XactErr	Don't care	Don't care	Move to next QH.
OUT	ACK	Not zero	Not Zero	Allowed to perform another bus transaction.[2]
		Not zero	Zero	Retire qTD and move to next QH
		Zero	Don't' care	Move to next QH.
	NYET, NAK	Don't care	Don't care	Move to next QH.
	STALL, XactErr	Don't care	Don't care	Move to next QH
PING	ACK	Not Zero	Not Zero	Allowed to perform another bus transaction. [2]
	NAK	Don't care	Don't care	Move to next QH
	STALL, XactErr	Don't care	Don't care	Move to next QH

1. The host controller may continue to execute bus transactions from the current high-speed queue head (if *PM-Count* is not equal to zero), if a PID mismatch is detected (for example expected DATA1 and received DATA0, or visa-versa).
2. This specification does not *require* that the host controller execute another bus transaction when *PM-Count* is non-zero. Implementations are encouraged to make appropriate complexity and performance trade-offs.

42.5.3.10.4 Write Back qTD

This state is entered from the Execute Transaction state when the *Active* bit is set to zero.

The source data for the write-back is the transfer results area of the queue head overlay area (see Table 42-44).

The host controller uses the *Current qTD Pointer* field as the target address for the qTD.

The queue head transfer result area is written back to the transfer result area of the target qTD. This state is also referred to as: qTD retirement. The fields that must be written back to the source qTD include *Total Bytes to Transfer*, *Cerr*, and *Status*.

The duration of this state depends on when the qTD write-back is committed.

42.5.3.10.5 Follow Queue Head Horizontal Pointer

The host controller must use the horizontal pointer in the queue head to the next schedule data structure when any of the following conditions exist:

- If the *Active* bit is one on exit from the Execute Transaction state, or
- When the host controller exits the Write Back qTD state, or
- If the Advance Queue state fails to advance the queue because the target qTD is not active, or
- If the *Halted* bit is one on exit from the Fetch QH state.

There is no functional requirement that the host controller wait until the current transaction is complete before using the horizontal pointer to read the next linked data structure. However, it must wait until the current transaction is complete before executing the next data structure.

42.5.3.10.6 Buffer Pointer List Use for Data Streaming with qTDs

A qTD has an array of buffer pointers, which is used to reference the data buffer for a transfer. This specification requires that the buffer associated with the transfer be *virtually contiguous*.

This means: if the buffer spans more than one physical page, it must obey the following rules (the figure below illustrates an example):

- The first portion of the buffer must begin at some offset in a page and extend through the end of the page.
- The remaining buffer cannot be allocated in small chunks scattered around memory. For each 4 K chunk beyond the first page, each buffer portion matches to a full 4 K page. The final portion, which may only be large enough to occupy a portion of a page, must start at the top of the page and be contiguous within that page.

The buffer pointer list in the qTD is long enough to support a maximum transfer size of 20 K bytes. This case occurs when all five buffer pointers are used and the first offset is zero. A qTD handles a 16 Kbyte buffer with any starting buffer alignment.

The host controller uses the field *C_Page* field as an index value to determine which buffer pointer in the list should be used to start the current transaction. The host controller uses a different buffer pointer for each physical page of the buffer. This is always true, even if the buffer is physically contiguous.

The host controller must detect when the current transaction spans a page boundary and automatically move to the next available buffer pointer in the page pointer list. The next available pointer is reached by incrementing *C_Page* and pulling the next page pointer from the list. Software must ensure there are sufficient buffer pointers to move the amount of data specified in the *Bytes to Transfer* field.

The following figure illustrates a nominal example of how System software would initialize the buffer pointers list and the *C_Page* field for a transfer size of 16383 bytes. *C_Page* is set to zero. The upper 20-bits of Page 0 references the start of the physical page. *Current Offset* (the lower 12-bits of queue head Dword 7) holds the offset in the page for example 2049 (for example 4096-2047). The remaining page pointers are set to reference the beginning of each subsequent 4 K page.

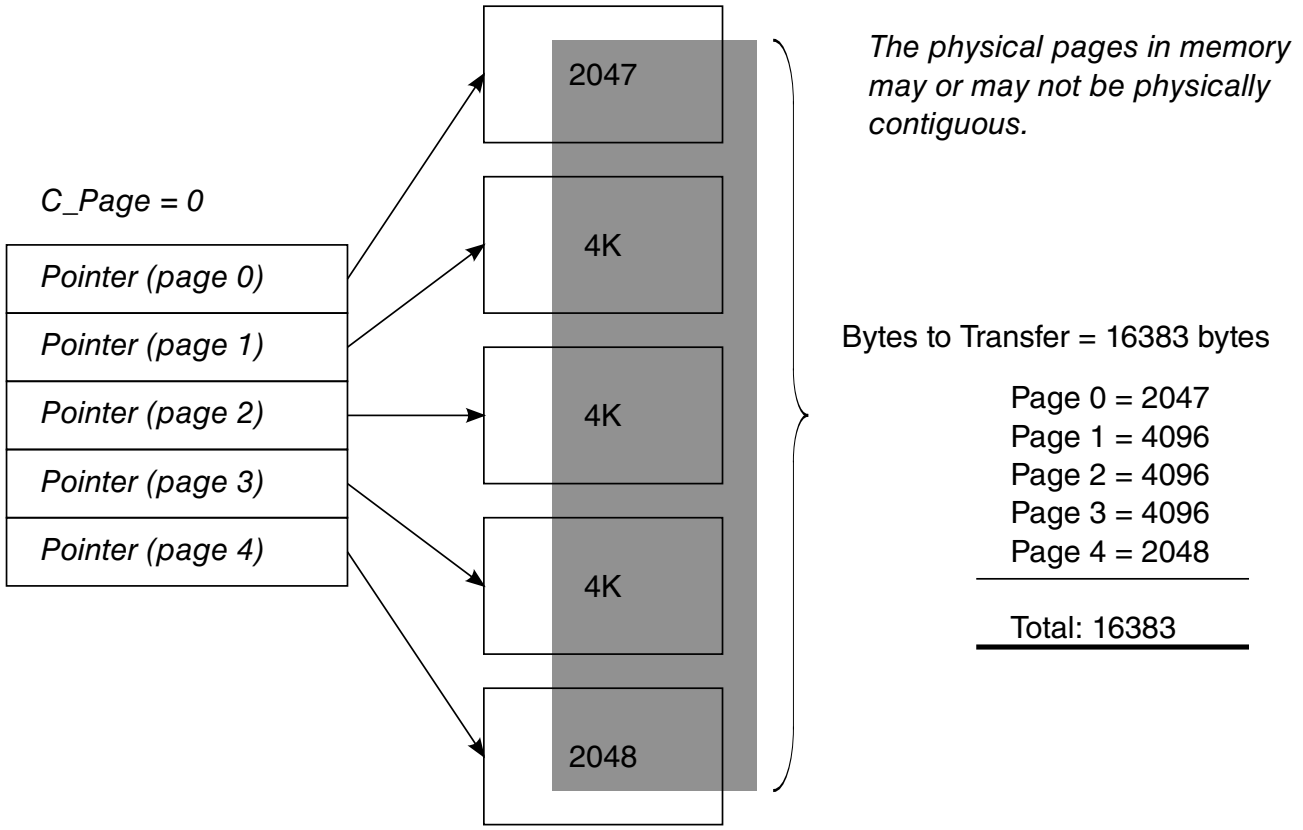


Figure 42-18. Example Mapping of qTD Buffer Pointers to Buffer Pages

For the first transaction on the qTD (assuming a 512-byte transaction), the host controller uses the first buffer pointer (page 0 because *C_Page* is set to zero) and concatenates the *Current Offset* field. The 512 bytes are moved during the transaction, the *Current Offset* and *Total Bytes to Transfer* are adjusted by 512 and written back to the queue head working area.

During the 4th transaction, the host controller needs 511 bytes in page 0 and one byte in page 1. The host controller increments *C_Page* (to 1) and use the page 1 pointer to move the final byte of the transaction. After the 4th transaction, the active page pointer is the page 1 pointer and *Current Offset* has rolled to one, and both are written back to the overlay area. The transactions continue for the rest of the buffer, with the host controller automatically moving to the next page pointer (that is *C_Page*) when necessary. The three conditions for how the host controller handles *C_Page*:

- The current transaction does not span a page boundary. The value of *C_Page* is not adjusted by the host controller.

- The current transaction does span a page boundary. The host controller must detect the page cross condition and advance to the next buffer while streaming data to/from the USB.
- The current transaction completes on a page boundary (that is the last byte moved for the current transaction is the last byte in the page for the current page pointer). The host controller must increment *C_Page* before writing back status for the transaction.

NOTE

The only valid adjustment the host controller may make to *C_Page* is to increment by one.

42.5.3.10.7 Adding Interrupt Queue Heads to the Periodic Schedule

The link path(s) from the periodic frame list to a queue head establishes in which frames a transaction can be executed for the queue head. Queue heads are linked into the periodic schedule so they are polled at the appropriate rate.

System software sets a bit in a queue head's *S-Mask* to indicate which micro-frame within 1 msec period a transaction should be executed for the queue head. Software must ensure that all queue heads in the periodic schedule have *S-Mask* set to a non-zero value. An *S-mask* with zero value in the context of the periodic schedule yields undefined results.

If the desired poll rate is greater than one frame, system software can use a combination of queue head linking and *S-Mask* values to spread interrupts of equal poll rates through the schedule so that the periodic bandwidth is allocated and managed in the most efficient manner possible. Some examples are illustrated in the following table.

Table 42-45. Example Periodic Reference Patterns for Interrupt Transfers with 2ms Poll Rate

Frame # Reference Sequence	Description
0, 2, 4, 6, 8, and so on *S-Mask* = 01h	A queue head for the *bInterval* of 2 msec (16 micro-frames) is linked into the periodic schedule so that it is reachable from the periodic frame list locations indicated in the previous column. In addition, the *S-Mask* field in the queue head is set to 01h, indicating that the transaction for the endpoint should be executed on the bus during micro-frame 0 of the frame.
0, 2, 4, 6, 8, and so on *S-Mask* = 02h	Another example of a queue head with a *bInterval* of 2 msec is linked into the periodic frame list at exactly the same interval as the previous example. However, the *S-Mask* is set to 02h indicating that the transaction for the endpoint should be executed on the bus during micro-frame 1 of the frame.

42.5.3.10.8 Managing Transfer Complete Interrupts from Queue Heads

The host controller sets an interrupt to be signaled at the next interrupt threshold when the completed transfer (qTD) has an *Interrupt on Complete* (*IOC*) bit set to one, or whenever a transfer (qTD) completes with a short packet.

If system software needs multiple qTDs to complete a client request (that is like a control transfer) the intermediate qTDs do not require interrupts. System software may only need a single interrupt to notify it that the complete buffer has been transferred. System software may set IOC's to occur more frequently. A motivation for this may be that it wants early notification so that interface data structures can be re-used in a timely manner.

42.5.3.11 Ping Control

USB 2.0 defines an addition to the protocol for high-speed devices called Ping. Ping is required for all USB 2.0 High-speed bulk and control endpoints.

Ping is not allowed for a split-transaction stream. This extension to the protocol eliminates the bad side-effects of Naking OUT endpoints. The *Status* field has a *Ping State* bit, which the host controller uses to determine the *next* actual PID it uses in the next transaction to the endpoint (see the table below).

The Ping State bit is only managed by the host controller for queue heads that meet the following criteria:

- Queue head is not an interrupt and
- *EPS* field equals High-Speed and
- *PIDCode* field equals OUT

The following table illustrates the state transition table for the host controller's responsibility for maintaining the PING protocol. Refer to Chapter 8 in the USB Specification Revision 2.0 for detailed description on the Ping protocol.

Table 42-46. Ping Control State Transition Table

Event			
Current	Host	Device	Next
Do Ping	PING	Nak	Do Ping
Do Ping	PING	Ack	Do OUT
Do Ping	PING	XactErr [1]	Do Ping
Do Ping	PING	Stall	N/C [2] Do
OUT	OUT	Nak	Do Ping
Do OUT	OUT	Nyet	Do Ping

Table continues on the next page...

Table 42-46. Ping Control State Transition Table (continued)

Do OUT	OUT	Ack	Do OUT
Do OUT	OUT	XactErr[1]	Do Ping
Do OUT	OUT	Stall	N/C [2]

1. Transaction Error (XactErr) is any time the host misses the handshake.
2. No transition change required for the Ping State bit. The Stall handshake results in the endpoint being halted (for example Active set to zero and Halt set to one). Software intervention is required to restart queue. 3 A Nyet response to an OUT means that the device has accepted the data, but cannot receive any more at this time. Host must advance the transfer state and additionally, transition the Ping State bit to Do Ping. The Ping State bit has the following encoding:

Table 42-47. Ping State bit Encoding

Value	Meaning
0B	Do OUT The host controller uses an OUT PID during the next bus transaction to this endpoint.
1B	Do Ping The host controller uses a PING PID during the next bus transaction to this endpoint.

The defined ping protocol (see USB 2.0 Specification, Chapter 8) allows the host to be *imprecise* on the initialization of the ping protocol (that is start in *Do OUT* when we don't know whether there is space on the device or not). The host controller manages the *Ping State* bit. System software sets the initial value in the queue head when it initializes a queue head. The host controller preserves the *Ping State* bit across all queue advancements. This means that when a new qTD is written into the queue head overlay area, the previous value of the *Ping State* bit is preserved.

42.5.3.12 Split Transactions

USB 2.0 defines extensions to the bus protocol for managing USB 1.x data streams through USB 2.0 Hubs.

This section describes how the host controller uses the interface data structures to manage data streams with full- and low-speed devices, connected below USB 2.0 hub, utilizing the split transaction protocol.

Refer to USB 2.0 Specification for the complete definition of the split transaction protocol. Full- and Low-speed devices are enumerated identically as high-speed devices, but the transactions to the Full- and Low-speed endpoints use the split-transaction protocol on the high-speed bus. The split transaction protocol is an encapsulation of (or wrapper around) the Full- or Low-speed transaction. The high-speed wrapper portion of the protocol is addressed to the USB 2.0 Hub and Transaction Translator below which the Full- or Low-speed device is attached.

The EHCI interface uses dedicated data structures for managing full-speed isochronous data streams (see Split Transaction Isochronous Transfer Descriptor (siTD)). Control, Bulk and Interrupt are managed using the queuing data structures (see Queue Head). The interface data structures need to be programmed with the device address and the Transaction Translator number of the USB 2.0 Hub operating as the Low-/Full-speed host controller for this link. The following sections describe the details of how the host controller must process and manage the split transaction protocol.

42.5.3.12.1 Split Transactions for Asynchronous Transfers

A queue head in the asynchronous schedule with an *EPS* field indicating a full-or low-speed device indicates to the host controller that it must use split transactions to stream data for this queue head.

All full-speed bulk and full-, low-speed control are managed through queue heads in the asynchronous schedule.

Software must initialize the queue head with the appropriate device address and port number for the transaction translator that is serving as the full/low-speed host controller for the links connecting the endpoint. Software must also initialize the split transaction state bit (*SplitXState*) to Do-Start-Split. Finally, if the endpoint is a control endpoint, then system software must set the *Control Transfer Type* (*C*) bit in the queue head to one. If this is not a control transfer type endpoint, the *C* bit must be initialized by software to be zero. This information is used by the host controller to properly set the Endpoint Type (ET) field in the split transaction bus token. When the *C* bit is zero, the split transaction token's ET field is set to indicate a bulk endpoint. When the *C* bit is one, the split transaction token's ET field is set to indicate a control endpoint. Refer to Chapter 8 of USB Specification Revision 2.0 for details.

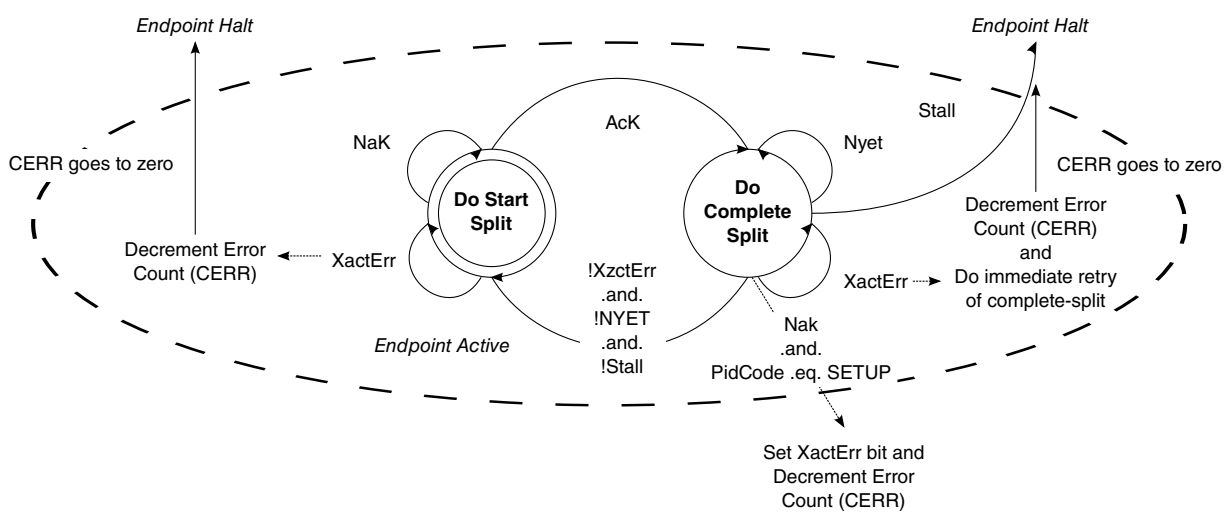


Figure 42-19. Host Controller Asynchronous Schedule Split-Transaction State Machine

42.5.3.12.1.1 Asynchronous - Do Start Split

This is the state which software must initialize a full- or low-speed asynchronous queue head. This state is entered from the Do Complete Split state only after a complete-split transaction receives a valid response from the transaction translator that is not a Nyet handshake.

For queue heads in this state, the host controller executes a start-split transaction to the appropriate transaction translator. If the bus transaction completes without an error and *PidCode* indicates an IN or OUT transaction, then the host controller reloads the error counter (*CErr*). If it is a successful bus transaction and the *PidCode* indicates a SETUP, the host controller does not reload the error counter. If the transaction translator responds with a Nak, the queue head is left in this state, and the host controller proceeds to the next queue head in the asynchronous schedule.

If the host controller times out the transaction (no response, or bad response) the host controller decrements *Cerr* and proceeds to the next queue head in the asynchronous schedule.

42.5.3.12.1.2 Asynchronous - Do Complete Split

This state is entered from the Do Start Split state only after a start-split transaction receives an Ack handshake from the transaction translator.

For queue heads in this state, the host controller executes a complete-split transaction to the appropriate transaction translator. If the transaction translator responds with a Nyet handshake, the queue head is left in this state, the error counter is reset and the host controller proceeds to the next queue head in the asynchronous schedule. When a Nyet handshake is received for a bus transaction where the queue head's *PidCode* indicates an IN or OUT, the host controller reloads the error counter (*CErr*). When a Nyet handshake is received for a complete-split bus transaction where the queue head's *PidCode* indicates a SETUP, the host controller must not adjust the value of *CErr*.

Independent of *PIDCode*, the following responses have the effects:

- Transaction Error (XactErr). Timeout or data CRC failure, and so on. The error counter (*Cerr*) is decremented by one and the complete split transaction is *immediately* retried (if possible). If there is not enough time in the micro-frame to execute the retry, the host controller MUST ensure that the next time the host controller begins executing from the Asynchronous schedule, it must begin executing from this queue head. If another start-split (for some other endpoint) is sent to the transaction translator before the complete-split is really completed, the transaction translator could dump the results (which were never delivered to the host). This is why the core specification states the retries must be immediate. A method to

accomplish this behavior is to not advance the asynchronous schedule. When the host controller returns to the asynchronous schedule in the next micro-frame, the first transaction from the schedule is the retry for this endpoint.

If *Cerr* went to zero, the host controller must halt the queue.

- NAK. The target endpoint Nak'd the full- or low-speed transaction. The state of the transfer is not advanced and the state is exited. If the *PidCode* is a SETUP, then the Nak response is a protocol error. The *XactErr* status bit is set to one and the *CErr* field is decremented.
- STALL. The target endpoint responded with a STALL handshake. The host controller sets the *halt* bit in the status byte, retires the qTD but does not attempt to advance the queue.

If the *PidCode* indicates an IN, then any of following responses are expected:

- DATA0/1. On reception of data, the host controller ensures the PID matches the expected data toggle and checks CRC. If the packet is *good*, the host controller advances the state of the transfer, for example move the data pointer by the number of bytes received, decrement *BytesToTransfer* field by the number of bytes received, and toggle the *dt* bit. The host controller then exit this state. The response and advancement of transfer may trigger other processing events, such as retirement of the qTD and advancement of the queue.

If the data sequence PID does not match the expected, the data is ignored, the transfer state is not advanced and this state is exited. If the *PidCode* indicates an OUT/SETUP, then any of following responses are expected:

- ACK. The target endpoint accepted the data, so the host controller must advance the state of the transfer. The *Current Offset* field is incremented by *Maximum Packet Length* or *Bytes to Transfer*, whichever is less. The field *Bytes To Transfer* is decremented by the same amount and the data toggle bit (*dt*) is toggled. The host controller then exit this state.
- Advancing the transfer state may cause other processing events such as retirement of the qTD and advancement of the queue (see Managing Control/Bulk/Interrupt Transfers through Queue Heads).

42.5.3.12.2 Split Transaction Interrupt

Split-transaction Interrupt-IN/OUT endpoints are managed through the same data structures used for high-speed interrupt endpoints. They both co-exist in the periodic schedule.

Queue heads/qTDs offer the set of features required for reliable data delivery, which is characteristic to interrupt transfer types. The split-transaction protocol is managed completely within this defined functional transfer framework. For example, for a high-speed endpoint, the host controller visits a queue head, execute a high-speed transaction (if criteria are met) and advance the transfer state (or not) depending on the results of the entire transaction. For low- and full-speed endpoints, the details of the *execution* phase are different (that is takes more than one bus transaction to complete), but the remainder of the operational framework is intact. This means that the transfer advancement, and so on, occurs as defined in Managing Control/Bulk/Interrupt Transfers through Queue Heads, but only occurs on the completion of a split transaction.

42.5.3.12.2.1 Split Transaction Scheduling Mechanisms for Interrupt

Full- and low-speed Interrupt queue heads have an *EPS* field indicating full- or low-speed and have a non-zero *S-mask* field.

The host controller can detect this combination of parameters and assume the endpoint is a periodic endpoint. Low- and full-speed interrupt queue heads require the use of the split transaction protocol. The host controller sets the Endpoint Type (ET) field in the split token to indicate the transaction is an interrupt. These transactions are managed through a transaction translator's periodic pipeline. Software should not set these fields to indicate the queue head is an interrupt unless the queue head is used in the periodic schedule.

System software manages the per/transaction translator periodic pipeline by budgeting and scheduling exactly during which micro-frames the start-splits and complete-splits for each endpoint occurs. The characteristics of the transaction translator are such that the high-speed transaction protocol must execute during explicit micro-frames, or the data or response information in the pipeline is lost.

The following figure illustrates the general scheduling boundary conditions that are supported by the EHCI periodic schedule and queue head data structure. The S and ^{C}X labels indicate micro-frames where software can schedule start-splits and complete splits (respectively).

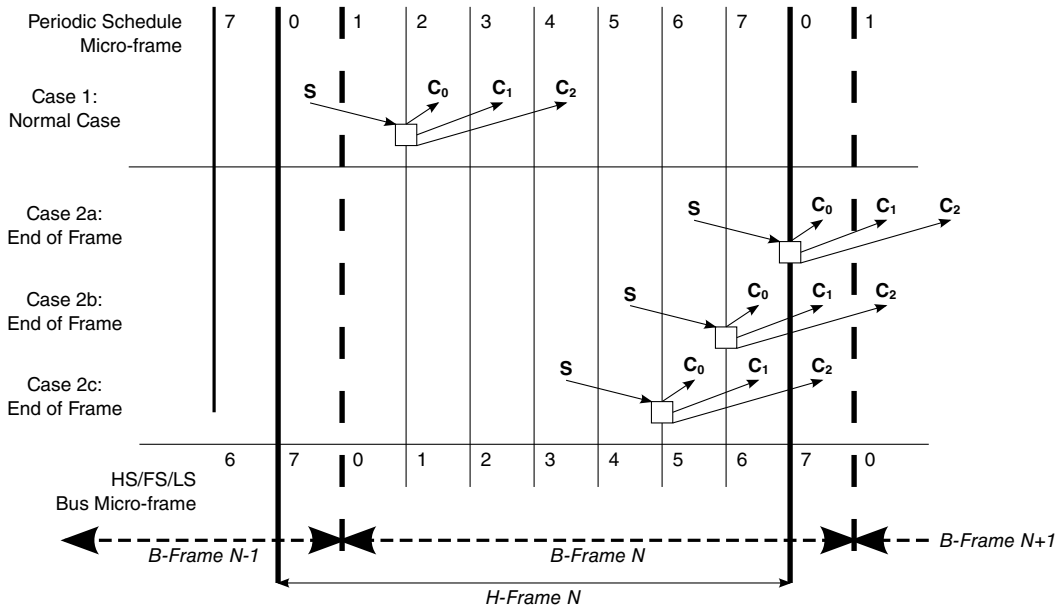


Figure 42-20. Split Transaction, Interrupt Scheduling Boundary Conditions

The scheduling cases are:

- Case 1: The normal scheduling case is where the entire split transaction is completely bounded by a frame (*H-Frame* in this case).
- Case 2a through Case 2c: The USB 2.0 Hub pipeline rules states clearly, when and how many complete-splits must be scheduled to account for earliest to latest execution on the full/low-speed link. The complete-splits may span the *H-Frame* boundary when the start-split is in micro-frame 4 or later. When this occurs, the *H-Frame* to *B-Frame* alignment requires that the queue head be reachable from consecutive periodic frame list locations. System software cannot build an efficient schedule that satisfies this requirement unless it uses FSTNs.

The figure below illustrates the general layout of the periodic schedule.

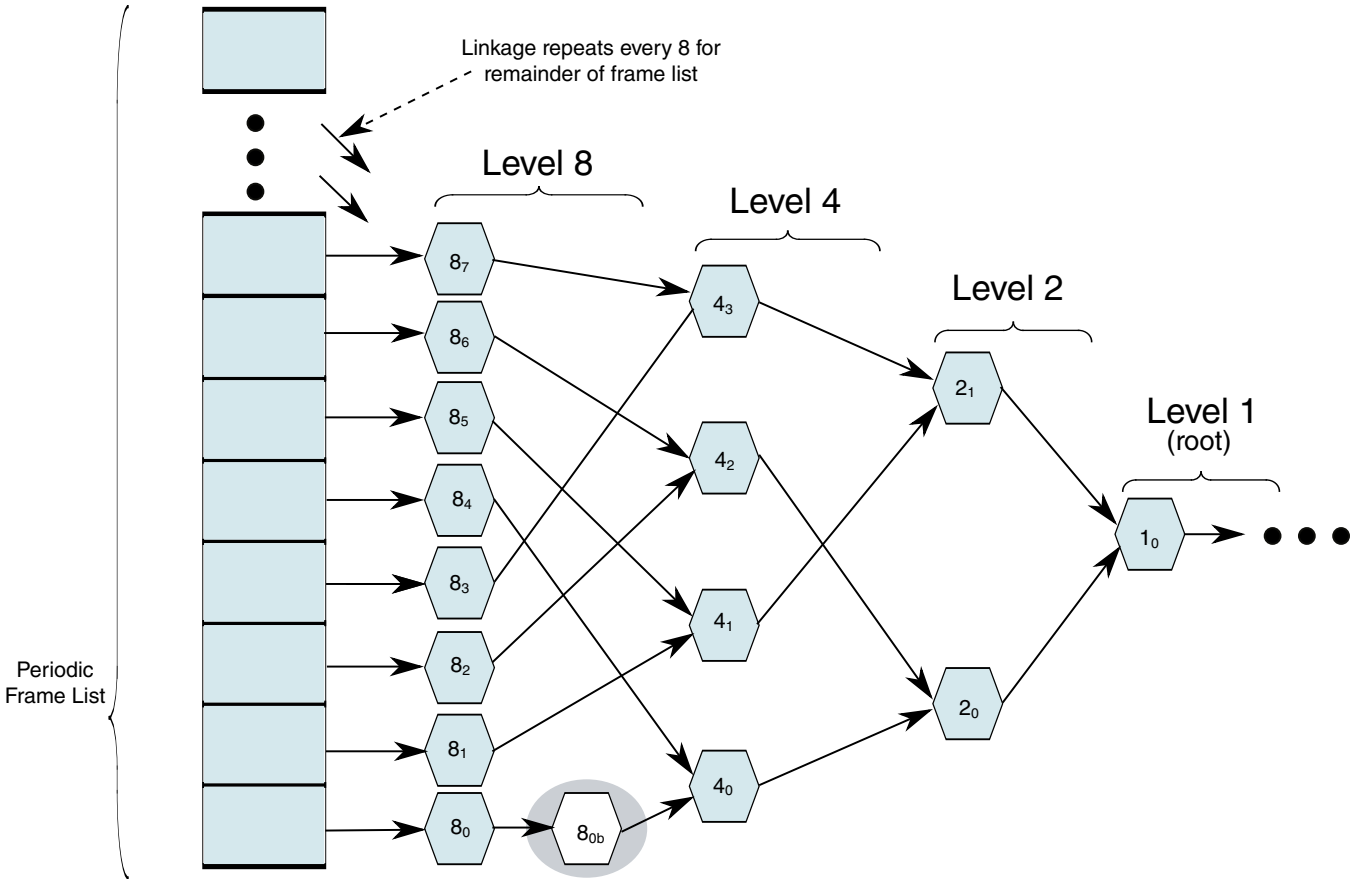


Figure 42-21. General Structure of EHCI Periodic Schedule Utilizing Interrupt Spreading

The periodic frame list is effectively the leaf level a binary tree, which is always traversed leaf to root. Each level in the tree corresponds to a 2^N poll rate. Software can efficiently manage periodic bandwidth on the USB by *spreading* interrupt queue heads that have the same poll rate requirement across all the available paths from the frame list. For example, system software can schedule eight poll rate 8 queue heads and account for them once in the high-speed bus bandwidth allocation.

When an endpoint is allocated an execution footprint that spans a frame boundary, the queue head for the endpoint must be reachable from consecutive locations in the frame list. An example would be if 8_{0b} where such an endpoint. Without additional support on the interface, to get 8_{0b} reachable at the correct time, software would have to link 8_1 to 8_{0b}. It would then have to move 4_1 and everything linked after into the same path as 4_0. This upsets the integrity of the binary tree and disallows the use of the spreading technique.

FSTN data structures are used to preserve the integrity of the binary-tree structure and enable the use of the spreading technique. Host Controller Operational Model for FSTNs defines the hardware and software operational model requirements for using FSTNs.

The following queue head fields are initialized by system software to instruct the host controller when to execute portions of the split-transaction protocol:

- *SplitXState*. This is single bit residing in the *Status* field of a queue head (see Table 42-24). This bit is used to track the current state of the split transaction.
- *Frame S-mask*. This is a bit-field where-in system software sets a bit corresponding to the micro-frame (within an *H-Frame*) that the host controller should execute a start-split transaction. This is always qualified by the value of the *SplitXState* bit in the *Status* field of the queue head. For example, referring to Figure 42-20, case one, the *S-mask* would have a value of 00000001b indicating that if the queue head is traversed by the host controller, and the *SplitXState* indicates Do_Start, and the current micro-frame as indicated by FRINDEX[2:0] is 0, then execute a start-split transaction.
- *Frame C-mask.* This is a bit-field where system software sets one or more bits corresponding to the micro-frames (within an *H-Frame*) that the host controller should execute complete-split transactions. The interpretation of this field is always qualified by the value of the *SplitXState* bit in the *Status* field of the queue head. For example, referring to Figure 42-20, case one, the *C-mask* would have a value of 00011100b indicating that if the queue head is traversed by the host controller, and the *SplitXState* indicates Do_Complete, and the current micro-frame as indicated by FRINDEX[2:0] is 2, 3, or 4, then execute a complete-split transaction. It is software's responsibility to ensure that the translation between *H-Frames* and *B-Frames* is correctly performed when setting bits in *S-mask* and *C-mask*

42.5.3.12.2.2 Host Controller Operational Model for FSTNs

The FSTN data structure is used to manage Low/Full-speed interrupt queue heads that need to be reached from consecutive frame list locations (that is boundary cases 2a through 2c).

An FSTN is essentially a *back pointer*, similar in intent to the back pointer field in the siTD data structure (see siTD Back Link Pointer).

This feature provides software a simple primitive to save a schedule position, redirect the host controller to traverse the necessary queue heads in the previous frame, then restore the original schedule position and complete normal traversal.

The four components to the use of FSTNs:

- FSTN data structure.
- A *Save Place* indicator. This is always an FSTN with its *Back Path Link Pointer.T-bit* set to zero.

- A *Restore* indicator. This is always an FSTN with its *Back Path Link Pointer.T-bit* set to one.
- Host controller FSTN traversal rules.

When the host controller encounters an FSTN during micro-frames 2 through 7 it simply follows the node's *Normal Path Link Pointer* to access the next schedule data structure.

NOTE

The FSTN's *Normal Path Link Pointer.T-bit* may set to one, which the host controller must interpret as the end of periodic list mark.

When the host controller encounters a *Save-Place* FSTN in micro-frames 0 or 1, it saves the value of the *Normal Path Link Pointer* and set an internal flag indicating that it is executing in *Recovery Path* mode. *Recovery Path* mode modifies the host controller's rules for how it traverses the schedule and limits which data structures is considered for execution of bus transactions. The host controller continues executing in *Recovery Path* mode until it encounters a *Restore* FSTN or it determines that it has reached the end of the micro-frame (see details in the list below).

The rules for schedule traversal and limited execution while in *Recovery Path* mode are:

- Always follow the *Normal Path Link Pointer* when it encounters an FSTN that is a *Save-Place* indicator. The host controller must not recursively follow *Save-Place* FSTNs. Therefore, while executing in *Recovery Path* mode, it must never follow an FSTN's *Back Path Link Pointer*.
- Do not process an siTD or, iTD data structure. Simply follow its *Next Link Pointer*.
- Do not process a QH (Queue Head) whose *EPS* field indicates a high-speed device. Simply follow its *Horizontal Link Pointer*.
- When a QH's *EPS* field indicates a Full/Low-speed device, the host controller considers only it for execution if its *SplitXState* is DoComplete (note: this applies whether the *PID Code* indicates an IN or an OUT). See Execute Transaction and Tracking Split Transaction Progress for Interrupt Transfers for a complete list of additional conditions that must be met in general for the host controller to issue a bus transaction.
  - The host controller must not execute a Start-split transaction while executing in *Recovery Path* mode. See Periodic Isochronous - Do Complete Split for special handling when in *Recovery Path* mode.
- Stop traversing the *recovery path* when it encounters an FSTN that is a *Restore* indicator. The host controller unconditionally uses the saved value of the *Save-Place* FSTN's *Normal Path Link Pointer* when returning to the normal path traversal. The

host controller must clear the context of executing a *Recovery Path* when it restores schedule traversal to the *Save-Place* FSTN's *Normal Path Link Pointer*.

- If the host controller determines that there is not enough time left in the micro-frame to complete processing of the periodic schedule, it abandons traversal of the recovery path, and clears the context of executing a recovery path. The result is that at the start of the next consecutive micro-frame, the host controller starts traversal at the frame list.

An example traversal of a periodic schedule that includes FSTNs is illustrated in the following figure.

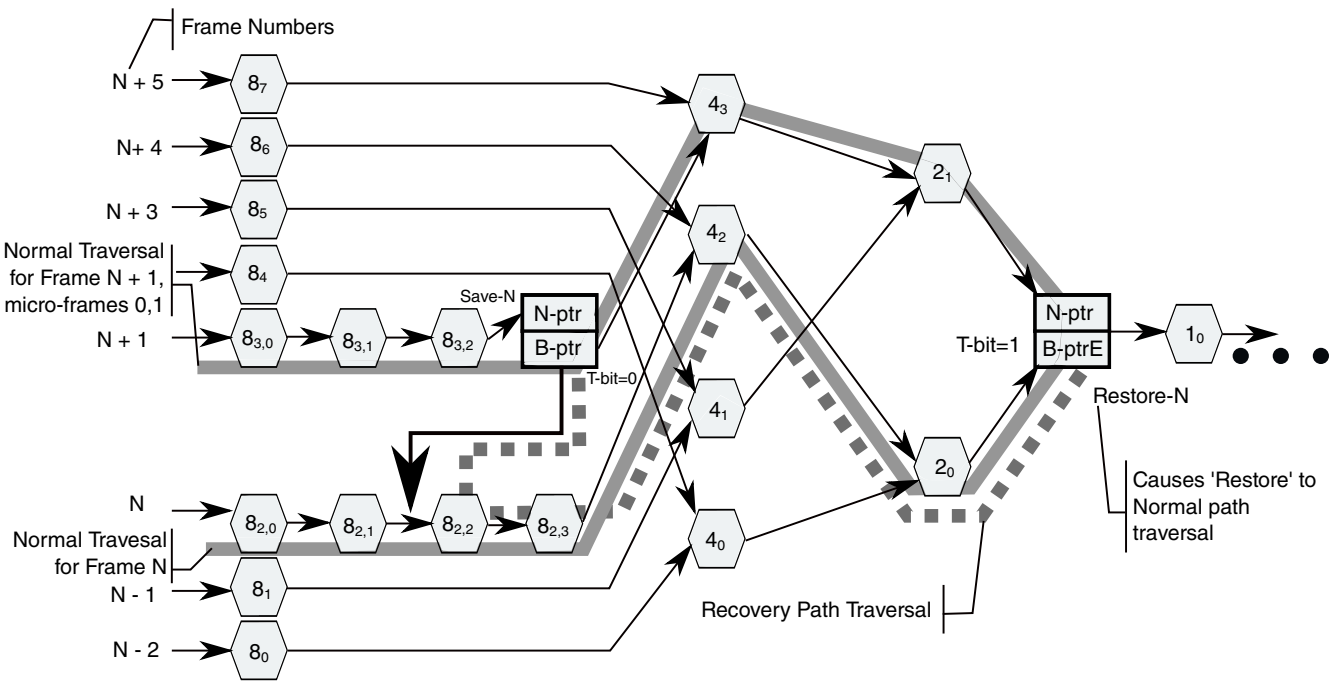


Figure 42-22. Example Host Controller Traversal of Recovery Path via FSTNs

In frame N+1 (micro-frames 0 and 1), when the host controller encounters Save-Path FSTN (Save-N), it observes that Save-N.Back Path Link Pointer.T-bit is zero (definition of a Save-Path indicator). The host controller saves the value of Save-N.Normal Path Link Pointer and follows Save-N.Back Path Link Pointer. At the same time, it sets an internal flag indicating that it is now in *Recovery Path* mode (the recovery path is annotated in the figure above with a large dashed line). The host controller continues traversing data structures on the recovery path and executing only those bus transactions as noted above, on the recovery path until it reaches Restore FSTN (Restore-N). Restore-N.Back Path Link Pointer.T-bit is set to one (definition of a Restore indicator), so the host controller exits *Recovery Path* mode by clearing the internal *Recovery Path* mode flag and commences (restores) schedule traversal using the saved value of the *Save-Place* FSTN's *Normal Path Link Pointer* (for example Save-N.Normal Path Link Pointer). The nodes traversed during these micro-frames include: {$8_{3.0}$, $8_{3.1}$, $8_{3.2}$, Save-A, $8_{2.2}$, $8_{2.3}$, 4_2,

2_0, Restore-N, 4_3, 2_1, Restore-N, 1_0 …}. The nodes on the recovery-path are in bold. In frame N (micro-frames 0-7), for this example, the host controller traverses all of the schedule data structures utilizing the *Normal Path Link Pointers* in any FSTNs it encounters. This is because the host controller has not yet encountered a *Save-Place* FSTN so it not executing in *Recovery Path* mode. When it encounters the *Restore* FSTN, (Restore-N), during micro-frames 0 and 1, it uses Restore-N.Normal Path Link Pointer to traverse to the next data structure (that is normal schedule traversal). This is because the host controller must use a Restore FSTN's *Normal Path Link Pointer* when not executing in a *Recovery-Path* mode. The nodes traversed during frame N include: {$8_{2.0}$, $8_{2.1}$, $8_{2.2}$, $8_{2.3}$, 4_2, 2_0, Restore-N, 1_0 …}.

In frame N+1 (micro-frames 2-7), when the host controller encounters Save-Path FSTN Save-N, it unconditionally follows Save-N.Normal Path Link Pointer. The nodes traversed during these micro-frames include: {$8_{3.0}$, $8_{3.1}$, $8_{3.2}$, Save-A, 4_3, 2_1, Restore-N, 1_0 …}.

42.5.3.12.2.3 Software Operational Model for FSTNs

Software must create a consistent, coherent schedule for the host controller to traverse.

When using FSTNs, system software must adhere to the following rules:

- Each *Save-Place* indicator requires a matching *Restore* indicator.
  - The *Save-Place* indicator is an FSTN with a valid *Back Path Link Pointer and T-bit* equal to zero.
    - *Back Path Link Pointer.Typ* field must be set to indicate the referenced data structure is a queue head. The *Restore* indicator is an FSTN with its *Back Path Link Pointer.T-bit* set to one.
  - A *Restore* FSTN may be matched to one or more *Save-Place* FSTNs. For example, if the schedule includes a poll-rate 1 level, then system software only needs to place a *Restore* FSTN at the beginning of this list in order to match all possible *Save-Place* FSTNs.
- If the schedule does not have elements linked at a poll-rate level of one, and one or more *Save-Place* FSTNs are used, then System Software must ensure the *Restore* FSTN's *Normal Path Link Pointer*'s *T-bit* is set to one, as this is used to mark the end of the periodic list.
- When the schedule does have elements linked at a poll rate level of one, a *Restore* FSTN must be the first data structure on the poll rate one list. All traversal paths from the frame list converge on the poll-rate one list. System software must ensure that *Recovery Path* mode is exited before the host controller is allowed to traverse the poll rate level one list.
- A *Save-Place* FSTN's *Back Path Link Pointer* must reference a queue head data structure. The referenced queue head must be reachable from the previous frame list

location. In other words, if the *Save-Place* FSTN is reachable from frame list offset N, then the FSTN's *Back Path Link Pointer* must reference a queue head that is reachable from frame list offset N-1.

Software should make the schedule as efficient as possible. What this means in this context is that software should have no more than one *Save-Place* FSTN reachable in any single frame. Note there is times when two (or more, depending on the implementation) could exist as full/low-speed footprints change with bandwidth adjustments. This could occur, for example when a bandwidth re-balance causes system software to move the *Save-Place* FSTN from one poll rate level to another. During the transition, software must preserve the integrity of the previous schedule until the new schedule is in place.

42.5.3.12.2.4 Tracking Split Transaction Progress for Interrupt Transfers

To correctly maintain the data stream, the host controller must be able to detect and report errors where data is lost.

For interrupt-IN transfers, data is lost when it makes it into the USB 2.0 hub, but the USB 2.0 host system is unable to get it from the USB 2.0 Hub and into the system before it expires from the transaction translator pipeline.

When a lost data condition is detected, the queue must be halted, thus signaling system software to recover from the error. A data-loss condition exists whenever a start-split is issued, accepted and successfully executed by the USB 2.0 Hub, but the complete-splits get unrecoverable errors on the high-speed link, or the complete-splits do not occur at the correct times. One reason complete-splits might not occur at the right time would be due to host-induced system hold-offs that cause the host controller to miss bus transactions because it cannot get timely access to the schedule in system memory.

The same condition can occur for an interrupt-OUT, but the result is not an endpoint halt condition, but rather effects only the progress of the transfer. The queue head has the following fields to track the progress of each split transaction. These fields are used to keep incremental state about which (and when) portions have been executed.

- *C-prog-mask*. This is an eight-bit bit-vector where the host controller keeps track of which complete-splits have been executed. Due to the nature of the Transaction Translator periodic pipeline, the complete-splits need to be executed in-order. The host controller needs to detect when the complete-splits have not been executed in order. This can only occur due to system hold-offs where the host controller cannot get to the memory-based schedule. *C-prog-mask* is a simple bit-vector that the host controller sets one of the *C-prog*-mask bits for each complete-split executed. The bit position is determined by the micro-frame number in which the complete-split was executed. The host controller always checks *C-prog-mask* before executing a

complete-split transaction. If the previous complete-splits have not been executed then it means one (or more) have been skipped and data has potentially been lost.
- *FrameTag*. This field is used by the host controller during the complete-split portion of the split transaction to tag the queue head with the frame number (*H-Frame* number) when the next complete split must be executed.
- *S-bytes*. This field can be used to store the number of data payload bytes sent during the start-split (if the transaction was an OUT). The *S-bytes* field must be used to accumulate the data payload bytes received during the complete-splits (for an IN).

42.5.3.12.2.5 Split Transaction Execution State Machine for Interrupt

In the following presentation, all references to micro-frame are in the context of a micro-frame within an *H-Frame.*

As with asynchronous Full- and Low-speed endpoints, a split-transaction state machine is used to manage the split transaction sequence.

Aside from the fields defined in the queue head for scheduling and tracking the split transaction, the host controller calculates one internal mechanism that is also used to manage the split transaction. The internal calculated mechanism is:

- *cMicroFrameBit* is a single-bit encoding of the current micro-frame number. It is an eight-bit value calculated by the host controller at the beginning of every micro-frame. It is calculated from the three least significant bits of the *FRINDEX* register (that is, *cMicroFrameBit* = (1 shifted-left(*FRINDEX*[2:0])). The *cMicroFrameBit* has at most one bit asserted, which always corresponds to the current micro-frame number. For example, if the current micro-frame is 0, then *cMicroFrameBit* will equal 00000001b. The variable *cMicroFrameBit* is used to compare against the *S-mask* and *C-mask* fields to determine whether the queue head is marked for a start- or complete-splt transaction for the current micro-frame.

The following figure illustrates the state machine for managing a complete interrupt split transaction. There are two phases to each split transaction. The first is a single start-split transaction, which occurs when the *SplitXState* is at Do_Start and the single bit in *cMicroFrameBit* has a corresponding bit active in *QH.S-mask*. The transaction translator does not acknowledge the receipt of the periodic start-split, so the host controller unconditionally transitions the state to Do_Complete. Due to the available jitter in the transaction translator pipeline, there will be more than one complete-split transaction scheduled by software for the Do_Complete state. This translates simply to the fact that there are multiple bits set to a one in the *QH.C-mask* field.

The host controller keeps the queue head in the Do_Complete state until the split transaction is complete (see definition below), or an error condition triggers the *three-strikes-rule* (for example, after the host tries the same transaction three times, and each

encounters an error, the host controller will stop retrying the bus transaction and halt the endpoint, thus requiring system software to detect the condition and perform system-dependent recovery).

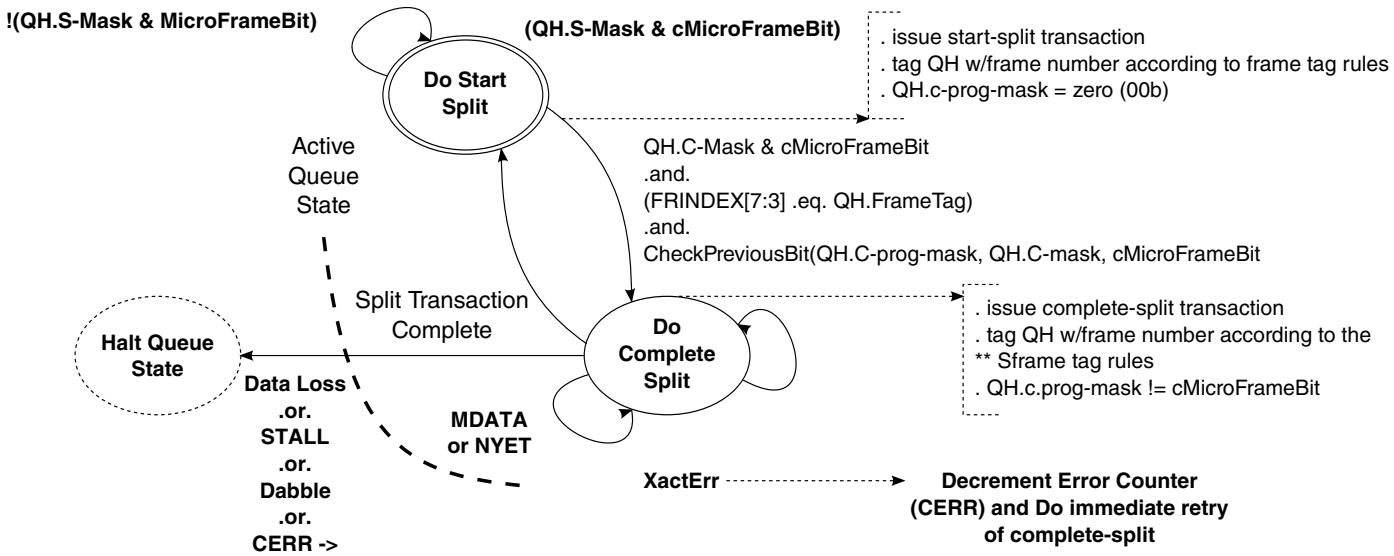


Figure 42-23. Split Transaction State Machine for Interrupt

See Previous Section for the frame tag management rules.

Periodic Interrupt - Do Start Split

This is the state software must initialize a full- or low-speed interrupt queue head *StartXState* bit. This state is entered from the Do_Complete Split state only after the split transaction is complete. This occurs when one of the following events occur: The transaction translator responds to a complete-split transaction with one of the following:

- NAK. A NAK response is a propagation of the full- or low-speed endpoint's NAK response.
- ACK. An ACK response is a propagation of the full- or low-speed endpoint's ACK response. Only occurs on an OUT endpoint.
- DATA 0/1. Only occurs for INs. Indicates that this is the last of the data from the endpoint for this split transaction.
- ERR. The transaction on the low-/full-speed link below the transaction translator had a failure (for example, timeout, bad CRC, etc.).
- NYET (and Last). The host controller issued the last complete-split and the transaction translator responded with a NYET handshake. This means that the start-split was not correctly received by the transaction translator, so it never executed a transaction to the full- or low-speed endpoint, see Section Periodic Isochronous - Do Complete Split for the definition of 'Last'.

Each time the host controller visits a queue head in this state (once within the Execute Transaction state), it performs the following test to determine whether to execute a start-split.

- *QH.S-mask* is bit-wise anded with *cMicroFrameBit*.

If the result is non-zero, then the host controller will issue a start-split transaction. If the *PIDCode* field indicates an IN transaction, the host controller must zero-out the *QH.S-bytes* field. After the split-transaction has been executed, the host controller sets up state in the queue head to track the progress of the complete-split phase of the split transaction. Specifically, it records the expected frame number into *QH.FrameTag* field (see Section), set *C-prog-mask* to zero (00h), and exits this state. Note that the host controller must not adjust the value of *CErr* as a result of completion of a start-split transaction.

Periodic Interrupt - Do Complete Split

This state is entered unconditionally from the Do Start Split state after a start-split transaction is executed on the bus. Each time the host controller visits a queue head in this state (once within the Execute Transaction state), it checks to determine whether a complete-split transaction should be executed now.

There are four tests to determine whether a complete-split transaction should be executed.

- Test A. *cMicroFrameBit* is bit-wise anded with *QH.C-mask* field. A non-zero result indicates that software scheduled a complete-split for this endpoint, during this micro-frame.
- Test B. *QH.FrameTag* is compared with the current contents of *FRINDEX*[7:3]. An equal indicates a match.
- Test C. The complete-split progress bit vector is checked to determine whether the previous bit is set, indicating that the previous complete-split was appropriately executed. An example algorithm for this test is provided below:

```
Algorithm Boolean CheckPreviousBit(QH.C-prog-mask, QH.C-mask, cMicroFrameBit)
Begin
-- Return values:
-- TRUE - no error
-- FALSE - error
--
Boolean rvalue = TRUE;
previousBit = cMicroframeBit logical-rotate-right(1)
-- Bit-wise anding previousBit with C-mask indicates
-- whether there was an intent
-- to send a complete split in the previous micro-frame. So,
-- if the
-- 'previous bit' is set in C-mask, check C-prog-mask to
-- make sure it
-- happened.
If (previousBit bitAND QH.C-mask)then
                                If not(previousBit bitAND QH.C-prog-mask) then
                      rvalue = FALSE;
          End if
End If
-- If the C-prog-mask already has a one in this bit position,
```

```
-- then an aliasing
-- error has occurred. It will probably get caught by the
-- FrameTag Test, but
-- at any rate it is an error condition that as detectable here
-- should not allow
-- a transaction to be executed.
If (cMicroFrameBit bitAND QH.C-prog-mask) then
 rvalue = FALSE;
End if
return (rvalue)
End Algorithm
```

- Test D. Check to see if a start-split should be executed in this micro-frame. Note this is the same test performed in the Do Start Split state (see Section Periodic Isochronous - Do Start Split). Whenever it evaluates to TRUE and the controller is NOT processing in the context of a *Recovery Path* mode, it means a start-split should occur in this micro-frame. Test D and Test A evaluating to TRUE at the same time is a system software error. Behavior is undefined.

If (A .and. B .and. C .and. not(D)) then the host controller will execute a complete-split transaction. When the host controller commits to executing the complete-split transaction, it updates *QH.C-prog-mask* by bit-ORing with *cMicroFrameBit*. On completion of the complete-split transaction, the host controller records the result of the transaction in the queue head and sets *QH.FrameTag* to the expected *H-Frame* number (see Section). The effect to the state of the queue head and thus the state of the transfer depends on the response by the transaction translator to the complete-split transaction. The following responses have the effects (note that any responses that result in decrementing of the *CErr* will result in the queue head being halted by the host controller if the result of the decrement is zero):

- NYET (and Last). On each NYET response, the host controller checks to determine whether this is the last complete-split for this split transaction. Last is defined in this context as the condition where all of the scheduled complete-splits have been executed. If it is the last complete-split (with a NYET response), then the transfer state of the queue head is not advanced (never received any data) and this state exited. The transaction translator must have responded to all the clompete-splits with NYETs, meaning that the start-split issued by the host controller was not received. The start-split should be retried at the next poll period.
- The test for whether this is the Last complete split can be performed by XOR *QH.C-mask* with *QH.C-prog-mask*. If the result is all zeros then all complete-splits have been executed. When this condition occurs, the *XactErr* status bit is set to a one and the *CErr* field is decremented.
- NYET (and not Last). See above description for testing for Last. The complete-split transaction received a NYET response from the transaction translator. Do not update any transfer state (except for *C-prog-mask* and *FrameTag*) and stay in this state. The host controller must not adjust *CErr* on this response.

- Transaction Error (XactErr). Timeout, data CRC failure, etc. The *CErr* field is decremented and the *XactErr* bit in the *Status* field is set to a one. The complete split transaction is *immediately* retried (if *Cerr* is non-zero).If there is not enough time in the micro-frame to complete the retry and the endpoint is an IN, or *CErr* is decremented to a zero from a one, the queue is halted. If there is not enough time in the micro-frame to complete the retry and the endpoint is an OUT and *CErr* is not zero, then this state is exited (that is, return to Do Start Split). This results in a retry of the entire OUT split transaction, at the next poll period. Refer to Chapter 11 Hubs (specifically the section full- and low-speed Interrupts) in the USB Specification Revision 2.0 for detailed requirements on why these errors must be immediately retried.
- ACK. This can only occur if the target endpoint is an OUT. The target endpoint ACK'd the data and this response is a propagation of the endpoint ACK up to the host controller. The host controller must advance the state of the transfer. The *Current Offset* field is incremented by *Maximum Packet Length* or *Bytes to Transfer*, whichever is less. The field *Bytes To Transfer* is decremented by the same amount. And the data toggle bit (*dt*) is toggled. The host controller will then exit this state for this queue head. The host controller must reload *CErr* with maximum value on this response. Advancing the transfer state may cause other process events such as retirement of the qTD and advancement of the queue (see Section Managing Control/Bulk/Interrupt Transfers through Queue Heads).
- MDATA. This response will only occur for an IN endpoint. The transaction translator responded with zero or more bytes of data and an MDATA PID. The incremental number of bytes received is accumulated in *QH.S-bytes*. The host controller must not adjust *CErr* on this response.
- DATA0/1. This response may only occur for an IN endpoint. The number of bytes received is added to the accumulated byte count in *QH.S-bytes*. The state of the transfer is advanced by the result and the host controller will exit this state for this queue head.
- Advancing the transfer state may cause other processing events such as retirement of the qTD and advancement of the queue (see Section Managing Control/Bulk/Interrupt Transfers through Queue Heads).
- If the data sequence PID does not match the expected, the entirety of the data received in this split transaction is ignored, the transfer state is not advanced and this state is exited.
- NAK. The target endpoint Nak'd the full- or low-speed transaction. The state of the transfer is not advanced, and this state is exited. The host controller must reload *CErr* with maximum value on this response.

- ERR. There was an error during the full- or low-speed transaction. The ERR status bit is set to a one, *Cerr* is decremented, the state of the transfer is not advanced, and this state is exited.
- STALL. The queue is halted (an exit condition of the Execute Transaction state). The status field bits: *Active* bit is set to zero and the *Halted* bit is set to a one and the qTD is retired. Responses which are not enumerated in the list or which are received out of sequence are illegal and may result in undefined host controller behavior. The other possible combinations of tests A, B, C, and D may indicate that data or response was lost. The table below lists the possible combinations and the appropriate action.

Table 42-48. Interrupt IN/OUT Do Complete Split State Execution Criteria

Condition	Action	Description
not(A) not(D)	Ignore QHD	Neither a start nor complete-split is scheduled for the current micro-frame.Host controller should continue walking the schedule.
A not(C)	If PIDCode = IN Halt QHD If PIDCode = OUT Retry start-split	Progress bit check failed. These means a complete-split has been missed. There is the possibility of lost data. If *PIDCode* is an IN, then the Queue head must be halted. If *PIDCode* is an OUT, then the transfer state is not advanced and the state exited (for example, start-split is retried). This is a host-induced error and does not effect *CER*R. In either case, set the *Missed Micro-frame* bit in the status field to a one.
A not(B) C	If PIDCode = IN Halt QHD If PIDCode = OUT Retry start-split	*QH.FrameTag* test failed. This means that exactly one or more *H-Frames* have been skipped. This means complete-splits and have missed. There is the possibility of lost data. If *PIDCode* is an IN, then the Queue head must be halted. If *PIDCode* is an OUT, then the transfer state is not advanced and the state exited (for example, start-split is retried). This is a host-induced error and does not effect *CER*R. In either case, set the *Missed Micro-frame* bit in the status field to a one.
A B C not(D)	Execute complete-split	This is the non-error case where the host controller executes a complete-split transaction.
D	If PIDCode = IN Halt QHD If PIDCode = OUT Retry start-split	This is a degenerate case where the start-split was issued, but all of the complete-splits were skipped and all possible intervening opportunities to detect the missed data failed to fire. If *PIDCode* is an IN, then the Queue head must be halted. If *PIDCode* is an OUT, then the transfer state is not advanced and the state exited (for example, start-split is retried). This is a host-induced error and does not effect *CER*R. In either case, set the *Missed Micro-frame* bit in the status field to a one. Note: When executing in the context of a *Recovery Path* mode, the host controller is allowed to process the queue head and take the actions indicated above, or it may wait until the queue head is visited in the

Table 42-48. Interrupt IN/OUT Do Complete Split State Execution Criteria

		normal processing mode. Regardless, the host controller must not execute a start-split in the context of a executing in a *Recovery Path* mode.

Managing QH.FrameTag Field

The *QH.FrameTag* field in a queue head is completely managed by the host controller. The rules for setting *QH.FrameTag* are simple:

- Rule 1: If transitioning from Do Start Split to Do Complete Split and the current value of *FRINDEX*[2:0] is 6 *QH.FrameTag* is set to *FRINDEX*[7:3] + 1. This accommodates split transactions whose start-split and complete-splits are in different *H-Frames* (case 2a, see Figure 42-20).
- Rule 2: If the current value of *FRINDEX*[2:0] is 7, *QH.FrameTag* is set to *FRINDEX*[7:3] + 1. This accommodates staying in Do Complete Split for cases 2a, 2b, and 2c (Figure 42-20).
- Rule 3: If transitioning from Do_Start Split to Do Complete Split and the current value of *FRINDEX*[2:0] is not 6, or currently in Do Complete Split and the current value of (*FRINDEX*[2:0]) is not 7, *FrameTag* is set to *FRINDEX*[7:3]. This accommodates all other cases (Figure 42-20).

42.5.3.12.2.6 Rebalancing the periodic schedule

System software must occasionally adjust a periodic queue head's S-mask and C-mask fields during operation.

This need occurs when adjustments to the periodic schedule create a new bandwidth budget and one or more queue head's are assigned new execution footprints (that is, new S-mask and C-mask values).

It is imperative that System software must not update these masks to new values in the midst of a split transaction. In order to avoid any race conditions with the update, the EHCI host controller provides a simple assist to system software. System software sets the *Inactivate-on-next-Transaction* (*I*) bit to a one to signal the host controller that it intends to update the S-mask and C-mask on this queue head. System software will then wait for the host controller to observe the *I-bit* is a one and transition the *Active* bit to a zero. The rules for how and when the host controller sets the *Active* bit to zero are enumerated below:

- If the *Active* bit is a zero, no action is taken. The host controller does not attempt to advance the queue when the *I-bit* is a one.
- If the *Active* bit is a one and the *SplitXState* is DoStart (regardless of the value of *S-mask*), the host controller will simply set *Active* bit to a zero. The host controller is

not required to write the transfer state back to the *current* qTD. Note that if the *S-mask* indicates that a start-split is scheduled for the current micro-frame, the host controller must not issue the start-split bus transaction. It must set the *Active* bit to zero.

System software must save transfer state before setting the *I-bit* to a one. This is required so that it can correctly determine what transfer progress (if any) occurred after the *I-bit* was set to a one and the host controller executed its final bus-transaction and set *Active* to a zero.

After system software has updated the S-mask and C-mask, it must then reactivate the queue head. Because the *Active* bit and the *I-bit* cannot be updated with the same write, system software needs to use the following algorithm to coherently re-activate a queue head that has been stopped via the *I-bit*.

1. Set the *Halted* bit to a one, then
2. Set the *I-bit* to a zero, then
3. Set the *Active* bit to a one and the *Halted* bit to a zero in the same write.

Setting the *Halted* bit to a one inhibits the host controller from attempting to advance the queue between the time the *I-bit* goes to a zero and the *Active* bit goes to a one.

42.5.3.12.3 Split Transaction Isochronous

Full-speed isochronous transfers are managed using the split-transaction protocol through a USB 2.0 transaction translator in a USB2.0 Hub. The EHCI controller utilizes siTD data structure to support the special requirements of isochronous split-transactions.

This data structure uses the scheduling model of isochronous TDs (iTD, Section Isochronous (High-Speed) Transfer Descriptor (iTD)) (see Section Managing Isochronous Transfers Using iTDs for the operational model of iTDs) with the contiguous data feature provided by queue heads. This simple arrangement allows a single isochronous scheduling model and adds the additional feature that all data received from the endpoint (per split transaction) must land into a contiguous buffer.

42.5.3.12.3.1 Split Transaction Scheduling Mechanisms for Isochronous

Full-speed isochronous transactions are managed through a transaction translator's periodic pipeline. As with full- and low-speed interrupt, system software manages each transaction translator's periodic pipeline by budgeting and scheduling exactly during which micro-frames the start-splits and complete-splits for each full-speed isochronous endpoint occur.

The requirements described in Section Split Transaction Scheduling Mechanisms for Interrupt apply. The following figure illustrates the general scheduling boundary conditions that are supported by the EHCI periodic schedule. The ^{S}X and ^{C}X labels indicate micro-frames where software can schedule start- and complete-splits (respectively). The *H-Frame* boundaries are marked with a large, solid bold vertical line. The *B-Frame* boundaries are marked with a large, bold, dashed line. The bottom of the figure illustrates the relationship of an siTD to the *H-Frame*.

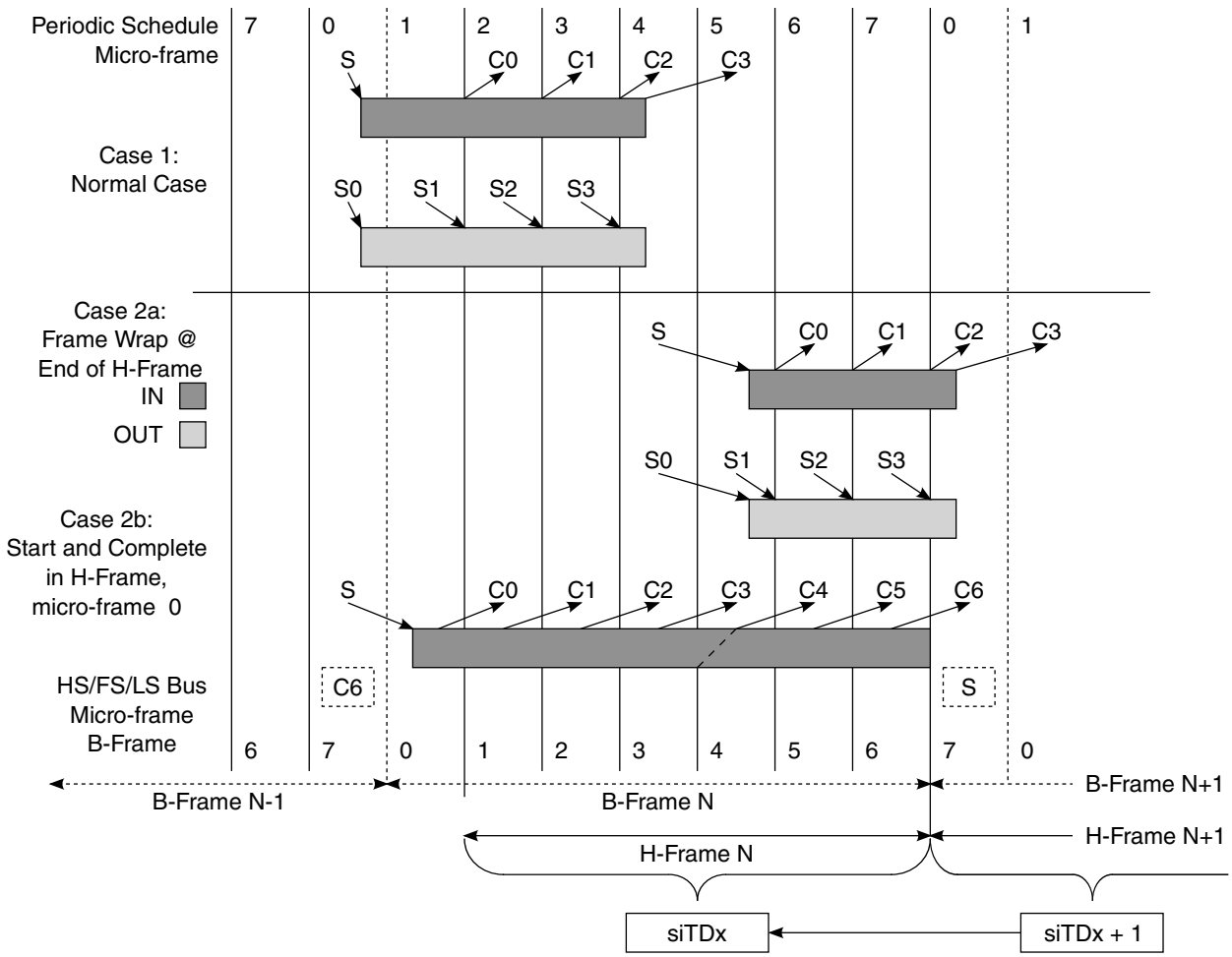


Figure 42-24. Split Transaction, Isochronous Scheduling Boundary Conditions

When the endpoint is an isochronous OUT, there are only start-splits, and no complete-splits. When the endpoint is an isochronous IN, there is at most one start-split and one to *N* complete-splits. The scheduling boundary cases are:

- *Case 1*: The entire split transaction is completely bounded by an *H-Frame*. For example: the start-splits and complete-splits are all scheduled to occur in the same *H-Frame*.

- *Case 2a*: This boundary case is where one or more (at most two) complete-splits of a split transaction IN are scheduled across an *H-Frame* boundary. This can only occur when the split transaction has the possibility of moving data in *B-Frame,* micro-frames 6 or 7 (*H-Frame* micro-frame 7 or 0). When an *H-Frame* boundary wrap condition occurs, the scheduling of the split transaction spans more than one location in the periodic list.(For example, it takes two siTDs in adjacent periodic frame list locations to fully describe the scheduling for the split transaction.)
- Although the scheduling of the split transaction may take two data structures, all of the complete-splits for each full-speed IN isochronous transaction must use only one data pointer. For this reason, siTDs contain a back pointer, the use of which is described below.
- Software must never schedule full-speed isochronous OUTs across an *H-Frame* boundary.
- *Case 2b*: This case can only occur for a very large isochronous IN. It is the only allowed scenario where a start-split and complete-split for the same endpoint can occur in the same micro-frame. Software must enforce this rule by scheduling the large transaction first. Large is defined to be anything larger than 579 byte maximum packet size.

A subset of the same mechanisms employed by full- and low-speed interrupt queue heads are employed in siTDs to schedule and track the portions of isochronous split transactions. The following fields are initialized by system software to instruct the host controller when to execute portions of the split transaction protocol.

- *SplitXState*. This is a single bit residing in the *Status* field of an siTD (see Figure 42-25). This bit is used to track the current state of the split transaction. The rules for managing this bit are described in Section Split Transaction Execution State Machine for Interrupt.
- *Frame S-mask*. This is a bit-field where-in system software sets a bit corresponding to the micro-frame (within an *H-Frame*) that the host controller should execute a start-split transaction. This is always qualified by the value of the *SplitXState* bit. For example, referring to the IN example in Figure 42-24, case one, the *S-mask* would have a value of 00000001b indicating that if the siTD is traversed by the host controller, and the *SplitXState* indicates Do Start Split, and the current micro-frame as indicated by USB_n_FRINDEX[2:0] is 0, then execute a start-split transaction.
- *Frame C-mask.* This is a bit-field where system software sets one or more bits corresponding to the micro-frames (within an *H-Frame*) that the host controller should execute complete-split transactions. The interpretation of this field is always qualified by the value of the *SplitXState* bit. For example, referring to the IN example in Figure 42-24, case one, the *C-mask* would have a value of 00111100b indicating that if the siTD is traversed by the host controller, and the *SplitXState* indicates Do

Complete Split, and the current micro-frame as indicated by *USB_n_FRINDEX*[2:0] is 2, 3, 4, or 5, then execute a complete-split transaction.

- *Back Pointer*. This field in a siTD is used to complete an IN split-transaction using the previous *H-Frame*'s siTD. This is only used when the scheduling of the complete-splits span an *H-Frame* boundary.

There exists a one-to-one relationship between a high-speed isochronous split transaction (including all start- and complete-splits) and one full-speed isochronous transaction. An siTD contains (amongst other things) buffer state and split transaction scheduling information. An siTD's buffer state always maps to one full-speed isochronous data payload. This means that for any full-speed transaction payload, a single siTD's data buffer must be used. This rule applies to both IN an OUTs. An siTD's scheduling information usually also maps to one high-speed isochronous split transaction. The exception to this rule is the *H-Frame* boundary wrap cases mentioned above.

The siTD data structure describes at most, one frame's worth of high-speed transactions and that description is strictly bounded within a frame boundary. The figure below illustrates some examples. On the top are examples of the full-speed transaction footprints for the boundary scheduling cases described above. In the middle are time-frame references for both the *B-Frames* (HS/FS/LS Bus) and the *H-Frames*. On the bottom is illustrated the relationship between the scope of an siTD description and the time references. Each *H-Frame* corresponds to a single location in the periodic frame list. The implication is that each siTD is reachable from a single periodic frame list location at a time.

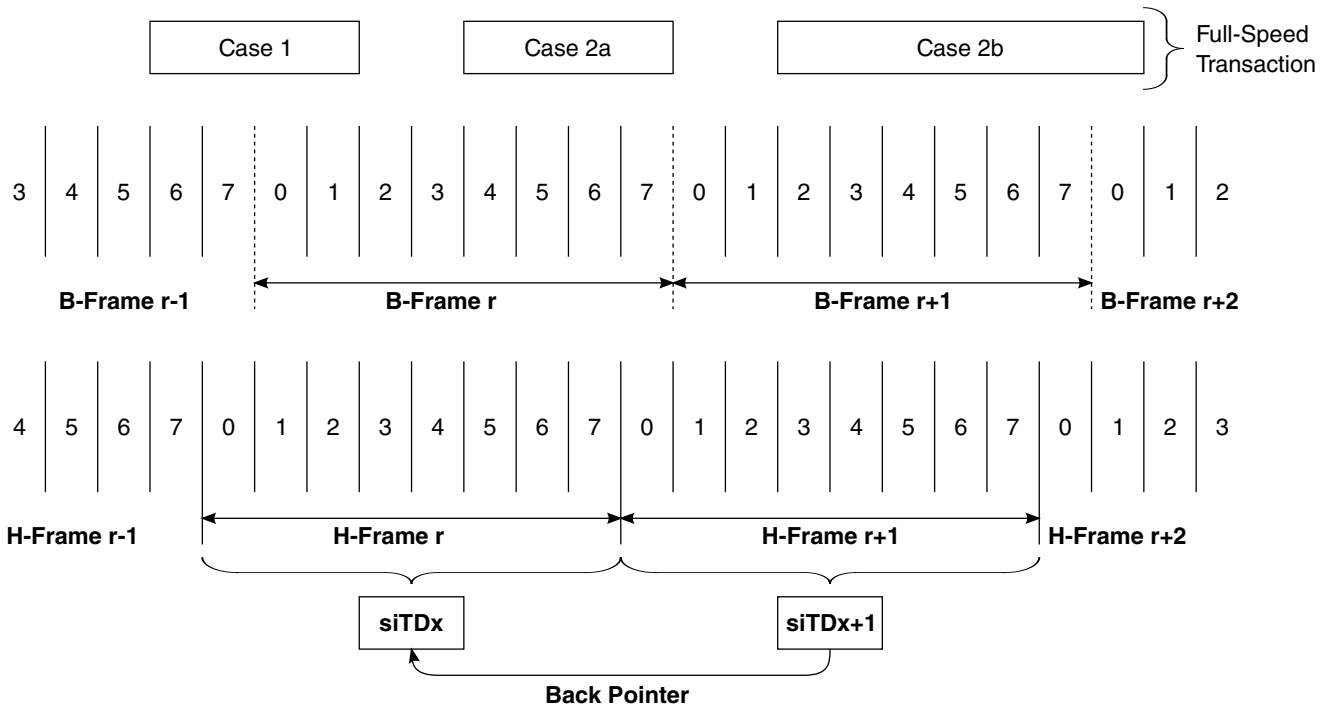


Figure 42-25. siTD Scheduling Boundary Examples

Each case is described below:

- *Case 1*: One siTD is sufficient to describe and complete the isochronous split transaction because the whole isochronous split transaction is tightly contained within a single *H-Frame*.
- *Case 2a, 2b*: Although both INs and OUTs can have these footprints, OUTs always take only one siTD to schedule. However, INs (for these boundary cases) require two siTDs to complete the scheduling of the isochronous split transaction $siTD_X$ is used to always issue the start-split and the first *N* complete-splits. The full-speed transaction (for these cases) can deliver data on the full-speed bus segment during micro-frame 7 of $H\text{-}Frame_{Y+1}$, or micro-frame 0 of $H\text{-}Frame_{Y+2}$. The complete splits are scheduled using $siTD_{X+2}$ (not shown). The complete-splits to extract this data must use the buffer pointer from $siTD_{X+1}$. The only way for the host controller to reach $siTD_{X+1}$ from $H\text{-}Frame_{Y+2}$is to use $siTD_{X+2}$'s back pointer. The host controller rules for when to use the back pointer are described is Section Periodic Isochronous - Do Complete Split .

Software must apply the following rules when calculating the schedule and linking the schedule data structures into the periodic schedule:

- Software must ensure that an isochronous split-transaction is started so that it will complete before the end of the *B-Frame*.
- Software must ensure that for a single full-speed isochronous endpoint, there is never a start-split and complete-split in *H-Frame, micro-frame 1*. This is mandated as a rule so that case 2a and case 2b can be discriminated. According to the core USB specification, the long isochronous transaction illustrated in Case 2b, could be scheduled so that the start-split was in micro-frame 1 of *H-Frame* N and the last complete-split would need to occur in micro-frame 1 of H-Frame N+1. However, it is impossible to discriminate between cases 2a and case 2b, which has significant impact on the complexity of the host controller.

42.5.3.12.3.2 Tracking Split Transaction Progress for Isochronous Transfers

To correctly maintain the data stream, the host controller must be able to detect and report errors where device to host data is lost. Isochronous endpoints do not employ the concept of a halt on error, however the host is required to identify and report per-packet errors observed in the data stream. This includes schedule traversal problems (skipped micro-frames), timeouts and corrupted data received.

In similar kind to interrupt split-transactions, the portions of the split transaction protocol must execute in the micro-frames they are scheduled. The queue head data structure used to manage full- and low-speed interrupt has several mechanisms for tracking when portions of a transaction have occurred. Isochronous transfers use siTDs, for their transfers, and the data structures are only reachable via the schedule in the exact micro-frame in which they are required (so all the mechanism employed for tracking in queue heads is not required for siTDs). Software has the option of reusing siTD several times in the complete periodic schedule. However, it must ensure that the results of split transaction N are consumed and the siTD reinitialized (activated) before the host controller gets back to the siTD (in a future micro-frame).

Split-transaction isochronous OUTs utilize a low-level protocol to indicate which portions of the split transaction data have arrived. Control over the low-level protocol is exposed in an siTD via the fields *Transaction Position* (*TP*) and *Transaction Count* (*T-count*). If the entire data payload for the OUT split transaction is larger than 188 bytes, there will be more than one start-split transaction, each of which require proper annotation. If host hold-offs occur, then the sequence of annotations received from the host will not be complete, which is detected and handled by the transaction translator. See Section Periodic Isochronous - Do Start Split for a description on how these fields are used during a sequence of start-split transactions.

The fields *siTD.T-Count* and *siTD.TP* are used by the host controller to drive and sequence the transaction position annotations. It is the responsibility of system software to properly initialize these fields in each siTD. Once the budget for a split-transaction

isochronous endpoint is established, *S-mask*, *T-Count*, and *TP* initialization values for all the siTD associated with the endpoint are constant. They remain constant until the budget for the endpoint is recalculated by software and the periodic schedule adjusted.

For IN-endpoints, the transaction translator simply annotates the response data packets with enough information to allow the host controller to identify the last data. As with split transaction Interrupt, it is the host controller's responsibility to detect when it has missed an opportunity to execute a complete-split. The following field in the siTD is used to track and detect errors in the execution of a split transaction for an IN isochronous endpoint.

- *C-prog-mask*. This is an eight-bit bit-vector where the host controller keeps track of which complete-splits have been executed. Due to the nature of the Transaction Translator periodic pipeline, the complete-splits need to be executed in-order. The host controller needs to detect when the complete-splits have not been executed in order. This can only occur due to system hold-offs where the host controller cannot get to the memory-based schedule. *C-prog-mask* is a simple bit-vector that the host controller sets a bit for each complete-split executed. The bit position is determined by the micro-frame (USB_n_FRINDEX[2:0]) number in which the complete-split was executed. The host controller always checks *C-prog-mask* before executing a complete-split transaction. If the previous complete-splits have not been executed, then it means one (or more) have been skipped and data has potentially been lost. System software is required to initialize this field to zero before setting an siTD's *Active* bit to a one.

If a transaction translator returns with the final data before all of the complete-splits have been executed, the state of the transfer is advanced so that the remaining complete-splits are not executed. Refer to Section Asynchronous - Do Complete Split for a description on how the state of the transfer is advanced. It is important to note that an IN siTD is retired based solely on the responses from the Transaction Translator to the complete-split transactions. This means, for example, that it is possible for a transaction translator to respond to a complete-split with an MDATA PID. The number of bytes in the MDATA's data payload could cause the siTD field *Total Bytes to Transfer* to decrement to zero. This response can occur, before all of the scheduled complete-splits have been executed. In other interface, data structures (for example, high-speed data streams through queue heads), the transition of *Total Bytes to Transfer* to zero signals the end of the transfer and results in setting of the *Active* bit to zero. However, in this case, the result has not been delivered by the Transaction Translator and the host must continue with the next complete-split transaction to extract the residual transaction state. This scenario occurs because of the pipeline rules for a Transaction Translator (see Chapter 11 of the Universal Serial Bus Revision 2.0). In summary the periodic pipeline rules require that on a micro-frame boundary, the Transaction Translator will hold the final two bytes received (if it has not seen an End Of Packet (EOP)) in the full-speed bus pipe stage and give the

remaining bytes to the high-speed pipeline stage. At the micro-frame boundary, the Transaction Translator could have received the entire packet (including both CRC bytes) but not received the packet EOP. In the next micro-frame, the Transaction Translator will respond with an MDATA and send all of the data bytes (with the two CRC bytes being held in the full-speed pipeline stage). This could cause the siTD to decrement its *Total Bytes to Transfer* field to zero, indicating it has received all expected data. The host must still execute one more (scheduled) complete-split transaction in order to extract the results of the full-speed transaction from the Transaction Translator (for example, the Transaction Translator may have detected a CRC failure, and this result must be forwarded to the host).

If the host experiences hold-offs that cause the host controller to skip one or more (but not all) scheduled split transactions for an isochronous OUT, then the protocol to the transaction translator will not be consistent and the transaction translator will detect and react to the problem. Likewise, for host hold-offs that cause the host controller to skip one or more (but not all) scheduled split transactions for an isochronous IN, the *C-prog-mask* is used by the host controller to detect errors. However, if the host experiences a hold-off that causes it to skip all of an siTD, or an siTD expires during a host hold off (for example, a hold-off occurs and the siTD is no longer reachable by the host controller in order for it to report the hold-off event), then system software must detect that the siTDs have not been processed by the host controller (that is, state not advanced) and report the appropriate error to the client driver.

42.5.3.12.3.3 Split Transaction Execution State Machine for Isochronous

In the following presentation, all references to micro-frame are in the context of a micro-frame within an *H-Frame.*

If the *Active* bit in the *Status* byte is a zero, the host controller will ignore the siTD and continue traversing the periodic schedule. Otherwise the host controller will process the siTD as specified below. A split transaction state machine is used to manage the split-transaction protocol sequence. The host controller uses the fields defined in Section Tracking Split Transaction Progress for Interrupt Transfers , plus the variable *cMicroFrameBit* defined in Section Split Transaction Execution State Machine for Interrupt to track the progress of an isochronous split transaction. The figure below illustrates the state machine for managing an siTD through an isochronous split transaction. Bold, dotted circles denote the state of the *Active* bit in the *Status* field of a siTD. The Bold, dotted arcs denote the transitions between these states. Solid circles denote the states of the split transaction state machine and the solid arcs denote the transitions between these states. Dotted arcs and boxes reference actions that take place either as a result of a transition or from being in a state.

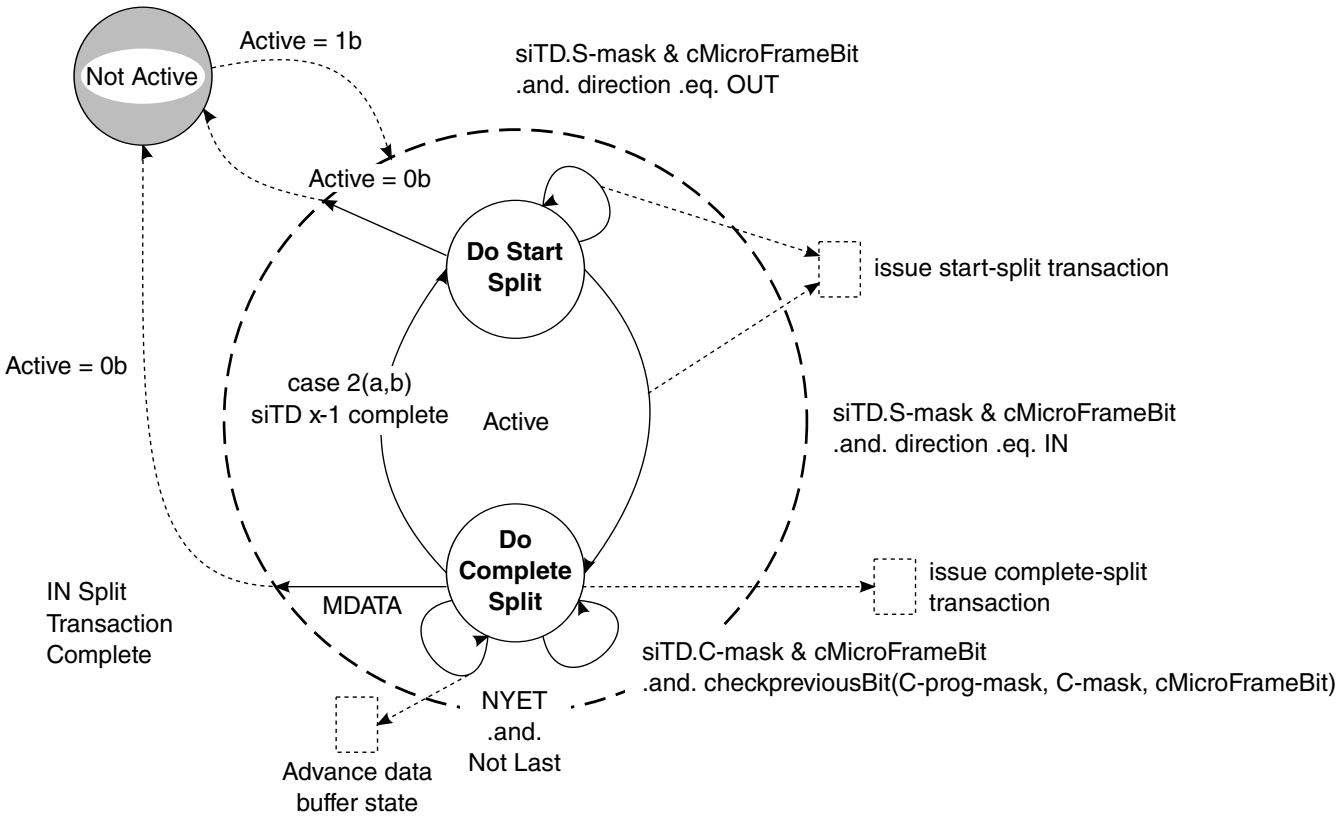


Figure 42-26. Split Transaction State Machine for Isochronous

42.5.3.12.3.4 Periodic Isochronous - Do Start Split

Isochronous split transaction OUTs use only this state.

An siTD for a split-transaction isochronous IN is either initialized to this state, or the siTD transitions to this state from Do Complete Split when a case 2a (IN) or 2b scheduling boundary isochronous split-transaction completes.

Each time the host controller reaches an active siTD in this state, it checks the *siTD.S-mask* against *cMicroFrameBit*. If there is a one in the appropriate position, the siTD will execute a start-split transaction. By definition, the host controller cannot *reach* an siTD at the wrong time. If the *I/O* field indicates an IN, then the start-split transaction includes only the extended token plus the full-speed token. Software must initialize the *siTD.Total Bytes To Transfer* field to the number of bytes expected. This is usually the maximum packet size for the full-speed endpoint. The host controller exits this state when the start-split transaction is complete.

The remainder of this section is specific to an isochronous OUT endpoint (that is, the *I/O* field indicates an OUT). When the host controller executes a start-split transaction for an isochronous OUT it includes a data payload in the start-split transaction. The memory

buffer address for the data payload is constructed by concatenating *siTD.Current Offset* with the page pointer indicated by the page selector field (*siTD.P*). A zero in this field selects Page 0 and a 1 selects Page 1. During the start-split for an OUT, if the data transfer crosses a page boundary during the transaction, the host controller must detect the page cross, update the *siTD.P*-bit from a zero to a one, and begin using the *siTD.Page 1* with *siTD.Current Offset* as the memory address pointer. The field *siTD.TP* is used to annotate each start-split transaction with the indication of which part of the split-transaction data the current payload represents (ALL, BEGIN, MID, END). In all cases the host controller simply uses the value in *siTD.TP* to mark the start-split with the correct transaction position code.

T-Count is always initialized to the number of start-splits for the current frame. *TP* is always initialized to the first required transaction position identifier. The scheduling boundary case (see Figure 42-25) is used to determine the initial value of *TP*. The initial cases are summarized in the following table.

Table 42-49. Initial Conditions for OUT siTD's TP and T-count Fields

Case	T-count	TP	Description
1, 2a	=1	ALL	When the OUT data payload is less than (or equal to) 188 bytes, only one start-split is required to move the data. The one start-split must be marked with an ALL.
1, 2a	!=1	BEGIN	When the OUT data payload is greater than 188 bytes more than one start-split must be used to move the data. The initial start-split must be marked with a BEGIN.

After each start-split transaction is complete, the host controller updates *T-Count* and *TP* appropriately so that the next start-split is correctly annotated.

The table below illustrates all of the *TP* and *T-count* transitions, which must be accomplished by the host controller.

.

Table 42-50. Transaction Position (TP)/Transaction Count (T-Count) Transition Table

TP	T-count next	TP next	Description
ALL	0	N/A	Transition from ALL, to done.
BEGIN	1	END	Transition from BEGIN to END. Occurs when *T-count* starts at 2.
BEGIN	!=1	MID	Transition from BEGIN to MID. Occurs when *T-count* starts at greater than 2.
MID	!=1	MID	*TP* stays at MID while *T-count* is not equal to 1 (that is, greater than 1). This case can occur for any of the scheduling boundary cases where the *T-count* starts greater than 3.
MID	1	END	Transition from MID to END. This case can occur for any of the scheduling boundary cases where the *T-count* starts greater than 2.

The start-split transactions do not receive a handshake from the transaction translator, so the host controller always advances the transfer state in the siTD after the bus transaction is complete. To advance the transfer state the following operations take place:

- The *siTD.Total Bytes To Transfer* and the *siTD.Current Offset* fields are adjusted to reflect the number of bytes transferred.
- The *siTD.P* (page selector) bit is updated appropriately.
- The *siTD.TP* and *siTD.T-count* fields are updated appropriately as defined in Table 42-50.

These fields are then written back to the memory based siTD. The *S-mask* is fixed for the life of the current budget. As mentioned above, *TP* and *T-count* are set specifically in each siTD to reflect the data to be sent from this siTD. Therefore, regardless of the value of *S-mask*, the actual number of start-split transactions depends on *T-count* (or equivalently, *Total Bytes to Transfer*). The host controller must set the *Active* bit to a zero when it detects that all of the schedule data has been sent to the bus. The preferred method is to detect when *T-Count* decrements to zero as a result of a start-split bus transaction. Equivalently, the host controller can detect when *Total Bytes to Transfer* decrements to zero. Either implementation must ensure that if the initial condition is *Total Bytes to Transfer* equal to zero and *T-count* is equal to a one, then the host controller will issue a single start-split, with a zero-length data payload. Software must ensure that *TP*, *T-count* and *Total Bytes to Transfer* are set to deliver the appropriate number of bus transactions from each siTD. An inconsistent combination will yield undefined behavior.

If the host experiences hold-offs that cause the host controller to skip start-split transactions for an OUT transfer, the state of the transfer will not progress appropriately. The transaction translator will observe protocol violations in the arrival of the start-splits for the OUT endpoint (that is, the transaction position annotation will be incorrect as received by the transaction translator).

Example scenarios are described in Section Split Transaction for Isochronous - Processing Examples.

A host controller implementation can optionally track the progress of an OUT split transaction by setting appropriate bits in the *siTD.C-prog-mask* as it executes each scheduled start-split. The *checkPreviousBit*() algorithm defined in Periodic Isochronous - Do Complete Split can be used prior to executing each start-split to determine whether start-splits were skipped. The host controller can use this mechanism to detect missed micro-frames. It can then set the siTD's *Active* bit to zero and stop execution of this siTD. This saves on both memory and high-speed bus bandwidth.

42.5.3.12.3.5 Periodic Isochronous - Do Complete Split

This state is only used by a split-transaction isochronous IN endpoint.

This state is entered unconditionally from the Do Start State after a start-split transaction is executed for an IN endpoint. Each time the host controller visits an siTD in this state, it conducts a number of tests to determine whether it should execute a complete-split transaction. The individual tests are listed below. The sequence they are applied depends on which micro-frame the host controller is currently executing which means that the tests might not be applied until after the siTD referenced from the back pointer has been fetched.

- Test A. *cMicroFrameBit* is bit-wise anded with *siTD.C-mask* field. A non-zero result indicates that software scheduled a complete-split for this endpoint, during this micro-frame. This test is always applied to a newly fetched siTD that is in this state.
- Test B. The *siTD.C-prog-mask* bit vector is checked to determine whether the previous complete splits have been executed. An example algorithm is below (this is slightly different than the algorithm used in Section Periodic Isochronous - Do Complete Split). The sequence in which this test is applied depends on the current value of USB_n_FRINDEX[2:0]. If USB_n_FRINDEX[2:0] is 0 or 1, it is not applied until the back pointer has been used. Otherwise it is applied immediately.

```
Algorithm Boolean CheckPreviousBit(siTD.C-prog-mask, siTD.C-mask, cMicroFrameBit)
Begin
          Boolean rvalue = TRUE;
          previousBit = cMicroFrameBit rotate-right(1)
          -- Bit-wise anding previousBit with C-mask indicates whether there was an intent
          -- to send a complete split in the previous micro-frame. So, if the
          -- 'previous bit' is set in C-mask, check C-prog-mask to make sure it
          -- happened.
          if previousBit bitAND siTD.C-mask then
                    if not (previousBit bitAND siTD.C-prog-mask) then
                              rvalue = FALSE
                    End if
          End if
          Return rvalue
End Algorithm
```

If Test A is true and USB_n_FRINDEX[2:0] is zero or one, then this is a case 2a or 2b scheduling boundary (see Figure 42-24). See Section Periodic Isochronous - Do Complete Split for details in handling this condition.

If Test A and Test B evaluate to true, then the host controller will execute a complete-split transaction using the transfer state of the current siTD. When the host controller commits to executing the complete-split transaction, it updates *QH.C-prog-mask* by bit-ORing with *cMicroFrameBit*. The transfer state is advanced based on the completion status of the complete-split transaction. To advance the transfer state of an IN siTD, the host controller must:

- Decrement the number of bytes received from *siTD.Total Bytes To Transfer,*
- Adjust *siTD.Current Offset* by the number of bytes received,

- Adjust *siTD.P* (page selector) field if the transfer caused the host controller to use the next page pointer, and
- Set any appropriate bits in the *siTD.Status* field, depending on the results of the transaction.

Note that if the host controller encounters a condition where *siTD.Total Bytes To Transfer* is zero, and it receives more data, the host controller must not write the additional data to memory. The *siTD.Status.Active* bit must be set to zero and the *siTD.Status.Babble Detected* bit must be set to a one. The fields *siTD.Total Bytes To Transfer*, *siTD.Current Offset*, and *siTD.P* (page selector) are not required to be updated as a result of this transaction attempt.

The host controller must accept (assuming good data packet CRC and sufficient room in the buffer as indicated by the value of *siTD.Total Bytes To Transfer*) MDATA and DATA0/1 data payloads up to and including 192 bytes. A host controller implementation may optionally set *siTD.Status Active* to a zero and *siTD.Status.Babble Detected* to a one when it receives and MDATA or DATA0/1 with a data payload of more than 192 bytes. The following responses have the noted effects:

- ERR. The full-speed transaction completed with a time-out or bad CRC and this is a reflection of that error to the host. The host controller sets the *ERR* bit in the *siTD.Status* field and sets the *Active* bit to a zero.
- Transaction Error (XactErr). The complete-split transaction encounters a Timeout, CRC16 failure, etc. The *siTD.Status* field *XactErr* field is set to a one and the complete-split transaction must be retried immediately. The host controller must use an internal error counter to count the number of retries as a counter field is not provided in the siTD data structure. The host controller will not retry more than two times. If the host controller exhausts the retries or the end of the micro-frame occurs, the *Active* bit is set to zero.
- DATAx (0 or 1). This response signals that the final data for the split transaction has arrived. The transfer state of the siTD is advanced and the *Active* bit is set to a zero. If the *Bytes To Transfer* field has not decremented to zero (including the reception of the data payload in the DATAx response), then less data than was expected, or allowed for was actually received. This *short packet* event does not set the USBINT status bit in the USBSTS register to a one. The host controller will not detect this condition.
- NYET (and Last). On each NYET response, the host controller also checks to determine whether this is the last complete-split for this split transaction. Last was defined in Section Periodic Isochronous - Do Complete Split . If it is the last complete-split (with a NYET response), then the transfer state of the siTD is not advanced (never received any data) and the *Active* bit is set to a zero. No bits are set in the *Status* field because this is essentially a skipped transaction. The transaction translator must have responded to all the scheduled clompete-splits with NYETs,

meaning that the start-split issued by the host controller was not received. This result should be interpreted by system software as if the transaction was completely skipped. The test for whether this is the last complete split can be performed by XORing C-mask with C-prog-mask. A zero result indicates that all complete-splits have been executed.

- MDATA (and Last). See above description for testing for Last. This can only occur when there is an error condition. Either there has been a babble condition on the full-speed link, which delayed the completion of the full-speed transaction, or software set up the *S-mask* and/or *C-mask*s incorrectly. The host controller must set *XactErr* bit to a one and the *Active* bit is set to a zero.
- NYET (and not Last). See above description for testing for Last. The complete-split transaction received a NYET response from the transaction translator. Do not update any transfer state (except for *C-prog-mask*) and stay in this state.
- MDATA (and not Last). The transaction translator responds with an MDATA when it has partial data for the split transaction. For example, the full-speed transaction data payload spans from micro-frame X to X+1 and during micro-frame X, the transaction translator will respond with an MDATA and the data accumulated up to the end of micro-frame X. The host controller advances the transfer state to reflect the number of bytes received.

If Test A succeeds, but Test B fails, it means that one or more of the complete-splits have been skipped. The host controller sets the *Missed Micro-Frame* status bit and sets the *Active* bit to a zero.

42.5.3.12.3.6 Complete-Split for Scheduling Boundary Cases 2a, 2b

Boundary cases 2a and 2b (INs only) (see Figure 42-24) require that the host controller use the transaction state context of the previous siTD to finish the split transaction. The table below enumerates the transaction state fields.

Table 42-51. Summary siTD Split Transaction State

Buffer State	Status	Execution Progress
Total Bytes To Transfer P (page select) Current Offset TP (transaction position) T-count (transaction count)	All bits in the status field	C-prog-mask

NOTE

TP and *T-count* are used only for Host to Device (OUT) endpoints.

If software has budgeted the schedule of this data stream with a frame wrap case, then it must initialize the *siTD.Back Pointer* field to reference a valid siTD and will have the *siTD.Back Pointer.T-bit* in the *siTD.Back Pointer*

field set to a zero. Otherwise, software must set the *siTD.Back Pointer.T-bit* in the *siTD.Back Pointer* field to a one. The host controller's rules for interpreting when to use the *siTD.Back Pointer* field are listed below. These rules apply only when the siTD's *Active* bit is a one and the *SplitXState* is Do Complete Split.

- When cMicroFrameBit is a 1h and the siTDX.Back Pointer.T-bit is a zero, or
- If cMicroFrameBit is a 2h and siTDX.S-mask[0] is a zero

When either of these conditions apply, then the host controller must use the transaction state from $siTD_{X-1}$.

In order to access $siTD_{X-1}$, the host controller reads on-chip the siTD referenced from $siTD_X$.*Back Pointer*.

The host controller must save the entire state from $siTD_X$ while processing $siTD_{X-1}$. This is to accommodate for case 2b processing. The host controller must not recursively walk the list of *siTD.Back Pointers*.

If $siTD_{X-1}$ is active (*Active* bit is a one and *SplitXStat* is Do Complete Split), then both Test A and Test B are applied as described above. If these criteria to execute a complete-split are met, the host controller executes the complete split and evaluates the results as described above. The transaction state (see Table 42-51) of $siTD_{X-1}$ is appropriately advanced based on the results and written back to memory. If the resultant state of $siTD_{X-1}$'s *Active* bit is a one, then the host controller returns to the context of $siTD_X$, and follows its next pointer to the next schedule item. No updates to $siTD_X$ are necessary.

If $siTD_{X-1}$ is active (*Active* bit is a one and *SplitXStat* is Do Start Split), then the host controller must set *Active* bit to a zero and *Missed Micro-Frame* status bit to a one and the resultant status written back to memory.

If $siTD_{X-1}$*'sActive* bit is a zero, (because it was zero when the host controller first visited $siTD_{X-1}$ via $siTD_X$*'s* back pointer, it transitioned to zero as a result of a detected error, or the results of $siTD_{X-1}$*'s* complete-split transaction transitioned it to zero), then the host controller returns to the context of $siTD_X$ and transitions its *SplitXState* to Do Start Split. The host controller then determines whether the case 2b start split boundary condition exists (that is, if *cMicroframeBit* is a 1b and $siTD_X$.*S-mask*[0] is a 1b). If this criterion is met the host controller immediately executes a start-split transaction and appropriately advances the transaction state of $siTD_X$, then follows $siTD_X$.*Next Pointer* to the next schedule item. If the criterion is not met, the host controller simply follows $siTD_X$.*Next Pointer* to the next schedule item. Note that in the case of a 2b boundary case, the split-transaction of $siTD_{X-1}$ will have its *Active* bit set to zero when the host controller returns

to the context of $siTD_X$. Also, note that software should not initialize an siTD with *C-mask* bits 0 and 1 set to a one and an *S-mask* with bit zero set to a one. This scheduling combination is not supported and the behavior of the host controller is undefined.

42.5.3.12.3.7 Split Transaction for Isochronous - Processing Examples

There is an important difference between how the hardware/software manages the isochronous split transaction state machine and how it manages the asynchronous and interrupt split transaction state machines.

The asynchronous and interrupt split transaction state machines are encapsulated within a single queue head. The progress of the data stream depends on the progress of each split transaction. In some respects, the split-transaction state machine is sequenced via the Execute Transaction queue head traversal state machine (see Figure 42-17).

Isochronous is a pure time-oriented transaction/data stream. The interface data structures are optimized to efficiently describe transactions that need to occur at specific times. The isochronous split-transaction state machine must be managed across these time-oriented data structures. This means that system software must correctly describe the scheduling of split-transactions across more than one data structure.

Then the host controller must make the appropriate state transitions at the appropriate times, in the correct data structures.

For example, the table below illustrates a couple of frames worth of scheduling required to schedule a case 2a full-speed isochronous data stream.

Table 42-52. Example Case 2a - Software Scheduling siTDs for an IN Endpoint

siTDX		Micro-Frames								Initial
#	Masks	0	1	2	3	4	5	6	7	SplitXState
X	S-Mask	-	-	-	-	1	-	-	-	Do Start Split
	C-Mask	1	1	-	-	-	-	1	1	
X+1	S-Mask	-	-	-	-	1	-	-	-	Do Complete Split
	C-Mask	1	1					1	1	
X+2	S-Mask	-	-	-	-	1	-	-	-	Do Complete Split
	C-Mask	1	1					1	1	
X+3	S-Mask	Repeats previous pattern								Do Complete Split
	C-Mask									

This example shows the first three siTDs for the transaction stream. Because this is the case-2a frame-wrap case, *S-masks* of all siTDs for this endpoint have a value of 10h (a one bit in micro-frame 4) and *C-mask* value of C3h (one-bits in micro-frames 0,1, 6 and 7). Additionally, sofware ensures that the *Back Pointer* field of each siTD references the appropriate siTD data structure (and the *Back PointerT-bits* are set to zero).

The initial *SplitXState* of the first siTD is Do Start Split. The host controller will visit the first siTD eight times during frame X. The C-mask bits in micro-frames 0 and 1 are ignored because the state is Do Start Split. During micro-frame 4, the host controller determines that it can run a start-split (and does) and changes *SplitXState* to Do Complete Split. During micro-frames 6 and 7, the host controller executes complete-splits. Notice the siTD for frame X+1 has its *SplitXState* initialized to Do Complete Split. As the host controller continues to traverse the schedule during *H-Frame* X+1, it will visit the second siTD eight times. During micro-frames 0 and 1 it will detect that it must execute complete-splits.

During *H-Frame* X+1, micro-frame 0, the host controller detects that $siTD_{X+1}$'s *Back Pointer.T-bit* is a zero, saves the state of $siTD_{X+1}$ and fetches $siTD_X$. It executes the complete split transaction using the transaction state of $siTD_X$. If the $siTD_X$ split transaction is complete, siTD's *Active* bit is set to zero and results written back to $siTD_X$. The host controller retains the fact that $siTD_X$ is retired and transitions the *SplitXState* in the $siTD_{X+1}$ to Do Start Split. At this point, the host controller is prepared to execute the start-split for $siTD_{X+1}$ when it reaches micro-frame 4. If the split-transaction completes early (transaction-complete is defined in Section Periodic Isochronous - Do Complete Split), that is, before all the scheduled complete-splits have been executed, the host controller will transition $siTD_X$.*SplitXState* to Do Start Split early and naturally skip the remaining scheduled complete-split transactions. For this example, $siTD_{X+1}$ does not receive a DATA0 response until *H-Frame* X+2, micro-frame 1.

During *H-Frame X+2,* micro-frame 0, the host controller detects that $siTD_{X+2}$'s *Back Pointer.T-bit* is a zero, saves the state of $siTD_{X+2}$ and fetches $siTD_{X+1}$. As described above, it executes another split transaction, receives an MDATA response, updates the transfer state, but does not modify the *Active* bit. The host controller returns to the context of $siTD_{X+2}$, and traverses its next pointer without any state change updates to $siTD_{X+2}$. S

During *H-Frame X+2*, micro-frame 1, the host controller detects $siTD_{X+2}$'s *S-mask[0]* is a zero, saves the state of $siTD_{X+2}$ and fetches $siTD_{X+1}$. It executes another complete-split transaction, receives a DATA0 response, updates the transfer state and sets the *Active* bit to a zero. It returns to the state of $siTD_{X+2}$ and changes its *SplitXState* to Do Start Split. At this point, the host controller is prepared to execute start-splits for $siTD_{X+2}$ when it reaches micro-frame 4.

42.5.3.13 Host Controller Pause

When the host controller's *HCHalted* bit in the USBSTS register is a zero, the host controller is sending SOF (Start OF Frame) packets down all enabled ports.

When the schedules are enabled, the EHCI host controller will access the schedules in main memory each micro-frame. This constant pinging of main memory is known to create Arm platform power management problems for mobile systems. Specifically, mobile systems aggressively manage the state of the Arm platform, based on recent history usage. In the more aggressive power saving modes, the Arm platform can disable its caches. Current PC architectures assume that bus-master accesses to main memory must be cache-coherent. So, when bus masters are busy touching memory, the Arm platform power management software can detect this activity over time and inhibit the transition of the Arm platform into its lowest power savings mode. USB controllers are bus-masters and the frequency at which they access their memory-based schedules keeps the Arm platform power management software from placing the Arm platform into its lowest power savings state.

USB Host controllers don't access main memory when they are suspended. However, there are a variety of reasons why placing the USB controllers into suspend won't work, but they are beyond the scope of this document. The base requirement is that the USB controller needs to be kept out of main memory, while at the same time, the USB bus is kept from going into suspend.

EHCI controllers provide a large-grained mechanism that can be manipulated by system software to change the memory access pattern of the host controller. System software can manipulate the schedule enable bits in the USBCMD register to turn on/off the scheduling traversal. A software heuristic can be applied to implement an on/off duty cycle that allows the USB to make reasonable progress and allow the Arm platform power management to get the Arm platform into its lowest power state. This method is not intended to be applied at all times to throttle USB, but should only be applied in very specific configurations and usage loads. For example, when only a keyboard or mouse is attached to the USB, the heuristic could detect times when the USB is attempting to move data only very infrequently and can adjust the duty cycle to allow the Arm platform to reach its low power state for longer periods of time. Similarly, it could detect increases in the USB load and adjust the duty cycle appropriately, even to the point where the schedules are never disabled. The assumption here is that the USB is moving data and the Arm platform will be required to process the data streams.

It is suggested that in order to provide a complete solution for the system, the companion host controllers should also provide a similar method to allow system software to inhibit the companion host controller from accessing its shared memory based data structures (schedule lists or otherwise).

42.5.3.14 Port Test Modes -Host Operational Model

EHCI host controllers must implement the port test modes Test J_State, Test K_State, Test_Packet, Test Force_Enable, and Test SE0_NAK as described in the USB Specification Revision 2.0.

The system is only allowed to test ports that are owned by the EHCI controller (for example, *CF-bit* is a one and *PortOwner* bit is a zero). System software is allowed to have at most one port in test mode at a time. Placing more than one port in test mode will yield undefined results. The required, per port test sequence is (assuming the *CF-bit* in the USB_n_CONFIGFLAG register is a one):

- Disable the periodic and asynchronous schedules by setting the *Asynchronous Schedule Enable* and *Periodic Schedule Enable* bits in the USBCMD register to a zero.
- Place all enabled root ports into the suspended state by setting the *Suspend* bit in each appropriate USB_n_PORTSC register to a one.
- Set the *Run/Stop* bit in the USBCMD register to a zero and wait for the *HCHalted* bit in the USBSTS register, to transition to a one. Note that an EHCI host controller implementation may optionally allow port testing with the *Run/Stop* bit set to a one. However, all host controllers must support port testing with *Run/Stop* set to a zero and *HCHalted* set to a one.
- Set the *Port Test Control* field in the port under test PORTSC register to the value corresponding to the desired test mode. If the selected test is Test_Force_Enable, then the *Run/Stop* bit in the USBCMD register must then be transitioned back to one, in order to enable transmission of SOFs out of the port under test.
- When the test is complete, system software must ensure the host controller is halted (*HCHalted* bit is a one) then it terminates and exits test mode by setting *HCReset* to a one.

42.5.3.15 Interrupts-Host Operational Model

The EHCI Host Controller hardware provides interrupt capability based on a number of sources.

There are several general groups of interrupt sources:

- Interrupts as a result of executing transactions from the schedule (success and error conditions),
- Host controller events (Port change events, etc.), and
- Host Controller error events

All transaction-based sources are maskable through the Host Controller's Interrupt Enable register (USBINTR, see Section Interrupt Enable Register (USB_*n*USBINTR)). Additionally, individual transfer descriptors can be marked to generate an interrupt on completion. This section describes each interrupt source and the processing that occurs in response to the interrupt.

During normal operation, interrupts may be immediate or deferred until the next interrupt threshold occurs. The interrupt threshold is a tunable parameter via the *Interrupt Threshold Control* field in the USBCMD register. The value of this register controls when the host controller will generate an interrupt on behalf of normal transaction execution. When a transaction completes during an interrupt interval period, the interrupt signaling the completion of the transfer will not occur until the interrupt threshold occurs. For example, the default value is eight micro-frames. This means that the host controller will not generate interrupts any more frequently than once every eight micro-frames.

Section Host System Error details effects of a host system error.

If an interrupt has been scheduled to be generated for the current interrupt threshold interval, the interrupt is not signaled until after the status for the last complete transaction in the interval has been written back to host memory. This may sometimes result in the interrupt not being signaled until the next interrupt threshold.

Initial interrupt processing is the same, regardless of the reason for the interrupt. When an interrupt is signaled by the hardware, Arm platform control is transferred to host controller's USB interrupt handler. The precise mechanism to accomplish the transfer is OS specific. For this discussion it is just assumed that control is received. When the interrupt handler receives control, its first action is to reads the USBSTS (USB Status Register). It then acknowledges the interrupt by clearing all of the interrupt status bits by writing ones to these bit positions. The handler then determines whether the interrupt is due to schedule processing or some other event. After acknowledging the interrupt, the handler (via an OS-specific mechanism), schedules a deferred procedure call (DPC) which will execute later. The DPC routine processes the results of the schedule execution. The precise mechanisms used are beyond the scope of this document.

Note: the host controller is not required to de-assert a currently active interrupt condition when software sets the interrupt enables (in the USBINR register, see Section Interrupt Enable Register (USB_*n*USBINTR)) to a zero. The only reliable method software should use for acknowledging an interrupt is by transitioning the appropriate status bits in the USBSTS register (Section USB Status Register (USB_*n*USBSTS)) from a one to a zero.

42.5.3.15.1 Transfer/Transaction Based Interrupts

These interrupt sources are associated with transfer and transaction progress. They are all dependent on the next interrupt threshold.

42.5.3.15.1.1 Transaction Error

A transaction error is any error that caused the host controller to think that the transfer did not complete successfully.

The table below lists the events/responses that the host can observe as a result of a transaction. The effects of the error counter and interrupt status are summarized in the following paragraphs. Most of these errors set the *XactErr* status bit in the appropriate interface data structure.

There is a small set of protocol errors that relate only when executing a queue head and fit under the umbrella of a WRONG PID error that are significant to explicitly identify. When these errors occur, the *XactErr* status bit in the queue head is set and the *CErr* field is decremented. When the *PIDCode* indicates a SETUP, the following responses are protocol errors and result in *XactErr* bit being set to a one and the *CErr* field being decremented.

- *EPS* field indicates a high-speed device and it returns a Nak handshake to a SETUP.
- *EPS* field indicates a high-speed device and it returns a Nyet handshake to a SETUP.
- *EPS* field indicates a low- or full-speed device and the complete-split receives a Nak handshake.

Table 42-53. Summary of Transaction Errors

Event / Result	Queue Head/qTD/iTD/siTD Side-effects		USB Status Register (USBSTS)
	Cerr	Status Field	USBERRINT
CRC	-1	XactErr set to a one.	1[1]
Timeout	-1	XactErr set to a one.	1[1]
Bad PID[2]	-1	XactErr set to a one.	1[1]
Babble	N/A	Section Serial Bus Babble	1
Buffer Error	N/A	Section Data Buffer Error	

1. If occurs in a queue head, then *USBERRINT* is asserted only when *CErr* counts down from a one to a zero. In addition the queue is halted, see Halting a Queue Head.
2. The host controller received a response from the device, but it could not recognize the PID as a valid PID.

42.5.3.15.1.2 Serial Bus Babble

When a device transmits more data on the USB than the host controller is expecting for this transaction, it is defined to be babbling. In general, this is called a *Packet Babble*.

When a device sends more data than the *Maximum Length* number of bytes, the host controller sets the *Babble Detected* bit to a one and halts the endpoint if it is using a queue head (see Halting a Queue Head). *Maximum Length* is defined as the minimum of *Total Bytes to Transfer* and *Maximum Packet Size*. The *CErr* field is not decremented for a packet babble condition (only applies to queue heads). A babble condition also exists if IN transaction is in progress at High-speed EOF2 point. This is called a frame babble. A frame babble condition is recorded into the appropriate schedule data structure. In addition, the host controller must disable the port to which the frame babble is detected.

The *USBERRINT* bit in the USB_n_USBSTS register is set to a one and if the *USB Error Interrupt Enable* bit in the USB_n_USBINTR register is a one, then a hardware interrupt is signaled to the system at the next interrupt threshold. The host controller must never start an OUT transaction that will babble across a micro-frame EOF.

NOTE

When a host controller detects a data PID mismatch, it must either: disable the packet babble checking for the duration of the bus transaction or do packet babble checking based solely on *Maximum Packet Size*. The USB core specification defines the requirements on a data receiver when it receives a data PID mismatch (for example, expects a DATA0 and gets a DATA1 or visa-versa). In summary, it must ignore the received data and respond with an ACK handshake, in order to advance the transmitter's data sequence.

The EHCI interface allows System software to provide buffers for a Control, Bulk or Interrupt IN endpoint that are not an even multiple of the maximum packet size specified by the device. Whenever a device misses an ACK for an IN endpoint, the host and device are out of synchronization with respect to the progress of the data transfer. The host controller may have advanced the transfer to a buffer that is less than maximum packet size. The device will re-send its maximum packet size data packet, with the original data PID, in response to the next IN token. In order to properly manage the bus protocol, the host controller must disable the packet babble check when it observes the data PID mismatch.

42.5.3.15.1.3 Data Buffer Error

This event indicates that an overrun of incoming data or a underrun of outgoing data has occurred for this transaction.

This would generally be caused by the host controller not being able to access required data buffers in memory within necessary latency requirements. These conditions are not considered transaction errors, and do not effect the error count in the queue head. When these errors do occur, the host controller records the fact the error occurred by setting the *Data Buffer Error* bit in the queue head, iTD or siTD.

If the data buffer error occurs on a non-isochronous IN, the host controller will not issue a handshake to the endpoint. This will force the endpoint to resend the same data (and data toggle) in response to the next IN to the endpoint.

If the data buffer error occurs on an OUT, the host controller must corrupt the end of the packet so that it cannot be interpreted by the device as a good data packet. Simply truncating the packet is not considered acceptable. An acceptable implementation option is to 1's complement the CRC bytes and send them. There are other options suggested in the Transaction Translator section of the USB Specification Revision 2.0.

42.5.3.15.1.4 USB Interrupt (Interrupt on Completion (IOC))

Transfer Descriptors (iTDs, siTDs, and queue heads (qTDs)) contain a bit that can be set to cause an interrupt on their completion. The completion of the transfer associated with that schedule item causes the USB Interrupt (USBINT) bit in the USB_n_USBSTS register to be set to a one.

In addition, if a short packet is encountered on an IN transaction associated with a queue head, then this event also causes USBINT to be set to a one. If the USB Interrupt Enable bit in the USB_n_USBINTR register is set to a one, a hardware interrupt is signaled to the system at the next interrupt threshold. If the completion is because of errors, the *USBERRINT* bit in the USB_n_USBSTS register is also set to a one.

42.5.3.15.1.5 Short Packet

Reception of a data packet that is less than the endpoint's Max Packet size during Control, Bulk or Interrupt transfers signals the completion of the transfer. Whenever a short packet completion occurs during a queue head execution, the *USBINT* bit in the USB_n_USBSTS register is set to a one.

If the *USB Interrupt Enable* bit is set in the USB_n_USBINTR register, a hardware interrupt is signaled to the system at the next interrupt threshold.

42.5.3.15.2 Host Controller Event Interrupts

These interrupt sources are independent of the interrupt threshold (with the one exception being the Interrupt on Async Advance, see Section Interrupt on Async Advance).

42.5.3.15.2.1 Port Change Events

Port registers contain status and status change bits. When the status change bits are set to a one, the host controller sets the *Port Change Detect* bit in the USBSTS register to a one.

If the *Port Change Interrupt Enable* bit in the USB_n_USBINTR register is a one, then the host controller will issue a hardware interrupt. The port status change bits include:

- Connect Status Change
- Port Enable/Disable Change
- Over-current Change
- Force Port Resume

42.5.3.15.2.2 Frame List Rollover

This event indicates that the host controller has wrapped the frame list. The current programmed size of the frame list effects how often this interrupt occurs.

If the frame list size is 1024, then the interrupt will occur every 1024 milliseconds, if it is 512, then it will occur every 512 milliseconds, etc. When a frame list rollover is detected, the host controller sets the *Frame List Rollover* bit in the USB.USBSTS register to a one. If the *Frame List Rollover Enable* bit in the USB.USBINTR register is set to a one, the host controller issues a hardware interrupt. This interrupt is not delayed to the next interrupt threshold.

42.5.3.15.2.3 Interrupt on Async Advance

This event is used for deterministic removal of queue heads from the asynchronous schedule. Whenever the host controller advances the on-chip context of the asynchronous schedule, it evaluates the value of the *Interrupt on Async Advance Doorbell* bit in the USB.USBCMD register.

If it is a one, it sets the *Interrupt on Async Advance* bit in the USB.USBSTS register to a one. If the *Interrupton Async Advance Enable* bit in the USB.USBINTR register is a one, the host controller issues a hardware interrupt at the next interrupt threshold. A detailed explanation of this feature is described in Section Removing Queue Heads from Asynchronous Schedule .

42.5.3.15.2.4 Host System Error

The host controller is a bus master and any interaction between the host controller and the system may experience errors.

The type of host error may be catastrophic to the host controller (such as a Master Abort) making it impossible for the host controller to continue in a coherent fashion.In the presence of non-catastrophic host errors, such as parity errors, the host controller could potentially continue operation. The recommended behavior for these types of errors is to escalate it to a catastrophic error and halt the host controller. Host-based error must result in the following actions:

- The *Run/Stop* bit in the USB.USBCMD register is set to a zero.
- The following bits in the USB.USBSTS register are set:
  - *Host System Error* bit is to a one.
  - *HCHalted* bit is set to a one.
- If the *Host System Error Enable* bit in the USB.USBINTR register is a one, then the host controller will issue a hardware interrupt. This interrupt is not delayed to the next interrupt threshold. The following table summarizes the required actions taken on the various host errors.

Table 42-54. Summary Behavior of EHCI Host Controller on Host System Errors

Cycle Type	Master Abort	Target Abort	Data Phase Parity
Frame list pointer fetch (read)	Fatal	Fatal	Fatal [o]
siTD fetch (read)	Fatal	Fatal	Fatal [o]
siTD status write-back (write)	Fatal [o]	Fatal [o]	Fatal [o]
iTD fetch (read)	Fatal	Fatal	Fatal [o]
iTD status write-back (write)	Fatal [o]	Fatal [o]	Fatal [o]
qTD fetch (read)	Fatal	Fatal	Fatal [o]
qHD status write-back (write)	Fatal [o]	Fatal [o]	Fatal [o]
Data write	Fatal [o]	Fatal [o]	Fatal [o]
Data read	Fatal	Fatal	Fatal [o]

Potentially, a host controller implementation could continue operation without a halt. However, the recommended behavior is to halt the host controller.

NOTE

After a *Host System Error*, Software must reset the host controller through *HCReset* in the USB.USBCMD register before re-initializing and restarting the host controller.

42.5.4 EHCI Deviation

For the purposes a dual-role Host/Device controller with support for On-The-Go applications, it is necessary to deviate from the EHCI specification. Enhanced Host Controller Interface Specification for Universal Serial Bus, Revision 0.95, November 2000, Intel Corporation. http://www.intel.com. Device operation & On-The-Go operation is not specified in the EHCI and thus the implementation supported in this core is proprietary.

The host mode operation of the core is near EHCI compatible with few minor differences documented in this section.

The particulars of the deviations occur in the areas summarized here:

- Embedded Transaction Translator - Allows direct attachment of FS and LS devices in host mode without the need for a companion controller.
- Device operation - In host mode the device operational registers are generally disabled and thus device mode is mostly transparent when in host mode. However, there are a couple exceptions documented in the following sections.
- Embedded design interface - This core does not have a PCI Interface and therefore the PCI configuration registers described in the EHCI specification are not applicable.
- On-The-Go Operation - This design includes an On-The-Go controller for Port #1.

42.5.4.1 Embedded Transaction Translator Function

The OTG controller supports directly connected full and low speed devices without requiring a companion controller by including the capabilities of a USB 2.0 high speed hub transaction translator.

Although there is no separate Transaction Translator block in the system, the transaction translator function normally associated with a high speed hub has been implemented within the DMA and Protocol engine blocks. The embedded transaction translator function is an extension to EHCI interface, but makes use of the standard data structures and operational models that exist in the EHCI specification to support full and low speed devices.

42.5.4.1.1 Capability Registers

The following additions have been added to the capability registers to support the embedded Transaction Translator Function:

- N_TT added to USB.HCSPARAMS - Host Control Structural Parameters
- N_PTT added to USB.HCSPARAMS - Host Control Structural Parameters

42.5.4.1.2 Operational Registers

The following additions have been added to the operational registers to support the embedded TT:

- Addition of two-bit Port Speed (PSPD) to the Port Status & Control (USB_*n*PORTSC1) register.

42.5.4.1.3 Discovery-EHCI Deviation

In a standard EHCI controller design, the EHCI host controller driver detects a Full speed (FS) or Low speed (LS) device by noting if the port enable bit is set after the port reset operation.

The port enable will only be set in a standard EHCI controller implementation after the port reset operation and when the host and device negotiate a High-Speed connection (that is, Chirp completes successfully).

Because this controller has an embedded Transaction Translator, the port enable will always be set after the port reset operation regardless of the result of the host device chirp result and the resulting port speed will be indicated by the PSPD field in USB.PORTSCx.

Therefore, the standard EHCI host controller driver requires an alteration to handle directly connected Full and Low speed devices or hubs.

The change is a fundamental one in that is summarized in the following table.

Table 42-55. Summary of EHCI

Standard EHCI	EHCI with embedded Transaction Translator
After port enable bit is set following a connection and reset sequence, the device/hub is assumed to be HS.	After port enable bit is set following a connection and reset sequence, the device/hub speed is noted from USB.PORTSCx.
FS and LS devices are assumed to be downstream from a HS hub thus, all port-level control is performed through the Hub Class to the nearest Hub.	FS and LS device can be either downstream from a HS hub or directly attached. When the FS/LS device is downstream from a HS hub, then port-level control is done using the Hub Class through the nearest Hub. When a FS/LS device is directly attached, then port-level control is accomplished using USB.PORTSCx.
FS and LS devices are assumed to be downstream from a HS hub with HubAddr=X. [where HubAddr > 0 and HubAddr is the address of the Hub where the bus transitions from HS to FS/LS (ie. Split target hub)]	FS and LS device can be either downstream from a HS hub with HubAddr = X [HubAddr > 0] or directly attached [where HubAddr = 0 and HubAddr is the address of the Root Hub where the bus transitions from HS to FS/LS (ie. Split target hub is the root hub)]

42.5.4.1.4 Data Structures

The same data structures used for FS/LS transactions though a HS hub are also used for transactions through the Root Hub with sm embedded Transaction Translator.

Here it is demonstrated how the Hub Address and Endpoint Speed fields should be set for directly attached FS/LS devices and hubs:

1. QH (for direct attach FS/LS) - Async. (Bulk/Control Endpoints) Periodic (Interrupt)
   - Hub Address = 0
   - Transactions to direct attached device/hub.
     - QH.EPS = Port Speed
   - Transactions to a device downstream from direct attached FS hub.
     - QH.EPS = Downstream Device Speed

NOTE

When QH.EPS = 01 (LS) and PORTSCx.PSPD = 00 (FS), a LS-pre-pid will be sent before the transmitting LS traffic.

Maximum Packet Size must be less than or equal 64 or undefined behaviour may result.

2. siTD (for direct attach FS) - Periodic (ISO Endpoint)
   - All FS ISO transactions:
     - Hub Address = 0
     - siTD.EPS = 00 (full speed)
       - Maximum Packet Size must less than or equal to 1023 or undefined behaviour may result.

42.5.4.1.5 Operational Model

The operational models are well defined for the behavior of the Transaction Translator (see USB 2.0 specification. Universal Serial Bus Specification, Revision 2.0, April 2000, Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, NEC, Philips. http://www.usb.org) and for the EHCI controller moving packets between system memory and a USB-HS hub. Because the embedded Transaction Translator exists within the host controller there is no physical bus between EHCI host controller driver and the USB FS/LS bus. These sections will briefly discuss the operational model for how the EHCI and Transaction Translator operational models are combined without the physical bus between. The following sections assume the reader is familiar with both the EHCI and USB 2.0 Transaction Translator operational models.

42.5.4.1.5.1 Micro- frame Pipeline

The EHCI operational model uses the concept of H-frames and B-frames to describe the pipeline between the Host (H) and the Bus (B). The embedded Transaction Translator shall use the same pipeline algorithms specified in the USB 2.0 specification for a Hub-based Transaction Translator.

All periodic transfers always begin at B-frame 0 (after SOF) and continue until the stored periodic transfers are complete. As an example of the micro-frame pipeline implemented in the embedded Transaction Translator, all periodic transfers that are tagged in EHCI to execute in H-frame 0 will be ready to execute on the bus in B-frame 0.

It is important to note that when programming the S-mask and C-masks in the EHCI data structures to schedule periodic transfers for the embedded Transaction Translator, the EHCI host controller driver must follow the same rules specified in EHCI for programming the S-mask and C-mask for downstream Hub-based Transaction Translators.

Once periodic transfers are exhausted, any stored asynchronous transfer will be moved. Asynchronous transfers are opportunistic in that they shall execute whenever possible and their operation is not tied to H-frame and B-frame boundaries with the exception that an asynchronous transfer can not babble through the SOF (start of B-frame 0.)

42.5.4.1.5.2 Split State Machines

The start and complete split operational model differs from EHCI slightly because there is no bus medium between the EHCI controller and the embedded Transaction Translator.

Where a start or complete-split operation would occur by requesting the split to the HS hub, the start/complete split operation is simple an internal operation to the embedded Transaction Translator. The following table summarizes the conditions where handshakes are emulated from internal state instead of actual handshakes to HS split bus traffic.

Table 42-56. Summary of the Conditons of Handshakes[1]

Condition	Emulate TT Response
Start-Split: All asynchronous buffers full.	NAK
Start-Split: All periodic buffers full.	ERR
Start-Split: Success for start of Async. Transaction.	ACK
Start-Split: Start Periodic Transaction.	No Handshake (Ok)
Complete-Split: Failed to find transaction in queue.	Bus Time Out
Complete-Split: Transaction in Queue is Busy.	NYET
Complete-Split: Transaction in Queue is Complete.	[Actual Handshake from LS/FS device]

1. The un-shaded cells represent Start-Splits and the shaded cells represent Complete-Splits

42.5.4.1.5.3 Asynchronous Transaction Scheduling and Buffer Management

The following USB 2.0 specification items are implemented in the embedded Transaction Translator:

- Sequencing is provided & a packet length estimator ensures no full-speed/low-speed packet babbles into SOF time.
- Transaction tracking for 2 data pipes.

42.5.4.1.5.3.1 USB 2.0 - 11.17.3

- Sequencing is provided & a packet length estimator ensures no full-speed/low-speed packet babbles into SOF time.

42.5.4.1.5.3.2 USB 2.0 - 11.17.4

- Transaction tracking for 2 data pipes.

42.5.4.1.5.4 Periodic Transaction Scheduling and Buffer Management

The following USB 2.0 specification items are implemented in the embedded Transaction Translator:

- Abort of pending start-splits
  - EOF (and not started in micro-frames 6)
  - Idle for more than 4 micro-frames
- Abort of pending complete-splits
  - EOF
  - Idle for more than 4 micro-frames
- Transaction tracking for up to 16 data pipes.
- Complete-split transaction searching.

NOTE

There is no data schedule mechanism for these transactions other than the micro-frame pipeline. The embedded TT assumes the number of packets scheduled in a frame does not exceed the frame duration (1 ms) or else undefined behavior may result.

42.5.4.1.5.4.1 USB 2.0 - 11.18.6.[1-2]

- Abort of pending start-splits
  - EOF (and not started in micro-frames 6)
  - Idle for more than 4 micro-frames
- Abort of pending complete-splits

- EOF
- Idle for more than 4 micro-frames

42.5.4.1.5.4.2 USB 2.0 - 11.18.[7-8]

- Transaction tracking for up to 16 data pipes.
- Complete-split transaction searching.

NOTE

There is no data schedule mechanism for these transactions other than the micro-frame pipeline. The embedded TT assumes the number of packets scheduled in a frame does not exceed the frame duration (1 ms) or else undefined behavior may result.

42.5.4.1.5.5 Multiple Transaction Translators

The maximum number of embedded Transaction Translators that is currently supported is one as indicated by the N_TT field in the Host Controller Structural Parameters (USB_*n*HCSPARAMS) register.

42.5.4.2 Device Operation

The co-existence of a device operational controller within the host controller has little effect on EHCI compatibility for host operation except as noted in this section.

42.5.4.2.1 USB_USBMODE Register

Given that the dual-role controller is initialized in neither host nor device mode, the USB Device Mode (USB_*n*USBMODE) register must be programmed for host operation before the EHCI host controller driver can begin EHCI host operations.

42.5.4.2.2 Non-Zero Fields the Register File

Some of the reserved fields and reserved addresses in the capability registers and operational register have use in device mode, the following must be adhered to:

- Write operations to all EHCI reserved fields (some of which are device fields) with the operation registers should always be written to zero. This is an EHCI requirement of the device controller driver that must be adhered to.
- Read operations by the host controller must properly mask EHCI reserved fields (some of which are device fields) because fields that are used exclusive for device are undefined in host mode .

42.5.4.2.3 SOF Interrupt

This SOF Interrupt used for device mode is shared as a free running 125us interrupt for host mode.

EHCI does not specify this interrupt but it has been added for convenience and as a potential software time base. See USB Status Register (USB_*n*USBSTS) and Interrupt Enable Register (USB_*n*USBINTR) registers.

42.5.4.3 Embedded Design Interface

This is an Embedded USB Host Controller as defined by the EHCI specification and thus does not implement the PCI configuration registers.

42.5.4.3.1 Frame Adjust Register

Given that the optional PCI configuration registers are not included in this implementation, there is no corresponding bit level timing adjustments like is provided by the Frame Adjust register in the PCI configuration registers. Starts of micro-frames are timed precisely to 125 us using the transceiver clock as a reference clock. That is, a 60 Mhz transceiver clock for 8-bit physical interfaces & full-speed serial interfaces or 30 Mhz transceiver clock for 16-bit physical interfaces.

42.5.4.4 Miscellaneous variations from EHCI

42.5.4.4.1 Programmable Physical Interface Behaviour

This design supports multiple Physical interfaces which can operate in differing modes when the core is configured with software programmable Physical Interface Modes. Software programmability allows the selection of the Physical interface part during the board design phase instead of during the chip design phase. The control bits for selecting the Physical Interface operating mode have been added to the Port Status & Control (USB_*n*PORTSC1) register providing a capability that is not defined by EHCI.

42.5.4.4.2 Discovery

42.5.4.4.2.1 Port Reset

The port connect methods specified by EHCI require setting the port reset bit in the Port Status & Control (USB_*n*PORTSC1) register for a duration of 10ms. Due to the complexity required to support the attachment of devices that are not high speed there are counter already present in the design that can count the 10ms reset pulse to alleviate the requirement of the software to measure this duration. Therefore, the basic connection is then summarized as the following:

- [Port Change Interrupt] Port connect change occurs to notify the host controller driver that a device has attached.
- Software shall write a '1' to reset the device.
- Software shall write a '0' to reset the device after 10 ms.
  - This step, which is necessary in a standard EHCI design, may be omitted with this implementation. Should the EHCI host controller driver attempt to write a '0' to the reset bit while a reset is in progress, the write will simple be ignored and the reset will continue until completion.
- [Port Change Interrupt] Port enable change occurs to notify the host controller that the device in now operational and at this point the port speed has been determined.

42.5.4.4.2.2 Port Speed Detection

After the port change interrupt indicates that a port is enabled, the EHCI stack should determine the port speed. Unlike the EHCI implementation, which will re-assign the port owner for any device that does not connect at High-Speed, this host controller supports direct attach of non High-Speed devices.

Therefore, the following differences are important regarding port speed detection:

- Port Owner is read-only and always reads 0.
- A 2-bit Port Speed indicator has been added to PORTSC to provide the current operating speed of the port to the host controller driver.
- A 1-bit High Speed indicator has been added to PORTSC to signify that the port is in High-Speed vs. Full/Low Speed - *This information is redundant with the 2-bit Port Speed indicator above.*

42.5.4.4.3 Port Test Mode

Port Test Control mode behaves fully as described in EHCI. An alternate host controller driver procedure is not necessary or supported.

42.5.5 Device Data Structures

This section defines the interface data structures used to communicate control, status, and data between Device Controller Driver (DCD) Software and the Device Controller.

The data structure definitions in this chapter support a 32-bit memory buffer address space. The interface consists of device Queue Heads and Transfer Descriptors.

NOTE

Software must ensure that no interface data structure reachable by the Device Controller spans a 4K-page boundary.

The data structures defined in the chapter are (from the device controller's perspective) a mix of read-only and read/ writable fields. The device controller must preserve the read-only fields on all data structure writes.

The figure below shows the organization of the EndPoint Queue Head.

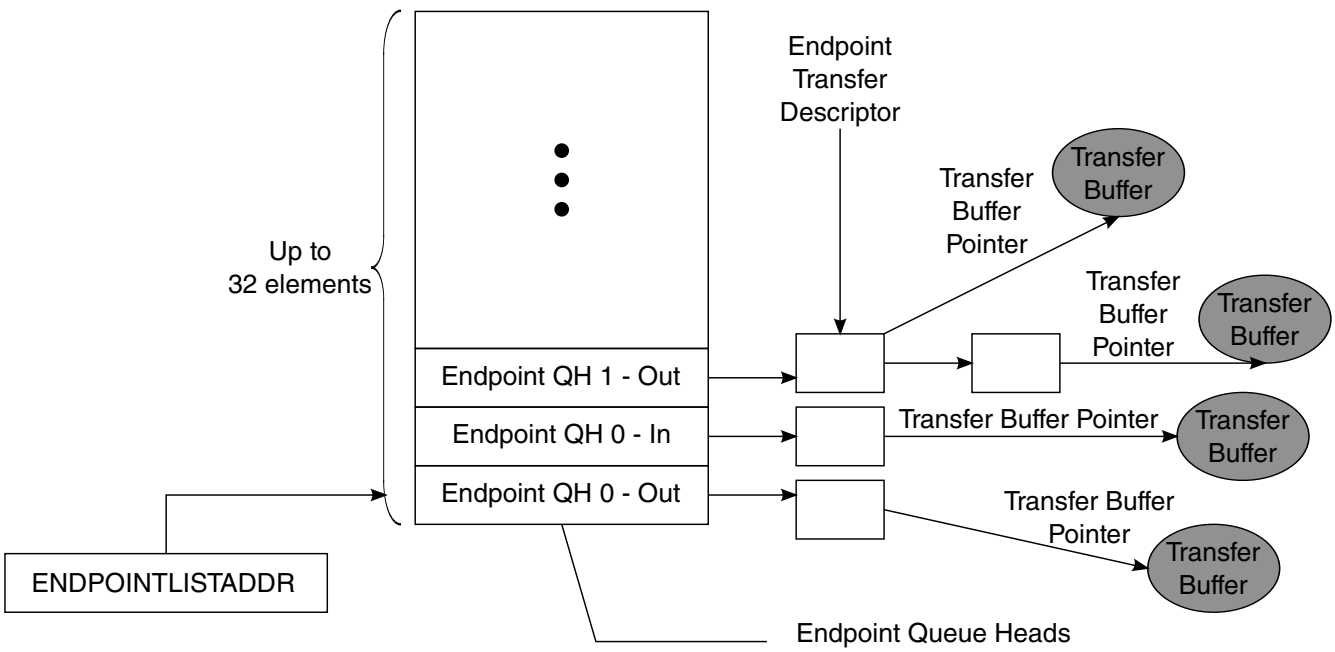


Figure 42-27. EndPoint Queue Head Organization

Endpoint queue heads are arranged in an array in a continuous area of memory pointed to by the USB.ENDPOINTLISTADDR pointer. The even -numbered device queue heads in the list support receive endpoints (OUT/SETUP) and the odd-numbered queue heads in the list are used for transmit endpoints (IN/INTERRUPT). The device controller will index into this array based upon the endpoint number received from the USB bus. All information necessary to respond to transactions for all primed transfers is contained in this list so the Device Controller can readily respond to incoming requests without having to traverse a linked list.

NOTE

The Endpoint Queue Head List must be aligned to a 2k boundary.

42.5.5.1 Endpoint Queue Head (dQH)

The device Endpoint Queue Head (dQH) is where all transfers for a given endpoint are managed. The dQH is a 48-byte data structure, but must be aligned on 64-byte boundaries.

During priming of an endpoint, the dTD (device transfer descriptor) is copied into the overlay area of the dQH, which starts at the nextTD pointer DWord and continues through the end of the buffer pointers DWords. After a transfer is complete, the dTD status DWord is updated in the dTD pointed to by the currentTD pointer. While a packet is in progress, the overlay area of the dQH is used as a staging area for the dTD so that the Device Controller can access needed information with little minimal latency.

Table 42-57. Endpoint Queue Head (dQH)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Mult		zlt	0		Maximum Packet Length											ios	0														
Current dTD Pointer																											0				
Next dTD Pointer																											0				T[1]
0	Total Bytes															ioc	0			MultO		0		Status							
Buffer Pointer (Page 0)																				Current Offset											
Buffer Pointer (Page 1)																				Reserved											
Buffer Pointer (Page 2)																				Reserved											
Buffer Pointer (Page 3)																				Reserved											
Buffer Pointer (Page 4)[1]																				Reserved											
Reserved																															
Set-up Buffer Bytes 3…0																															
Set-up Buffer Bytes 7…4																															

1. Transfer overlay starts at T and continues through Buffer Pointer (Page 4).

Host Controller Read/Write

Host Controller Read Only

42.5.5.1.1 Endpoint Capabilities/Characteristics

This DWord specifies static information about the endpoint, in other words, this information does not change over the lifetime of the endpoint. Device Controller software should not attempt to modify this information while the corresponding endpoint is enabled.

Table 42-58 describes the endpoint capabilities.

Table 42-58. Endpoint Capabilities/Characteristics

Bit	Description
31-30	Mult. This field is used to indicate the number of packets executed per transaction description as given by the following: 00 - Execute N Transactions as demonstrated by the USB variable length packet protocol where N is computed using the Maximum Packet Length (dQH) and the Total Bytes field (dTD) 01 Execute 1 Transaction. 10 Execute 2 Transactions. 11 Execute 3 Transactions. **NOTE:** Non-ISO endpoints must set Mult="00". ISO endpoints must set Mult="01", "10", or "11" as needed.
29	Zero Length Termination Select. This bit is used to indicate when a zero length packet is used to terminate transfers where to total transfer length is a multiple . This bit is not relevant for Isochronous 0 - Enable zero length packet to terminate transfers equal to a multiple of the Maximum Packet Length. (default). 1 - Disable the zero length packet on transfers that are equal in length to a multiple Maximum Packet Length.
28-27	Reserved. These bit reserved for future use and should be set to zero.
26-16	Maximum Packet Length. This directly corresponds to the maximum packet size of the associated endpoint (wMaxPacketSize). The maximum value this field may contain is 0x400 (1024).
15	Interrupt On Setup (IOS). This bit is used on control type endpoints to indicate if USBINT is set in response to a setup being received.
14-0	Reserved. Bits reserved for future use and should be set to zero.

42.5.5.1.2 Transfer Overlay-Endpoint Queue Head

The seven DWords in the overlay area represent a transaction working space for the device controller.

The general operational model is that the device controller can detect whether the overlay area contains a description of an active transfer. If it does not contain an active transfer, then it will not read the associated endpoint.

After an endpoint is readied, the dTD will be copied into this queue head overlay area by the device controller. Until a transfer is expired, software must not write the queue head overlay area or the associated transfer descriptor. When the transfer is complete, the device controller will write the results back to the original transfer descriptor and advance the queue.

See dTD for a description of the overlay fields.

42.5.5.1.3 Current dTD Pointer

The current dTD pointer is used by the device controller to locate the transfer in progress. This word is for Device Controller (hardware) use only and should not be modified by DCD software.

The following table describes the dTD Pointer.

Table 42-59. Next dTD Pointer

Bit	Description
31-5	Current dTD. This field is a pointer to the dTD that is represented in the transfer overlay area. This field will be modified by the Device Controller to next dTD pointer during endpoint priming or queue advance.
4-0	Reserved. Bit reserved for future use and should be set to zero.

42.5.5.1.4 Set-up Buffer

The set-up buffer is dedicated storage for the 8-byte data that follows a set-up PID.

NOTE

Each endpoint has a TX and an RX dQH associated with it, and only the RX queue head is used for receiving setup data packets.

The following table describes the Multiple Mode Control.

Table 42-60. Multiple Mode Control (HCCPARAMS)

DWord	Bits	Description
1	31-0	Setup Buffer 0. This buffer contains bytes 3 to 0 of an incoming setup buffer packet and is written by the device controller to be read by software.
2	31-0	Setup Buffer 1. This buffer contains bytes 7 to 4 of an incoming setup buffer packet and is written by the device controller to be read by software.

42.5.5.2 Endpoint Transfer Descriptor (dTD)

The dTD describes to the device controller the location and quantity of data to be sent/ received for a given transfer.

The DCD should not attempt to modify any field in an active dTD except the Next Like Pointer, which should only be modified as described in section Managing Transfers with Transfer Descriptors.

Table below shows the Endpoint Transfer Descriptor (dTD).

Table 42-61. Endpoint Transfer Descriptor (dTD)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Next Link Pointer																											0				T
0	Total Bytes															ioc	0			MultO		0		Status							
Buffer Pointer (Page 0)																				Current Offset											
Buffer Pointer (Page 1)																				0	Frame Number										
Buffer Pointer (Page 2)																				Reserved											
Buffer Pointer (Page 3)																				Reserved											
Buffer Pointer (Page 4)																				Reserved											

Host Controller Read/Write

Host Controller Read Only

The following table describes the dTD Pointer.

Table 42-62. Next dTD Pointer

Bit	Description
31-5	Next Transfer Element Pointer. This field contains the physical memory address of the next dTD to be processed. The field corresponds to memory address signals [31:5], respectively.
4-1	Reserved. Bits reserved for future use and should be set to zero.
0	Terminate (T). 1=pointer is invalid. 0=Pointer is valid (points to a valid Transfer Element Descriptor). This bit indicates to the Device Controller that there are no more valid entries in the queue.

The following table describes the dTD Token.

Table 42-63. dTD Token

Bit	Description
31	Reserved. Bit reserved for future use and should be set to zero.
30-16	Total Bytes. This field specifies the total number of bytes to be moved with this transfer descriptor. This field is decremented by the number of bytes actually moved during the transaction and only on the successful completion of the transaction. The maximum value software may store in the field is 5*4K (5000H). This is the maximum number of bytes 5 page pointers can access. Although it is possible to create a transfer up to 20K this assumes the 1st offset into the first page is 0. When the offset cannot be predetermined, crossing past the 5th page can be guaranteed by limiting the total bytes to 16K**. Therefore, the maximum recommended transfer is 16K (4000H). If the value of the field is zero when the host controller fetches this transfer descriptor (and the active bit is set), the device controller executes a zero-length transaction and retires the transfer descriptor.

Table continues on the next page...

Table 42-63. dTD Token (continued)

	It is not a requirement for IN transfers that Total Bytes To Transfer be an even multiple of *Maximum Packet Length*. If software builds such a transfer descriptor for an IN transfer, the last transaction will always be less that *Maximum Packet Length*.
15	Interrupt On Complete (IOC). This bit is used to indicate if USBINT is to be set in response to device controller being finished with this dTD.
14-12	Reserved. Bits reserved for future use and should be set to zero.
11-10	Multiplier Override (MultO). This field can be used for transmit ISO's (ie. ISO-IN) to override the multiplier in the QH. This field must be zero for all packet types that are not transmit-ISO. Example: if QH.multiplier = 3; Maximum packet size = 8; Total Bytes = 15; MultiO = 0 [default] Three packets are sent: {Data2(8); Data1(7); Data0(0)} if QH.multiplier = 3; Maximum packet size = 8; Total Bytes = 15; MultiO = 2 Two packets are sent: {Data1(8); Data0(7)} For maximal efficiency, software should compute MultO = greatest integer of (Total Bytes / Max. Packet Size) except for the case when Total Bytes = 0; then MultO should be 1. Note: Non-ISO and Non-TX endpoints must set MultO = "00".
9-8	Reserved. Bits reserved for future use and should be set to zero.
7-0	Status. This field is used by the Device Controller to communicate individual command execution states back to the Device Controller software. This field contains the status of the last transaction performed on this qTD. The bit encodings are: Bit Status Field Description 7 Active. 6 Halted. 5 Data Buffer Error. 3 Transaction Error. 4, 2, 0 Reserved.

The table below describes the dTD Buffer Page Pointer List.

Table 42-64. dTD Buffer Page Pointer List

Bit	Description
31-12	Buffer Pointer. Selects the page offset in memory for the packet buffer. Non virtual memory systems will typically set the buffer pointers to a series of incrementing integers.
0,11-0	Current Offset. Offset into the 4kb buffer where the packet is to begin.
1,10-0	Frame Number. Written by the device controller to indicate the frame number in which a packet finishes. This is typically be used to correlate relative completion times of packets on an ISO endpoint.

42.5.6 Device Operational Model

The function of the device operation is to transfer a request in the memory image to and from the Universal Serial Bus.

Using a set of linked list transfer descriptors, pointed to by a queue head, the device controller will perform the data transfers. The following sections explain the use of the device controller from the device controller driver (DCD) point-of-view and further describe how specific USB bus events relate to status changes in the device controller programmer's interface.

42.5.6.1 Device Controller Initialization

After hardware reset, the device is disabled until the Run/Stop bit is set to a '1'. In the disabled state, the pull-up on the USB D+ is not active which prevents an attach event from occurring. At a minimum, it is necessary to have the queue heads setup for endpoint zero before the device attach occurs.

Shortly after the device is enabled, a USB reset will occur followed by setup packet arriving at endpoint 0. A Queue head must be prepared so that the device controller can store the incoming setup packet.

In order to initialize a device, the software should perform the following steps:

- Set Controller Mode in the USB.USBMODE register to device mode.

NOTE

Transitioning from host mode to device mode requires a device controller reset before modifying USB.USBMODE.

- Allocate and Initialize device queue heads in system memory.
  - Minimum: Initialize device queue heads 0 Tx & 0 Rx.

NOTE

All device queue heads for control endpoints must be initialized before the endpoint is enabled. Non-Control device queue heads before the endpoint can be used.

  - For information on device queue heads, refer to section Device Data Structures.
- Configure USB.ENDPOINTLISTADDR Pointer.
  - For additional information on USB.ENDPOINTLISTADDR, refer to the register table.
- Enable the microprocessor interrupt associated with the USB core.
  - Recommended: enable all device interrupts including: USBINT, USBERRINT, Port Change Detect, USB Reset Received, DCSuspend.
  - For a list of available interrupts refer to the Interrupt Enable Register (USB_*n*USBINTR) and the USB Status Register (USB_*n*USBSTS) register tables.
- Set Run/Stop bit to Run Mode.

- After the Run bit is set and the device is connected to a host, a Bus Reset will be issued by host downstream port. The DCD must monitor the reset event and adjust the software state as described in the Bus Reset section of the Port State and Control section below.

NOTE

Endpoint 0 is designed as a control endpoint only and does not need to be configured using ENDPTCTRL0 register.

It is also not necessary to prime Endpoint 0 initially because the first packet received will always be a setup packet. The contents of the first setup packet will require a response in accordance with USB device framework (Chapter 9) command set.

42.5.6.2 Port State and Control

From a chip or system reset, the device controller enters the *powered* state. A transition from the *powered* state to the *attach* state occurs when the Run/Stop bit is set to a '1'.

After receiving a reset on the bus, the port will enter the *defaultFS or defaultHS* state in accordance with the reset protocol described in Appendix C.2 of the USB Specification Rev. 2.0.

The following state diagram depicts the state of a USB 2.0 device.

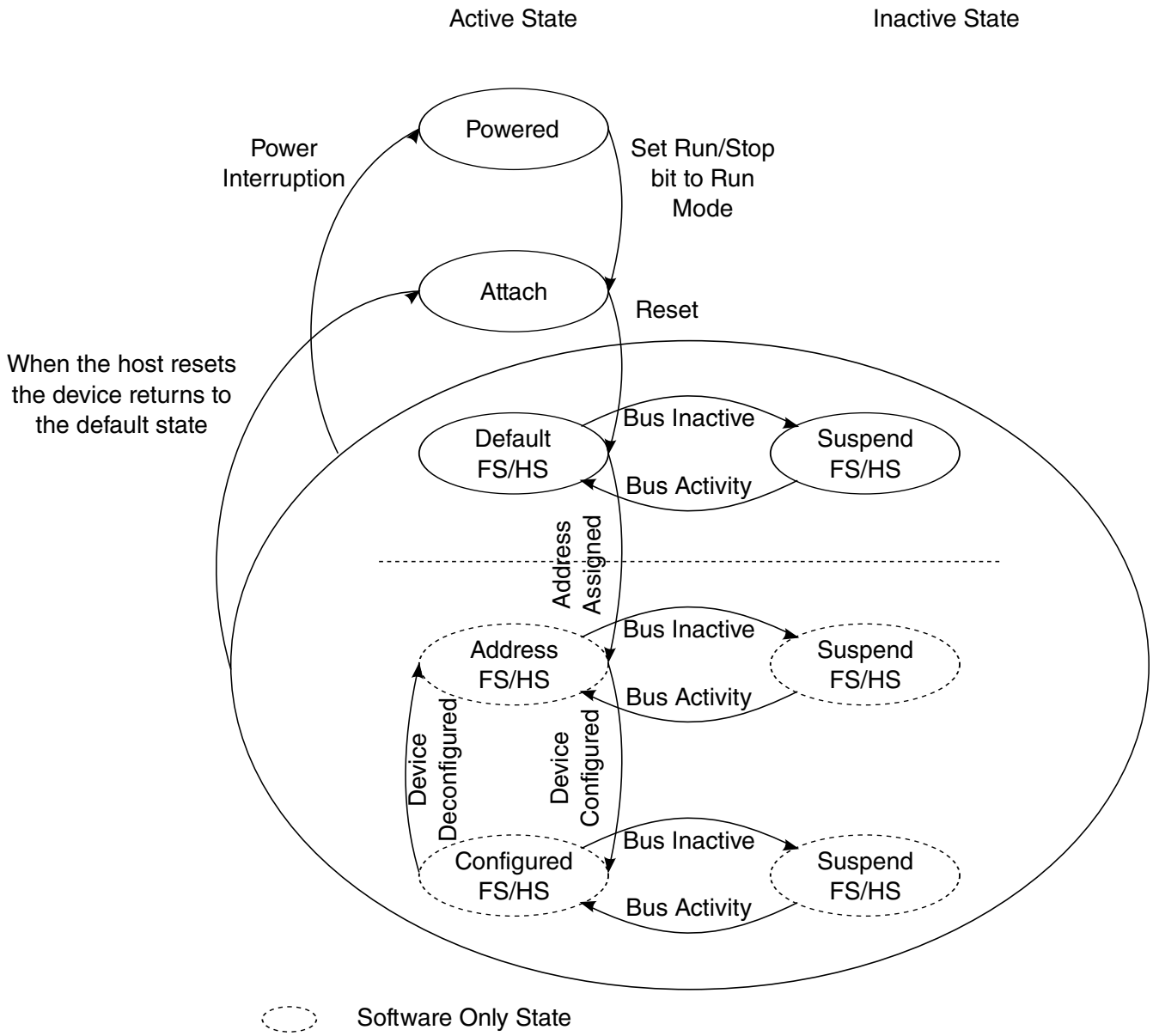


Figure 42-28. Device State Diagram

States *powered, attach, defaultFS/HS, suspendFS/HS* are implemented in the device controller and are communicated to the DCD using the following status bits:

The following table describes the Device Controller State Information Bits.

Table 42-65. Device Controller State Information Bits

Bit	Register
DCSuspend	USB Status Register (USB_*n*USBSTS)
USB Reset Received	USB Status Register (USB_*n*USBSTS)
Port Change Detect	USB Status Register (USB_*n*USBSTS)
High-Speed Port	Port Status & Control (USB_*n*PORTSC1)

It is the responsibility of the DCD to maintain a state variable to differentiate between the *DefaultFS/HS* state and the *Address/Configured* states. Change of state from *Default* to *Address* and the *Configured* states is part of the enumeration process described in the device framework section of the USB 2.0 Specification.

As a result of entering the *Address* state, the device address register (DEVICEADDR) must be programmed by the DCD.

Entry into the *Configured* indicates that all endpoints to be used in the operation of the device have been properly initialized by programming the USB_UOG_ENDPTCTRLx registers and initializing the associated queue heads.

42.5.6.2.1 Bus Reset

A bus reset is used by the host to initialize downstream devices.

When a bus reset is detected, the device controller will renegotiate its attachment speed, reset the device address to 0, and notify the DCD by interrupt (assuming the USB Reset Interrupt Enable is set). After a reset is received, all endpoints (except endpoint 0) are disabled and any primed transactions will be cancelled by the device controller. The concept of priming will be clarified below, but the DCD must perform the following tasks when a reset is received:

Clear all setup token semaphores by reading the Endpoint Status (USB_*n*ENDPTSTAT) register and writing the same value back to the Endpoint Status (USB_*n*ENDPTSTAT) register.

Clear all the endpoint complete status bits by reading the Endpoint Complete (USB_*n*ENDPTCOMPLETE) register and writing the same value back to the Endpoint Complete (USB_*n*ENDPTCOMPLETE) register.

Cancel all primed status by waiting until all bits in the Endpoint Prime (USB_*n*ENDPTPRIME) are 0 and then writing 0xFFFFFFFF to Endpoint Flush (USB_*n*ENDPTFLUSH).

Read the reset bit in the Port Status & Control (USB_*n*PORTSC1) register and make sure that it is still active. A USB reset will occur for a minimum of 3 ms and the DCD must reach this point in the reset cleanup before end of the reset occurs, otherwise a hardware reset of the device controller is recommended (rare.)

- A hardware reset can be performed by writing a one to the device controller reset bit in the USBCMD reset. Note: a hardware reset will cause the device to detach from the bus by clearing the Run/Stop bit. Thus, the DCD must completely re-initialize the device controller after a hardware reset.

Free all allocated dTDs because they will no longer be executed by the device controller. If this is the first time the DCD is processing a USB reset event, then it is likely that no dTDs have been allocated.

At this time, the DCD may release control back to the OS because no further changes to the device controller are permitted until a Port Change Detect is indicated.

After a Port Change Detect, the device has reached the default state and the DCD can read the Port Status & Control (USB_*n*PORTSC1) to determine if the device is operating in FS or HS mode. At this time, the device controller has reached normal operating mode and DCD can begin enumeration according to the USB Chapter 9 - Device Framework.

NOTE

The device DCD may use the FS/HS mode information to determine the bandwidth mode of the device

In some applications, it may not be possible to enable one or more pipes while in FS mode. *Beyond the data rate issue, there is no difference in DCD operation between FS and HS modes.*

42.5.6.2.2 Suspend/Resume

The detials of suspend and resume are explained in these sections.

42.5.6.2.2.1 Suspend

Suspend Description

In order to conserve power, USB devices automatically enter the suspended state when the device has observed no bus traffic for a specified period. When suspended, the USB device maintains any internal status, including its address and configuration. Attached devices must be prepared to suspend at any time they are powered, regardless of if they have been assigned a non-default address, are configured, or neither. Bus activity may cease due to the host entering a suspend mode of its own. In addition, a USB device shall also enter the suspended state when the hub port it is attached to is disabled.

A USB device exits suspend mode when there is bus activity. A USB device may also request the host to exit suspend mode or selective suspend by using electrical signaling to indicate remote wakeup. The ability of a device to signal remote wakeup is optional. If the USB device is capable of remote wakeup signaling, the device must support the ability of the host to enable and disable this capability. When the device is reset, remote wakeup signaling must be disabled.

Suspend Operational Model

The device controller moves into the suspend state when suspend signaling is detected or activity is missing on the upstream port for more than a specific period. After the device controller enters the suspend state, the DCD is notified by an interrupt (assuming *DC Suspend Interrupt* is enabled). When the *DCSuspend* bit in the Port Status & Control (USB_*n*PORTSC1) is set to a '1', the device controller is suspended.

DCD response when the device controller is suspended is application specific and may involve switching to low power operation.

Information on the bus power limits in suspend state can be found in USB 2.0 specification.

42.5.6.2.2.2 Resume

If the device controller is suspended, its operation is resumed when any non-idle signaling is received on its upstream facing port. In addition, the device can signal the system to resume operation by forcing resume signaling to the upstream port.

Resume signaling is sent upstream by writing a '1' to the Resume bit in the in the Port Status & Control (USB_*n*PORTSC1) while the device is in suspend state. Sending resume signal to an upstream port should cause the host to issue resume signaling and bring the suspended bus segment (one more devices) back to the active condition.

NOTE

Before resume signaling can be used, the host must enable it by using the Set Feature command defined in device framework (chapter 9) of the USB 2.0 Specification.

42.5.6.3 Managing Endpoints

The USB 2.0 specification defines an endpoint, also called a device endpoint or an address endpoint as a uniquely addressable portion of a USB device that can source or sink data in a communications channel between the host and the device.

The endpoint address is specified by the combination of the endpoint number and the endpoint direction.

The channel between the host and an endpoint at a specific device represents a data pipe. Endpoint 0 for a device is always a *control* type data channel used for device discovery and enumeration. Other types of endpoints support by USB include *bulk, interrupt*, and *isochronous*. Each endpoint type has specific behavior related to packet response and error handling. More detail on endpoint operation can be found in the USB 2.0 specification.

The USB OTG device controller hardware supports up to 8 endpoint numbers.

Each endpoint direction is essentially independent and can be configured with differing behavior in each direction. For example, the DCD can configure endpoint 1-IN to be a bulk endpoint and endpoint 1-OUT to be an isochronous endpoint. This helps to conserve the total number of endpoints required for device operation. The only exception is that control endpoints must use both directions on a single endpoint number to function as a control endpoint. Endpoint 0 is, for example, is always a control endpoint and uses the pair of directions.

Each endpoint direction requires a *queue head* allocated in memory. To support the 8 endpoint numbers, 16 *queue heads* are required. The operation of an endpoint and use of *queue heads* are described later in this document.

42.5.6.3.1 Endpoint Initialization

After hardware reset, all endpoints except endpoint zero are uninitialized and disabled. The DCD must configure and enable each endpoint by writing to configuration bit in the USB_UOG_ENDPTCTRLx register.

Each 32-bit USB_UOG_ENDPTCTRLx is split into an upper and lower half. The lower half of USB_UOG_ENDPTCTRLx is used to configure the receive or OUT endpoint and the upper half is likewise used to configure the corresponding transmit or IN endpoint. Control endpoints must be configured the same in both the upper and lower half of the USB_UOG_ENDPTCTRLx register otherwise the behavior is undefined. The following table shows how to construct a configuration word for endpoint initialization. The following table shows the fields and values for the Device Controller Endpoint initialization.

Table 42-66. Device Controller Endpoint Initialization

Field	Value
Data Toggle Reset	1
Data Toggle Inhibit	0
Endpoint Type	00 Control 01 Isochronous 10 Bulk 11 Interrupt
Endpoint Stall	0

42.5.6.3.2 Stalling

There are two occasions where the device controller may need to return to the host a STALL.

The first occasion is the functional stall, which is a condition set by the DCD as described in the USB 2.0 device framework. A functional stall is only used on non-control endpoints and can be enabled in the device controller by setting the endpoint stall bit in the USB_UOG_ENDPTCTRLx register associated with the given endpoint and the given direction. In a functional stall condition, the device controller will continue to return STALL responses to all transactions occurring on the respective endpoint and direction until the endpoint stall bit is cleared by the DCD.

A protocol stall, unlike a function stall, is used on control endpoints is automatically cleared by the device controller at the start of a new control transaction (setup phase). When enabling a protocol stall, the DCD should enable the stall bits (both directions) as a pair. A single write to the USB_UOG_ENDPTCTRLx register can ensure that both stall bits are set at the same instant.

NOTE

Any write to the USB_UOG_ENDPTCTRLx register during operational mode must preserve the endpoint type field (that is, perform a read-modify-write).

The following table shows the response matrix for the Device Controller Stall.

Table 42-67. Device Controller Stall Response Matrix

USB Packet	Endpoint Stall Bit.	Effect on STALL bit.	USB Response
SETUP packet received by a non-control endpoint.	N/A	None.	STALL
IN/OUT/PING packet received by a non-control endpoint.	'1'	None.	STALL
IN/OUT/PING packet received by a non-control endpoint.	'0'	None.	ACK/ NAK/ NYET
SETUP packet received by a control endpoint.	N/A	Cleared	ACK
IN/OUT/PING packet received by a control endpoint	'1'	None	STALL
IN/OUT/PING packet received by a control endpoint.	'0'	None.	ACK/ NAK/ NYET

42.5.6.3.3 Data Toggle

Data toggle is a mechanism to maintain data coherency between host and device for any given data pipe.

For more information on data toggle, refer to the USB 2.0 specification.

42.5.6.3.3.1 Data Toggle Reset

The DCD may reset the data toggle state bit and cause the data toggle sequence to reset in the device controller by writing a '1' to the data toggle reset bit in the USB_UOG_ENDPTCTRLx register.

This should only be necessary when configuring/initializing an endpoint or returning from a STALL condition.

42.5.6.3.3.2 Data Toggle Inhibit

NOTE

This feature is for test purposes only and should never be used during normal device controller operation.

Setting the *data toggle Inhibit bit* active ('1') causes the device controller to ignore the data toggle pattern that is normally sent and accept all incoming data packets regardless of the data toggle state.

In normal operation, the device controller checks the DATA0/DATA1 bit against the data toggle to determine if the packet is valid. If Data PID does not match the data toggle state bit maintained by the device controller for that endpoint, the Data toggle is considered not valid. If the data toggle is not valid, the device controller assumes the packet was already received and discards the packet (not reporting it to the DCD). To prevent the host controller from re-sending the same packet, the device controller will respond to the error packet by acknowledging it with either an ACK or NYET response.

42.5.6.3.3.3 Priming Transmit Endpoints

Priming a transmit endpoint will cause the device controller to fetch the device transfer descriptor (dTD) for the transaction pointed to by the device queue head (dQH).

After the dTD is fetched, it will be stored in the dQH until the device controller completes the transfer described by the dTD. Storing the dTD in the dQH allows the device controller to fetch the operating context needed to handle a request from the host without the need to follow the linked list, starting at the dQH when the host request is received.

After the device has loaded the dTD, the leading data in the packet is stored in a FIFO in the device controller. This FIFO is split into virtual channels so that the leading data can be stored for any endpoint up to the maximum number of endpoints configured at device synthesis time.

After a priming request is complete, an endpoint state of primed is indicated in the USB_UOG_ENDPTSTATUS register. For a primed transmit endpoint, the device controller can respond to an IN request from the host and meet the stringent bus turnaround time of High Speed USB.

Because only the leading data is stored in the device controller FIFO, it is necessary for the device controller to begin filling in behind leading data after the transaction starts. The FIFO must be sized to account for the maximum latency that can be incurred by the system memory bus. More information about FIFO sizing is presented in section .

42.5.6.3.3.4 Priming Receive Endpoints

Priming receive endpoints is identical to priming of transmit endpoints from the point of view of the DCD. At the device controller the major difference in the operational model is that there is no data movement of the leading packet data simply because the data is to be received from the host.

Note as part of the architecture, the FIFO for the receive endpoints is not partitioned into multiple channels like the transmit FIFO. Thus, the size of the RX FIFO does not scale with the number of endpoints.

42.5.6.4 Operational Model For Packet Transfers

All transactions on the USB bus are initiated by the host and in turn, the device must respond to any request from the host within the turnaround time stated in the USB 2.0 Specification.

At USB 1.1 Full or Low Speed rates, this turnaround time was significant and the USB 1.1 device controllers were architected so that the device controller could access main memory or interrupt a host protocol processor in order to respond to the USB 1.1 transaction. The architecture of the USB 2.0 device controller must be different because same methods will not meet USB 2.0 High-speed turnaround time requirements by simply increasing clock rate.

A USB host will send requests to the device controller in an order that can not be precisely predicted as a single pipeline, so it is not possible to prepare a single packet for the device controller to execute. However, the order of packet requests is predictable when the endpoint number and direction is considered. For example, if endpoint 3

(transmit direction) is configured as a bulk pipe, then we can expect the host will send IN requests to that endpoint. This device controller is architected in such a way that it can prepare packets for each endpoint/direction in anticipation of the host request. The process of preparing the device controller to send or receive data in response to host initiated transaction on the bus is referred to as "priming" the endpoint. This term will be used throughout the following documentation to describe the device controller operation so the DCD can be architected properly use priming. Further, note that the term "flushing" is used to describe the action of clearing a packet that was queued for execution.

42.5.6.4.1 Interrupt/Bulk Endpoint Operational Model

The behaviors of the device controller for interrupt and bulk endpoints are identical.

All valid IN and OUT transactions to bulk pipes will handshake with a NAK unless the endpoint had been primed. Once the endpoint has been primed, data delivery will commence.

A dTD will be retired by the device controller when the packets described in the transfer descriptor have been completed. Each dTD describes N packets to be transferred according to the USB Variable Length transfer protocol. The formula and table on the following page describe how the device controller computes the number and length of the packets to be sent/received by the USB vary according to the total number of bytes and maximum packet length.

With Zero Length Termination (ZLT) = 0

N = INT(Number Of Bytes/Max. Packet Length) + 1

With Zero Length Termination (ZLT) = 1

N = MAXINT(Number Of Bytes/Max. Packet Length)

Table 42-68. Variable Length Transfer Protocol Example (ZLT = 0)

Bytes (dTD)	Max. Packet Length (dQH)	N	P1	P2	P3
511	256	2	256	255	
512	256	3	256	256	0
512	512	2	512	0	

Table 42-69. Variable Length Transfer Protocol Example (ZLT = 1)

Bytes (dTD)	Max. Packet Length (dQH)	N	P1	P2	P3
511	256	2	256	255	
512	256	2	256	256	
512	512	1	512		

NOTE

The MULT field in the dQH must be set to "00" for bulk, interrupt, and control endpoints.

TX-dTD is complete when:

- All packets described dTD were successfully transmitted. *** Total bytes in dTD will equal zero when this occurs.

RX-dTD is complete when:

- All packets described in dTD were successfully received. *** Total bytes in dTD will equal zero when this occurs.
- A short packet (number of bytes < maximum packet length) was received. *** This is a successful transfer completion; DCD must check Total Bytes in dTD to determine the number of bytes that are remaining. From the total bytes remaining in the dTD, the DCD can compute the actual bytes received.
- A long packet was received (number of bytes > maximum packet size) OR (total bytes received > total bytes specified). *** This is an error condition. The device controller will discard the remaining packet, and set the Buffer Error bit in the dTD. In addition, the endpoint will be flushed and the USBERR interrupt will become active.

On the successful completion of the packet(s) described by the dTD, the active bit in the dTD will be cleared and the next pointer will be followed when the Terminate bit is clear. When the Terminate bit is set, the device controller will flush the endpoint/direction and cease operations for that endpoint/direction.

On the unsuccessful completion of a packet (see long packet above), the dQH will be left pointing to the dTD that was in error. In order to recover from this error condition, the DCD must properly reinitialize the dQH by clearing the active bit and update the nextTD pointer before attempting to re-prime the endpoint.

NOTE

All packet level errors such as a missing handshake or CRC error will be retried automatically by the device controller.

There is no required interaction with the DCD for handling such errors.

42.5.6.4.1.1 Interrupt/Bulk Endpoint Bus Response Matrix

The table below shows the response matrix for Interrput/Bulk Endpoint Bus.

Table 42-70. Interrupt/Bulk Endpoint Bus Response Matrix

	Stall	Not Primed	Primed	Underflow	Overflow
Setup	Ignore	Ignore	Ignore	N/A	N/A
In	STALL	NAK	Transmit	BS Error	N/A
Out	STALL	NAK	Receive + NYET/ACK	N/A	NAK
Ping	STALL	NAK	ACK	N/A	N/A
Invalid	Ignore	Ignore	Ignore	Ignore	Ignore

NOTE

BS Error = Force Bit Stuff Error

NYET/ACK - NYET unless the Transfer Descriptor has packets remaining according to the USB variable length protocol then ACK.

SYSERR - System error should never occur when the latency FIFOs are correctly sized and the DCD is responsive.

42.5.6.4.2 Control Endpoint Operation Model

This section details the setup phase, data phase, status phase, and the control endpoint bus response matrix.

42.5.6.4.2.1 Setup Phase

All requests to a control endpoint begin with a setup phase followed by an optional data phase and a required status phase. The device controller will always accept the setup phase unless the setup lockout is engaged.

The setup lockout will engage so that future setup packets are ignored. Lockout of setup packets ensures that while software is reading the setup packet stored in the queue head, that data is not written as it is being read potentially causing an invalid setup packet.

The setup lockout mechanism can be disabled and a new tripwire type semaphore will ensure that the setup packet payload is extracted from the queue head without being corrupted be an incoming setup packet. This is the preferred behavior because ignoring repeated setup packets due to long software interrupt latency would be a compliance issue.

- Disable Setup Lockout by writing 1 to Setup Lockout Mode (SLOM) in USB Device Mode (USB_*n*USBMODE). (once at initialization). Setup lockout is not necessary when using the tripwire as described below.

NOTE

Leaving the Setup Lockout Mode As 0 will result in pre-2.3 hardware behavior.

- After receiving an interrupt and inspecting Endpoint Setup Status (USB_*n*ENDPTSETUPSTAT) to determine that a setup packet was received on a particular pipe:
  a. Write 1 to clear corresponding bit Endpoint Setup Status (USB_*n*ENDPTSETUPSTAT).
  b. Write 1 to Setup Tripwire (SUTW) in USB Command Register (USB_*n*USBCMD) register.
  c. Duplicate contents of dQH.SetupBuffer into local software byte array.
  d. Read Setup TripWire (SUTW) in USB Command Register (USB_*n*USBCMD) register. (if set - continue; if cleared - goto 2)
  e. Write 0 to clear Setup Tripwire (SUTW) in USB Command Register (USB_*n*USBCMD) register.
  f. Process setup packet using local software byte array copy and execute status/ handshake phases.

NOTE

After receiving a new setup packet the status and/or handshake phases may still be pending from a previous control sequence. These should be flushed & deallocated before linking a new status and/or handshake dTD for the most recent setup packet.

42.5.6.4.2.2 Data Phase

Following the setup phase, the DCD must create a device transfer descriptor for the data phase and prime the transfer.

After priming the packet, the DCD must verify a new setup packet has not been received by reading the USB.ENDPTSETUPSTAT register immediately verifying that the prime had completed. A prime will complete when the associated bit in the Endpoint Prime (USB_*n*ENDPTPRIME) register is zero and the associated bit in the Endpoint Status (USB_*n*ENDPTSTAT) register is a one. If a prime fails, ie. The Endpoint Prime (USB_*n*ENDPTPRIME) bit goes to zero and the Endpoint Status (USB_*n*ENDPTSTAT) bit is not set, then the prime has failed. This can only be due to improper setup of the dQH, dTD or a setup arriving during the prime operation. If a new setup packet is indicated after the ENDPTPRIME bit is cleared, then the transfer descriptor can be freed and the DCD must reinterpret the setup packet.

Should a setup arrive after the data stage is primed, the device controller will automatically clear the prime status (Endpoint Status (USB_*n*ENDPTSTAT)) to enforce data coherency with the setup packet.

NOTE

- The MULT field in the dQH must be set to "00" for bulk, interrupt, and control endpoints.
- Error handling of data phase packets is the same as bulk packets described previously.

42.5.6.4.2.3 Status Phase

Similar to the data phase, the DCD must create a transfer descriptor (with byte length equal zero) and prime the endpoint for the status phase.

The DCD must also perform the same checks of the USB.ENDPTSETUPSTAT as described above in the data phase.

NOTE

- The MULT field in the dQH must be set to 00 for bulk, interrupt, and control endpoints.
- Error handling of data phase packets is the same as bulk packets described previously.

42.5.6.4.2.4 Control Endpoint Bus Response Matrix

Shown in the following table is the device controller response to packets on a control endpoint according to the device controller state.

The table below shows the response matrix for the Control Endpoint Bus.

Table 42-71. Control Endpoint Bus Response Matrix

Token Type	Endpoint State					Setup Lockout
	Stall	Not Primed	Primed	Underflow	Overflow	
Setup	ACK	ACK	ACK	N/A	SYSERR	
In	STALL	NAK	Transmit	BS Error	N/A	N/A
Out	STALL	NAK	Receive + NYET/ACK	N/A	NAK	N/A
Ping	STALL	NAK	ACK	N/A	N/A	N/A
Invalid	Ignore	Ignore	Ignore	Ignore	Ignore	Ignore

BS Error = Force Bit Stuff Error

NYET/ACK - NYET unless the Transfer Descriptor has packets remaining according to the USB variable length protocol then ACK.

SYSERR - System error should never occur when the latency FIFOs are correctly sized and the DCD is responsive.

42.5.6.4.3 Isochronous Endpoint Operational Model

Isochronous endpoints are used for real-time scheduled delivery of data and their operational model is significantly different than the host throttled Bulk, Interrupt, and Control data pipes.

Real time delivery by the device controller will be accomplished by the following:

- Exactly MULT Packets per (micro) Frame are transmitted/received. Note: MULT is a two-bit field in the device Queue Head. The variable length packet protocol is not used on isochronous endpoints.
- NAK responses are not used. Instead, zero length packets are sent in response to an IN request to an unprimed endpoints. For unprimed RX endpoints, the response to an OUT transaction is to ignore the packet within the device controller.
- Prime requests always schedule the transfer described in the dTD for the next (micro) frame. If the ISO-dTD is still active after that frame, then the ISO-dTD will be held ready until executed or canceled by the DCD.

An EHCI compatible host controller uses the periodic frame list to schedule data exchanges to Isochronous endpoints. The operational model for device mode does not use such a data structure. Instead, the same dTD used for Control/Bulk/Interrupt endpoints is also used for isochronous endpoints. The difference is in the handling of the dTD.

The first difference between bulk and ISO-endpoints is that priming an ISO-endpoint is a delayed operation such that an endpoint will become primed only after a SOF is received. After the DCD writes the prime bit, the prime bit will be cleared as usual to indicate to software that the device controller completed a priming the dTD for transfer. Internal to the design, the device controller hardware masks that prime start until the next frame boundary. This behavior is hidden from the DCD but occurs so that the device controller can match the dTD to a specific (micro)frame.

Another difference with isochronous endpoints is that the transaction must wholly complete in a (micro)frame. Once an ISO transaction is started in a (micro)frame it will retire the corresponding dTD when MULT transactions occur or the device controller finds a fulfillment condition.

The transaction error bit set in the status field indicates a fulfillment error condition. When a fulfillment error occurs, the frame after the transfer failed to complete wholly, the device controller will force retire the ISO-dTD and move to the next ISO-dTD.

It is important to note that fulfillment errors are only caused due to partially completed packets. If no activity occurs to a primed ISO-dTD, the transaction will stay primed indefinitely. This means it is up to software discard transmit ISO-dTDs that pile up from a failure of the host to move the data.

Finally, the last difference with ISO packets is in the data level error handling. When a CRC error occurs on a received packet, the packet is not retried similar to bulk and control endpoints. Instead, the CRC is noted by setting the *Transaction Error* bit and the data is stored as usual for the application software to sort out.

- TX Packet Retired
  - MULT counter reaches zero.
  - Fulfillment Error [*Transaction Error* bit is set]
    - # Packets Occurred > 0 AND # Packets Occurred < MULT

NOTE

For TX-ISO, MULT Counter can be loaded with a lesser value in the dTD Multiplier Override field in hardware versions 2.3 and later. If the Multiplier Override is zero, the MULT Counter is initialized to the Multiplier in the QH.

- RX Packet Retired:
  - MULT counter reaches zero.
  - Non-MDATA Data PID is received**
    - ** Exit criteria only valid in hardware version 2.3 or later. Previous to hardware version 2.3, any PID sequence that did not match the MULT field exactly would be flagged as a transaction error due to PID mismatch or fulfillment error.
  - Overflow Error:
    - Packet received is > maximum packet length. [*Buffer Error* bit is set]
    - Packet received exceeds total bytes allocated in dTD. [*Buffer Error* bit is set]
  - Fulfillment Error [*Transaction Error* bit is set]
    - # Packets Occurred > 0 AND # Packets Occurred < MULT
  - CRC Error [*Transaction Error* bit is set]

NOTE

For ISO, when a dTD is retired, the next dTD is primed for the next frame. For continuous (micro)frame to (micro)frame operation the DCD should ensure that the dTD linked-list is out ahead of the device controller by at least two (micro)frames.

42.5.6.4.3.1 Isochronous Pipe Synchronization

When it is necessary to synchronize an isochronous data pipe to the host, the (micro) frame number (USB_UOG_FRINDEX register) can be used as a marker.

To cause a packet transfer to occur at a specific (micro) frame number [N], the DCD should interrupt on SOF during frame N-1. When the USB_UOG_FRINDEX=N-1, the DCD must write the prime bit. The device controller will prime the isochronous endpoint in (micro) frame N-1 so that the device controller will execute delivery during (micro) frame N.

NOTE

Priming an endpoint towards the end of (micro) frame N-1 will not guarantee delivery in (micro) frame N. The delivery may actually occur in (micro) frame N+1 if device controller does not have enough time to complete the prime before the SOF for packet N is received.

42.5.6.4.3.2 Isochronous Endpoint Bus Response Matrix

The following table shows the response matrix for the Isochronous Endpoint Bus.

Table 42-72. Isochronous Endpoint Bus Response Matrix

	Stall	Not Primed	Primed	Underflow	Overflow
Setup	STALL	STALL	STALL	N/A	N/A
In	NULL Packet	NULL Packet	Transmit	BS Error	N/A
Out	Ignore	Ignore	Receive	N/A	Drop Packet
Ping	Ignore	Ignore	Ignore	Ignore	Ignore
Invalid	Ignore	Ignore	Ignore	Ignore	Ignore

1. BS Error = Force Bit Stuff Error

   NULL Packet = Zero Length Packet

42.5.6.5 Managing Queue Heads

The following figure shows the End Point Queue Head.

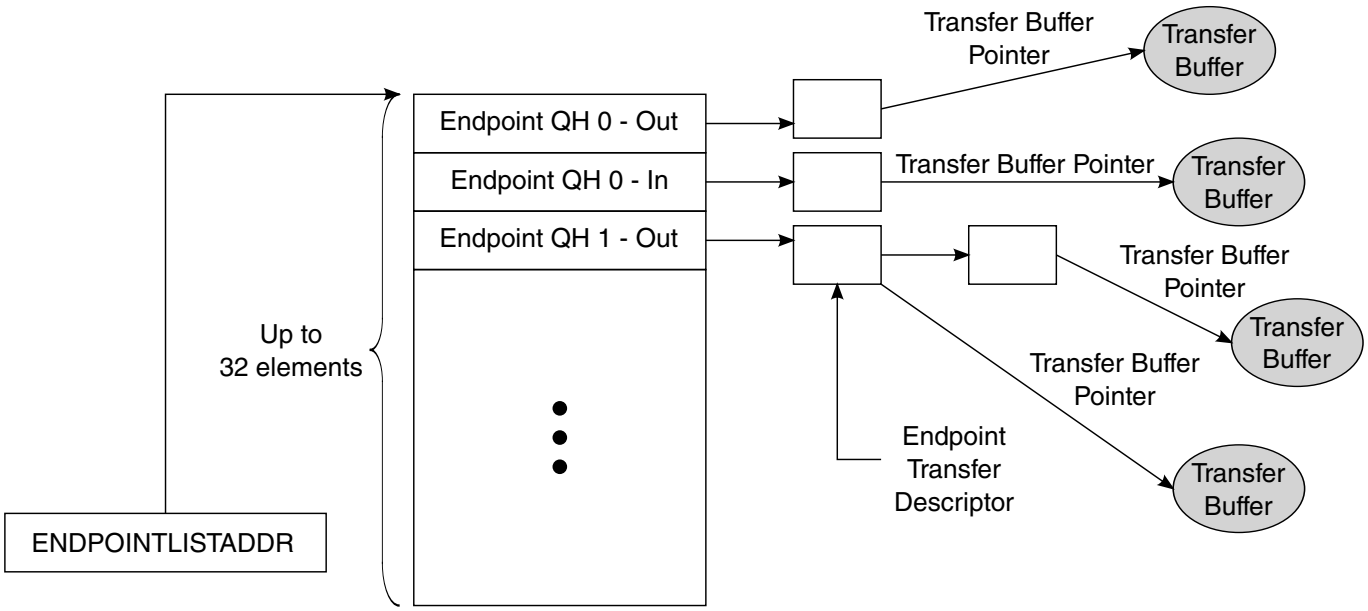


Figure 42-29. End Point Queue Head Diagram

The device queue head (dQH) points to the linked list of transfer tasks, each depicted by the device Transfer Descriptor (dTD). An area of memory pointed to by USB.ENDPOINTLISTADDR contains a group of all dQH's in a sequential list as shown in Figure 42-29. The even elements in the list of dQH's are used for receive endpoints (OUT/SETUP) and the odd elements are used for transmit endpoints (IN/INTERRUPT). Device transfer descriptors are linked head to tail starting at the queue head and ending at a terminate bit. Once the dTD has been retired, it will no longer be part of the linked list from the queue head. Therefore, software is required to track all transfer descriptors because pointers will no longer exist within the queue head once the dTD is retired (see section Software Link Pointers).

In addition to the current and next pointers and the dTD overlay examined in section Operational Model For Packet Transfers, the dQH also contains the following parameters for the associated endpoint: Multipler, Maximum Packet Length, Interrupt On Setup. The complete initialization of the dQH including these fields is demonstrated in the next section.

42.5.6.5.1 Queue Head Initialization

One device queue head must be initialized for each active endpoint.

To initialize a device queue head:

- Write the wMaxPacketSize field as required by the USB Chapter 9 or application specific protocol.
- Write the multiplier field to 0 for control, bulk, and interrupt endpoints. For ISO endpoints, set the multiplier to 1,2, or 3 as required bandwidth and in conjunction with the USB Chapter 9 protocol.

NOTE

In FS mode, the multiplier field can only be 1 for ISO endpoints.

- Write the next dTD Terminate bit field to 1.
- Write the Active bit in the status field to 0.
- Write the Halt bit in the status field to 0.

NOTE

The DCD must only modify dQH if the associated endpoint is not primed and there are no outstanding dTD's.

42.5.6.5.2 Operational Model For Setup Transfers

As discussed in section Control Endpoint Operation Model, setup transfer requires special treatment by the DCD. A setup transfer does not use a dTD but instead stores the incoming data from a setup packet in an 8-byte buffer within the dQH.

Upon receiving notification of the setup packet, the DCD should handle the setup transfer as demonstrated here:

1. Copy setup buffer contents from dQH - RX to software buffer.
2. Acknowledge setup backup by writing a "1" to the corresponding bit in ENDPTSETUPSTAT.

NOTE

- The acknowledge must occur before continuing to process the setup packet.
- After the acknowledge has occurred, the DCD must not attempt to access the setup buffer in the dQH - RX. Only the local software copy should be examined.

3. Check for pending data or status dTD's from previous control transfers and flush if any exist as discussed in section Flushing/De-priming an Endpoint.
4. Decode setup packet and prepare data phase [optional] and status phase transfer as required by the USB Chapter 9 or application specific protocol.

NOTE

It is possible for the device controller to receive setup packets before previous control transfers complete. Existing control packets in progress must be flushed and the new control packet completed.

42.5.6.6 Managing Transfers with Transfer Descriptors

42.5.6.6.1 Software Link Pointers

It is necessary for the DCD software to maintain head and tail pointers to the for the linked list of dTDs for each respective queue head.

This is necessary because the dQH only maintains pointers to the current working dTD and the next dTD to be executed. The operations described in next section for managing dTD will assume the DCD can use reference the head and tail of the dTD linked list. The following figure shows the Software Link Pointers.

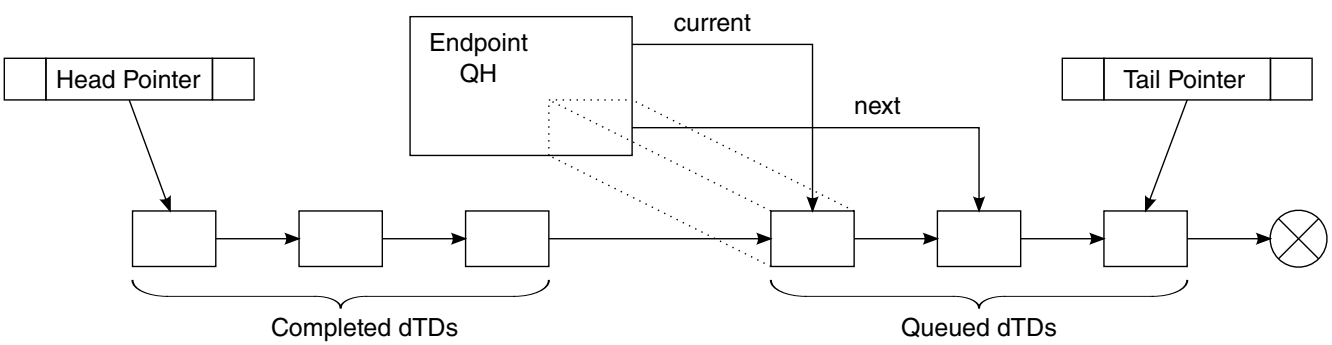


Figure 42-30. Software Link Pointers

NOTE

To conserve memory, the reserved fields at the end of the dQH can be used to store the Head & Tail pointers, but it still remains the responsibility of the DCD to maintain the pointers.

42.5.6.6.2 Building a Transfer Descriptor

Before a transfer can be executed from the linked list, a dTD must be built to describe the transfer.

Use the following procedure for building dTDs.

Allocate 8-DWord dTD block of memory aligned to 8-DWord boundaries. Example: bit address 4:0 would be equal to "00000"

Write the following fields:

1. Initialize first 7 DWords to 0.
2. Set the terminate bit to 1.
3. Fill in total bytes with transfer size.
4. Set the interrupt on complete if desired.
5. Initialize the status field with the active bit set to 1 and all remaining status bits set to 0.
6. Fill in buffer pointer page 0 and the current offset to point to the start of the data buffer.
7. Initialize buffer pointer page 1 through page 4 to be one greater than each of the previous buffer pointer.

42.5.6.6.3 Executing A Transfer Descriptor

To safely add a dTD, the DCD must be follow this procedure which will handle the event where the device controller reaches the end of the dTD list at the same time a new dTD is being added to the end of the list.

Determine whether the link list is empty: Check DCD driver to see if pipe is empty (internal representation of linked-list should indicate if any packets are outstanding).

- Case 1: Link list is empty
  a. Write dQH next pointer AND dQH terminate bit to 0 as a single DWord operation.
  b. Clear active & halt bit in dQH (in case set from a previous error).
  c. Prime endpoint by writing 1 to correct bit position in Endpoint Prime (USB_*n*ENDPTPRIME).
- Case 2: Link list is not empty
  a. Add dTD to end of linked list.
  b. Read correct prime bit in Endpoint Prime (USB_*n*ENDPTPRIME)- if 1 DONE.
  c. Set ATDTW bit in USBCMD register to 1.
  d. Read correct status bit in Endpoint Status (USB_*n*ENDPTSTAT). (store in tmp. variable for later)
  e. Read ATDTW bit in USBCMD register.
     - If 0 goto 3.
     - If 1 continue to 6.
  f. Write ATDTW bit in USBCMD register to 0.
  g. If status bit read in (3) is 1 DONE.

h. If status bit read in (3) is 0 then Goto Case 1: Step 1.

42.5.6.6.4 Transfer Completion

After a dTD has been initialized and the associated endpoint primed the device controller will execute the transfer upon the host-initiated request. The DCD will be notified with a USB interrupt if the Interrupt On Complete bit was set or alternately, the DCD can poll the endpoint complete register to find when the dTD had been executed. After a dTD has been executed, DCD can check the status bits to determine success or failure.

NOTE

Multiple dTD can be completed in a single endpoint complete notification. After clearing the notification, DCD must search the dTD linked list and retire all dTDs that have finished (Active bit cleared).

By reading the status fields of the completed dTDs, the DCD can determine if the transfers completed successfully. Success is determined with the following combination of status bits:

- Active = 0
- Halted = 0
- Transaction Error = 0
- Data Buffer Error = 0

Should any combination other than the one shown above exist, the DCD must take proper action. Transfer failure mechanisms are indicated in the Device Error Matrix.

In addition to checking the status bit the DCD must read the Transfer Bytes field to determine the actual bytes transferred. When a transfer is complete, the Total Bytes transferred is by decremented by the actual bytes transferred. For Transmit packets, a packet is only complete after the actual bytes reaches zero, but for receive packets, the host may send fewer bytes in the transfer according the USB variable length packet protocol.

42.5.6.6.5 Flushing/De-priming an Endpoint

It is necessary for the DCD to flush to de-prime one more endpoints on a USB device reset or during a broken control transfer.

There may also be application specific requirements to stop transfers in progress. The following procedure can be used by the DCD to stop a transfer in progress:

1. Write a '1' to the corresponding bit(s) in Endpoint Flush (USB_*n*ENDPTFLUSH).
2. Wait until all bits in Endpoint Flush (USB_*n*ENDPTFLUSH) are '0'.

- Software note: this operation may take a large amount of time depending on the USB bus activity. It is not desirable to have this wait loop within an interrupt service routine.

3. Read Endpoint Status (USB_*n*ENDPTSTAT) to ensure that for all endpoints commanded to be flushed, that the corresponding bits are now '0'. If the corresponding bits are '1' after step #2 has finished, then the flush failed as described in the following:
   - Explanation: In very rare cases, a packet is in progress to the particular endpoint when commanded flush using Endpoint Flush (USB_*n*ENDPTFLUSH). A safeguard is in place to refuse the flush to ensure that the packet in progress completes successfully. The DCD may need to repeatedly flush any endpoints that fail to flush be repeating steps 1-3 until each endpoint is successfully flushed.

42.5.6.6.6 Device Error Matrix

The following table summarizes packet errors that are not automatically handled by the Device Controller.

Table 42-73. Device Error Matrix

Error	Direction	Packet Type	Data Buffer Error Bit	Transaction Error Bit
Overflow **	RX	Any	1	0
ISO Packet Error	RX	ISO	0	1
ISO Fulfillment Error	Both	ISO	0	1

Notice that the device controller handles all errors on Bulk/Control/Interrupt Endpoints except for a data buffer overflow. However, for ISO endpoints, errors packets are not retried and errors are tagged as indicated. The table below describes the errors.

Table 42-74. Error Descriptions

Error	Description
Overflow	Number of bytes received exceeded max. packet size or total buffer length. ** This error will also set the Halt bit in the dQH and if there are dTDs remaining in the linked list for the endpoint, then those will not be executed.
ISO Packet Error	CRC Error on received ISO packet. Contents not guaranteed to be correct.
ISO Fulfillment Error	Host failed to complete the number of packets defined in the dQH mult field within the given (micro)frame. For scheduled data delivery the DCD may need to readjust the data queue because a fulfillment error will cause Device Controller to cease data transfers on the pipe for one (micro)frame. During the "dead" (micro)frame, the Device Controller reports error on the pipe and primes for the following frame.

42.5.6.7 Servicing Interrupts

The interrupt service routine must consider that there are high-frequency, low-frequency operations, and error operations and order accordingly.

42.5.6.7.1 High-Frequency Interrupts

High frequency interrupts in particular should be handled in the order below. The most important of these is listed first because the DCD must acknowledge a setup buffer in the timeliest manner possible.

The table below describes the High frequency interrupt events.

Table 42-75. High Frequency Interrupt Events

Execution Order	Interrupt	Action
1a	USB Interrupt - USB.ENDPTSETUPSTATUS	Copy contents of setup buffer and acknowledge setup packet (as indicated in Figure 42-29 shows the End Point Queue Head). Process setup packet according to USB 2.0 Chapter 9 or application specific protocol.
1b	USB Interrupt[1] - USB.ENDPTCOMPLETE	Handle completion of dTD as indicated in Figure 42-29 shows the End Point Queue Head.
2	SOF Interrupt	Action as deemed necessary by application. This interrupt may not have a use in all applications.

1. It is likely that multiple interrupts to stack up on any call to the Interrupt Service Routine AND during the Interrupt Service Routine.

42.5.6.7.2 Low-Frequency Interrupts

The low frequency interrupts can be handled in any order because they do not occur often in comparison to the high-frequency interrupts.

The table below shows the Low frequency interrupt events.

Table 42-76. Low Frequency Interrupt Events

Interrupt	Action
Port Change	Change software state information.
Sleep Enable (Suspend)	Change software state information. Low power handling as necessary.
Reset Received	Change software state information. Abort pending transfers.

42.5.6.7.3 Error Interrupts

Error interrupts will be least frequent and should be placed last in the interrupt service routine.

The following table shows the error interrput events.

Table 42-77. Error Interrupt Events

Interrupt	Action
USB Error Interrupt	This error is redundant because it combines USB Interrupt and an error status in the dTD. The DCD will more aptly handle packet-level errors by checking dTD status field upon receipt of USB Interrupt (w/ USB.ENDPTCOMPLETE).
System Error	Unrecoverable error. Immediate Reset of core; free transfers buffers in progress and restart the DCD.

42.6 USB Non-Core Memory Map/Register Definition

There are two kinds of registers in the USB module: USB core registers and USB non-core registers. USB core registers are used to control USB core functions, and more independent of USB features. Each USB controller core has its own core registers. USB non-core registers are additional to USB core registers, and more dependent on USB features. i.MX series products vary in non-core registers.

This section describes only the USB non-core registers. For detailed descriptions of USB core registers, please refer to Register Interface.

NOTE

- For reserved bits, please preserve the value when writing (read its reset value, then write this value back)
- "USB_UOG1_", "USB_UOG2_" prefix in register name indicates it is a core register for OTG1/OTG2 controller core respectively.
- USBNC_USB_" prefix in register name indicates it is a USB non-core register.

USBNC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
402E_0800	USB OTG1 Control Register (USBNC_USB_OTG1_CTRL)	32	R/W	3000_1000h	42.6.1/2426

Table continues on the next page...

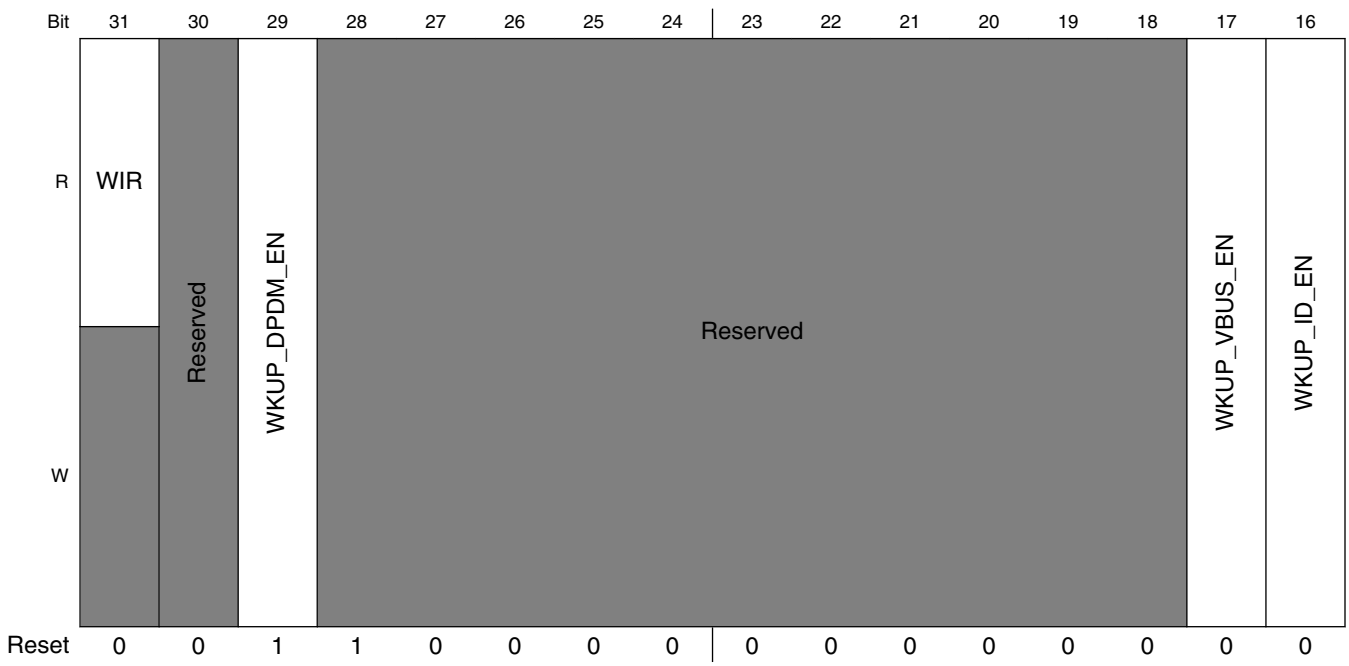
USBNC memory map (continued)

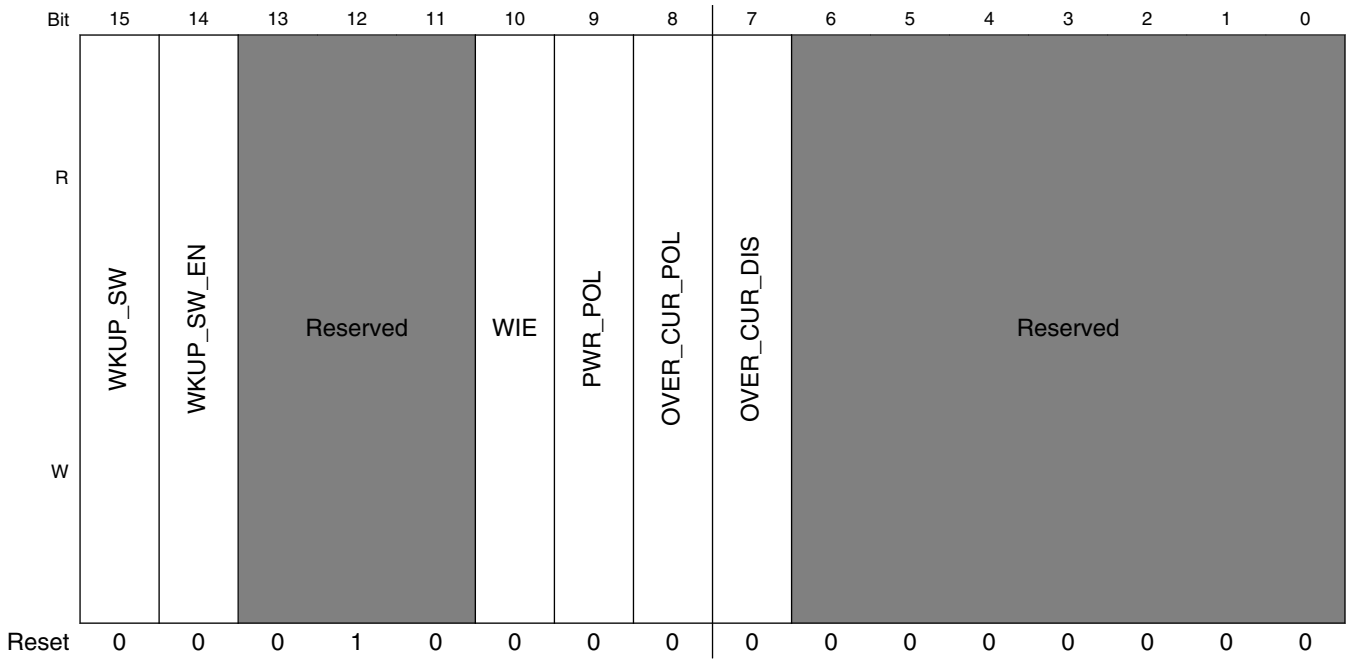
Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
402E_0804	USB OTG2 Control Register (USBNC_USB_OTG2_CTRL)	32	R/W	3000_1000h	42.6.2/2429
402E_0818	OTG1 UTMI PHY Control 0 Register (USBNC_USB_OTG1_PHY_CTRL_0)	32	R/W	0000_0000h	42.6.3/2432
402E_081C	OTG2 UTMI PHY Control 0 Register (USBNC_USB_OTG2_PHY_CTRL_0)	32	R/W	0000_0098h	42.6.4/2433

42.6.1 USB OTG1 Control Register (USBNC_USB_OTG1_CTRL)

The USB OTG1 control register controls the integration specific features of the USB OTG1 module. These features are not directly related to the USB functionality, but control special features, interfacing on the USB ports, as well as power control and wake-up functionality.

Address: 402E_0000h base + 800h offset = 402E_0800h





USBNC_USB_OTG1_CTRL field descriptions

Field	Description
31 WIR	OTG1 Wake-up Interrupt Request This bit indicates that a wake-up interrupt request is received on the OTG1 port. This bit is cleared by disabling the wake-up interrupt (clearing bit "OWIE"). 1 Wake-up Interrupt Request received 0 No wake-up interrupt request received
30 -	This field is reserved. Reserved
29 WKUP_DPDM_EN	Wake-up on DPDM change enable 1 (Default) DPDM changes wake-up to be enabled, it is for device only. 0 DPDM changes wake-up to be disabled only when VBUS is 0.
28–18 -	This field is reserved. Reserved
17 WKUP_VBUS_EN	OTG1 wake-up on VBUS change enable 1 Enable 0 Disable
16 WKUP_ID_EN	OTG1 Wake-up on ID change enable 1 Enable 0 Disable
15 WKUP_SW	OTG1 Software Wake-up 1 Force wake-up 0 Inactive

Table continues on the next page...

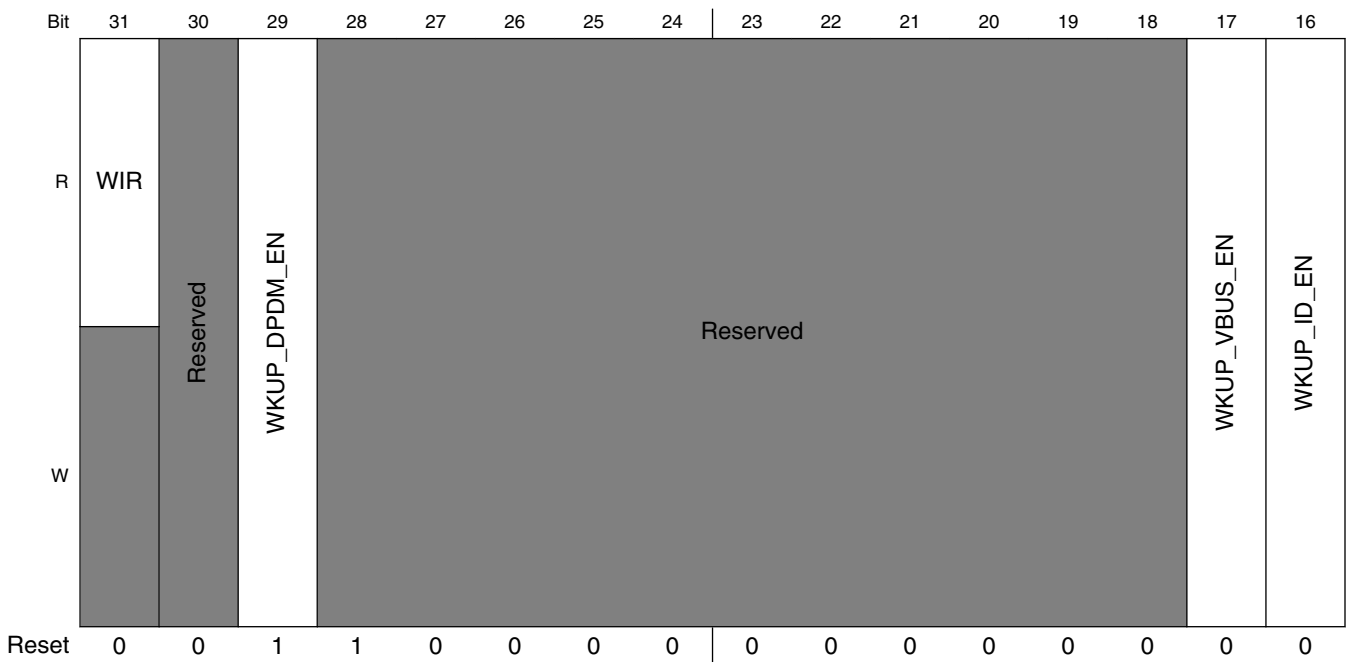
USBNC_USB_OTG1_CTRL field descriptions (continued)

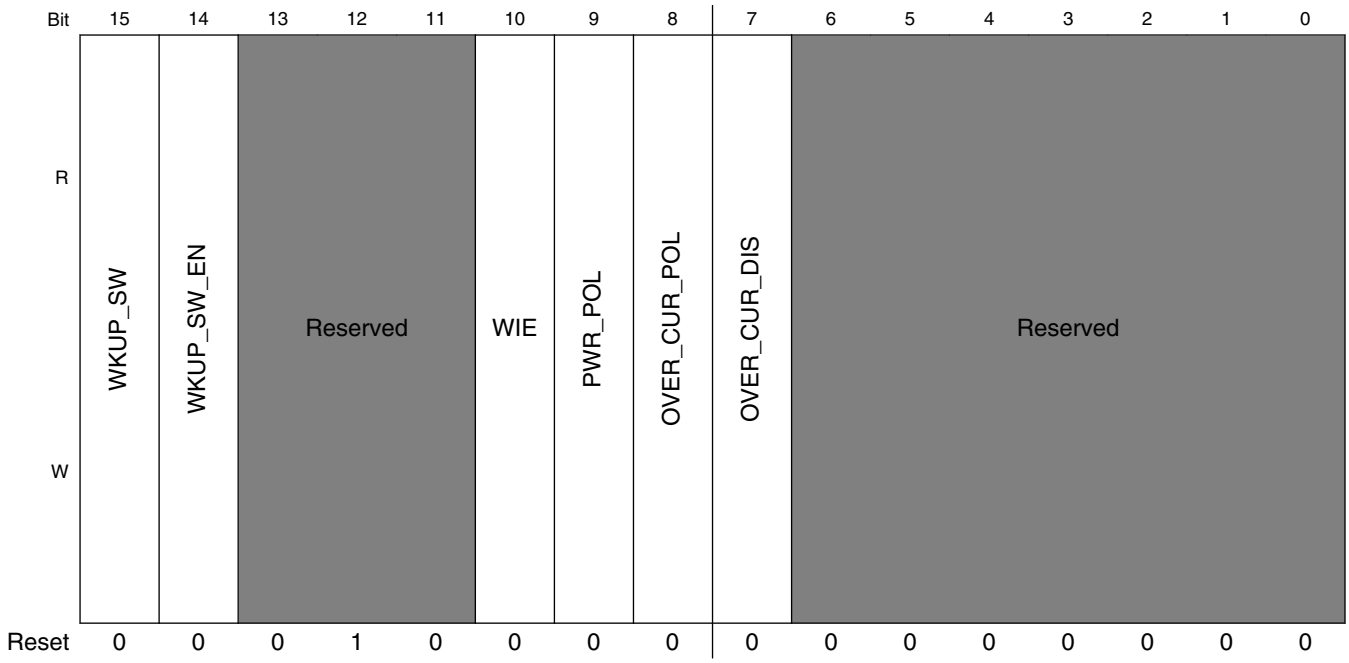
Field	Description
14 WKUP_SW_EN	OTG1 Software Wake-up Enable 1 Enable 0 Disable
13–11 -	This field is reserved. Reserved
10 WIE	OTG1 Wake-up Interrupt Enable This bit enables or disables the OTG1 wake-up interrupt. Disabling the interrupt also clears the Interrupt request bit. Wake-up interrupt enable should be turned off after receiving a wake-up interrupt and turned on again prior to going in suspend mode 1 Interrupt Enabled 0 Interrupt Disabled
9 PWR_POL	OTG1 Power Polarity This bit should be set according to PMIC Power Pin polarity. 1 PMIC Power Pin is High active. 0 PMIC Power Pin is Low active.
8 OVER_CUR_ POL	OTG1 Polarity of Overcurrent The polarity of OTG1 port overcurrent event 1 Low active (low on this signal represents an overcurrent condition) 0 High active (high on this signal represents an overcurrent condition)
7 OVER_CUR_DIS	Disable OTG1 Overcurrent Detection 1 Disables overcurrent detection 0 Enables overcurrent detection
-	This field is reserved. Reserved

42.6.2 USB OTG2 Control Register (USBNC_USB_OTG2_CTRL)

The USB OTG2 control register controls the integration specific features of the USB OTG2 module. These features are not directly related to the USB functionality, but control special features, interfacing on the USB ports, as well as power control and wake-up functionality.

Address: 402E_0000h base + 804h offset = 402E_0804h





USBNC_USB_OTG2_CTRL field descriptions

Field	Description
31 WIR	OTG2 Wake-up Interrupt Request This bit indicates that a wake-up interrupt request is received on the OTG port. This bit is cleared by disabling the wake-up interrupt (clearing bit "OWIE"). 1 Wake-up Interrupt Request received 0 No wake-up interrupt request received
30 -	This field is reserved. Reserved
29 WKUP_DPDM_EN	Wake-up on DPDM change enable 1 (Default) DPDM changes wake-up to be enabled, it is for device only. 0 DPDM changes wake-up to be disabled only when VBUS is 0.
28–18 -	This field is reserved. Reserved
17 WKUP_VBUS_EN	OTG2 wake-up on VBUS change enable 1 Enable 0 Disable
16 WKUP_ID_EN	OTG2 Wake-up on ID change enable 1 Enable 0 Disable
15 WKUP_SW	OTG2 Software Wake-up 1 Force wake-up 0 Inactive

Table continues on the next page...

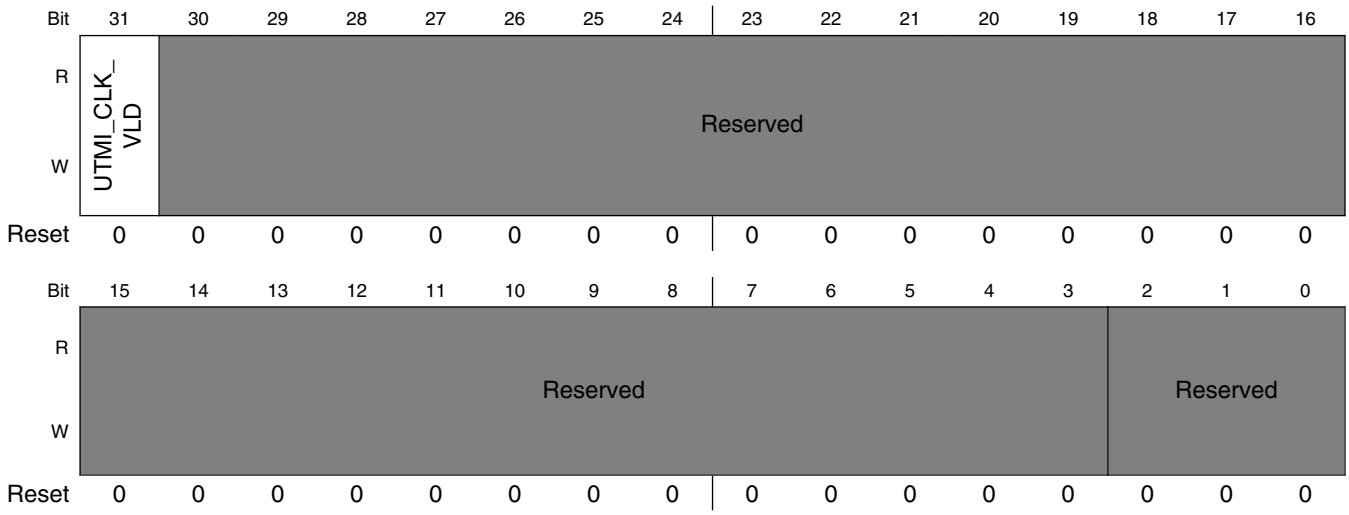
USBNC_USB_OTG2_CTRL field descriptions (continued)

Field	Description
14 WKUP_SW_EN	OTG2 Software Wake-up Enable 1 Enable 0 Disable
13–11 -	This field is reserved. Reserved
10 WIE	OTG2 Wake-up Interrupt Enable This bit enables or disables the OTG2 wake-up interrupt. Disabling the interrupt also clears the Interrupt request bit. Wake-up interrupt enable should be turned off after receiving a wake-up interrupt and turned on again prior to going in suspend mode 1 Interrupt Enabled 0 Interrupt Disabled
9 PWR_POL	OTG2 Power Polarity This bit should be set according to PMIC Power Pin polarity. 1 PMIC Power Pin is High active. 0 PMIC Power Pin is Low active.
8 OVER_CUR_POL	OTG2 Polarity of Overcurrent The polarity of OTG2 port overcurrent event 1 Low active (low on this signal represents an overcurrent condition) 0 High active (high on this signal represents an overcurrent condition)
7 OVER_CUR_DIS	Disable OTG2 Overcurrent Detection 1 Disables overcurrent detection 0 Enables overcurrent detection
-	This field is reserved. Reserved

42.6.3 OTG1 UTMI PHY Control 0 Register (USBNC_USB_OTG1_PHY_CTRL_0)

USB OTG1 UTMI PHY control register 0 is used to control the on-chip OTG1 UTMI PHY.

Address: 402E_0000h base + 818h offset = 402E_0818h



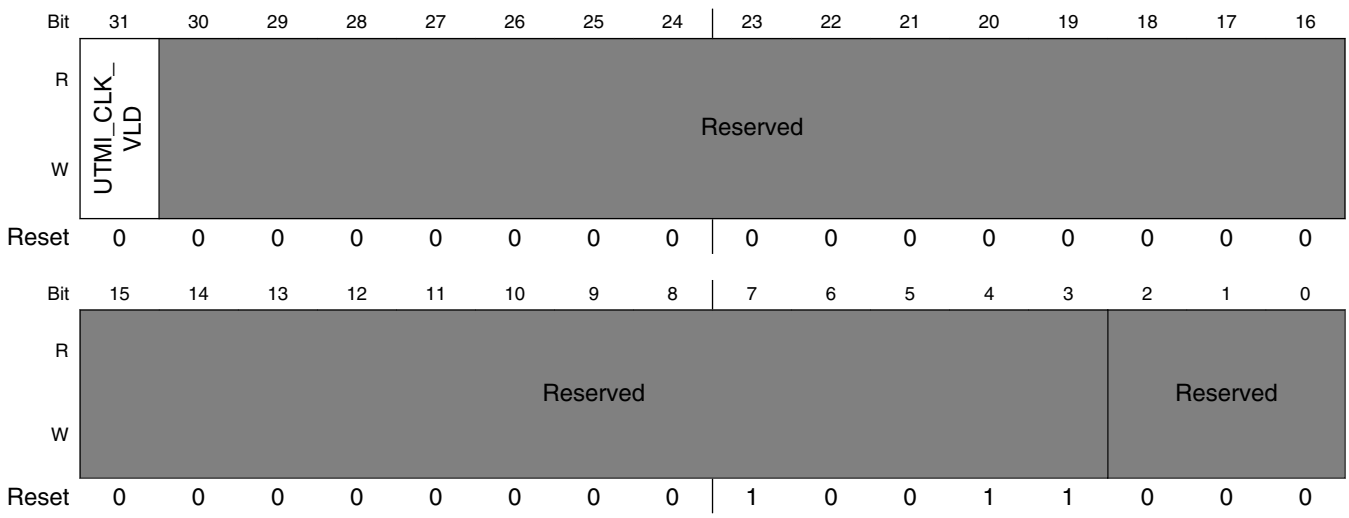
USBNC_USB_OTG1_PHY_CTRL_0 field descriptions

Field	Description
31 UTMI_CLK_VLD	Indicating whether OTG1 UTMI PHY clock is valid 1 Valid 0 Invalid
30–3 -	This field is reserved. Reserved
-	This field is reserved. Reserved

42.6.4 OTG2 UTMI PHY Control 0 Register (USBNC_USB_OTG2_PHY_CTRL_0)

USB OTG2 UTMI PHY Control Register 0 are used to control the on-chip OTG2 UTMI PHY.

Address: 402E_0000h base + 81Ch offset = 402E_081Ch



USBNC_USB_OTG2_PHY_CTRL_0 field descriptions

Field	Description
31 UTMI_CLK_VLD	Indicating whether OTG2 UTMI PHY clock is valid 1 Valid 0 Invalid
30–3 -	This field is reserved. Reserved
-	This field is reserved. Reserved

42.7 USB Core Memory Map/Register Definition

USB memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
402E_0000	Identification register (USB_UOG1_ID)	32	R	E4A1_FA05h	42.7.1/2438

Table continues on the next page...

USB memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
402E_0004	Hardware General (USB_UOG1_HWGENERAL)	32	R	0000_0035h	42.7.2/2438
402E_0008	Host Hardware Parameters (USB_UOG1_HWHOST)	32	R	1002_0001h	42.7.3/2440
402E_000C	Device Hardware Parameters (USB_UOG1_HWDEVICE)	32	R	0000_0011h	42.7.4/2440
402E_0010	TX Buffer Hardware Parameters (USB_UOG1_HWTXBUF)	32	R	8008_0B08h	42.7.5/2441
402E_0014	RX Buffer Hardware Parameters (USB_UOG1_HWRXBUF)	32	R	0000_0808h	42.7.6/2442
402E_0080	General Purpose Timer #0 Load (USB_UOG1_GPTIMER0LD)	32	R/W	0000_0000h	42.7.7/2442
402E_0084	General Purpose Timer #0 Controller (USB_UOG1_GPTIMER0CTRL)	32	R/W	0000_0000h	42.7.8/2443
402E_0088	General Purpose Timer #1 Load (USB_UOG1_GPTIMER1LD)	32	R/W	0000_0000h	42.7.9/2444
402E_008C	General Purpose Timer #1 Controller (USB_UOG1_GPTIMER1CTRL)	32	R/W	0000_0000h	42.7.10/ 2445
402E_0090	System Bus Config (USB_UOG1_SBUSCFG)	32	R/W	0000_0002h	42.7.11/ 2446
402E_0100	Capability Registers Length (USB_UOG1_CAPLENGTH)	8	R	40h	42.7.12/ 2447
402E_0102	Host Controller Interface Version (USB_UOG1_HCIVERSION)	16	R	0100h	42.7.13/ 2447
402E_0104	Host Controller Structural Parameters (USB_UOG1_HCSPARAMS)	32	R	0001_0011h	42.7.14/ 2448
402E_0108	Host Controller Capability Parameters (USB_UOG1_HCCPARAMS)	32	R	0000_0006h	42.7.15/ 2450
402E_0120	Device Controller Interface Version (USB_UOG1_DCIVERSION)	16	R	0001h	42.7.16/ 2452
402E_0124	Device Controller Capability Parameters (USB_UOG1_DCCPARAMS)	32	R	0000_0188h	42.7.17/ 2453
402E_0140	USB Command Register (USB_UOG1_USBCMD)	32	R/W	0008_0000h	42.7.18/ 2454
402E_0144	USB Status Register (USB_UOG1_USBSTS)	32	R/W	0000_0000h	42.7.19/ 2458
402E_0148	Interrupt Enable Register (USB_UOG1_USBINTR)	32	R/W	0000_0000h	42.7.20/ 2462
402E_014C	USB Frame Index (USB_UOG1_FRINDEX)	32	R/W	0000_0000h	42.7.21/ 2464
402E_0154	Frame List Base Address (USB_UOG1_PERIODICLISTBASE)	32	R/W	0000_0000h	42.7.22/ 2465
402E_0154	Device Address (USB_UOG1_DEVICEADDR)	32	R/W	0000_0000h	42.7.23/ 2465
402E_0158	Next Asynch. Address (USB_UOG1_ASYNCLISTADDR)	32	R/W	0000_0000h	42.7.24/ 2466
402E_0158	Endpoint List Address (USB_UOG1_ENDPTLISTADDR)	32	R/W	0000_0000h	42.7.25/ 2467

Table continues on the next page...

USB memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
402E_0160	Programmable Burst Size (USB_UOG1_BURSTSIZE)	32	R/W	0000_0808h	42.7.26/ 2467
402E_0164	TX FIFO Fill Tuning (USB_UOG1_TXFILLTUNING)	32	R/W	0000_0000h	42.7.27/ 2468
402E_0178	Endpoint NAK (USB_UOG1_ENDPTNAK)	32	R/W	0000_0000h	42.7.28/ 2470
402E_017C	Endpoint NAK Enable (USB_UOG1_ENDPTNAKEN)	32	R/W	0000_0000h	42.7.29/ 2470
402E_0180	Configure Flag Register (USB_UOG1_CONFIGFLAG)	32	R	0000_0001h	42.7.30/ 2471
402E_0184	Port Status & Control (USB_UOG1_PORTSC1)	32	R/W	1000_0000h	42.7.31/ 2471
402E_01A4	On-The-Go Status & control (USB_UOG1_OTGSC)	32	R/W	0000_1120h	42.7.32/ 2478
402E_01A8	USB Device Mode (USB_UOG1_USBMODE)	32	R/W	0000_5000h	42.7.33/ 2482
402E_01AC	Endpoint Setup Status (USB_UOG1_ENDPTSETUPSTAT)	32	R/W	0000_0000h	42.7.34/ 2483
402E_01B0	Endpoint Prime (USB_UOG1_ENDPTPRIME)	32	R/W	0000_0000h	42.7.35/ 2484
402E_01B4	Endpoint Flush (USB_UOG1_ENDPTFLUSH)	32	R/W	0000_0000h	42.7.36/ 2485
402E_01B8	Endpoint Status (USB_UOG1_ENDPTSTAT)	32	R	0000_0000h	42.7.37/ 2485
402E_01BC	Endpoint Complete (USB_UOG1_ENDPTCOMPLETE)	32	R/W	0000_0000h	42.7.38/ 2486
402E_01C0	Endpoint Control0 (USB_UOG1_ENDPTCTRL0)	32	R/W	0080_0080h	42.7.39/ 2487
402E_01C4	Endpoint Control 1 (USB_UOG1_ENDPTCTRL1)	32	R/W	0000_0000h	42.7.40/ 2489
402E_01C8	Endpoint Control 2 (USB_UOG1_ENDPTCTRL2)	32	R/W	0000_0000h	42.7.41/ 2492
402E_01CC	Endpoint Control 3 (USB_UOG1_ENDPTCTRL3)	32	R/W	0000_0000h	42.7.42/ 2494
402E_01D0	Endpoint Control 4 (USB_UOG1_ENDPTCTRL4)	32	R/W	0000_0000h	42.7.43/ 2497
402E_01D4	Endpoint Control 5 (USB_UOG1_ENDPTCTRL5)	32	R/W	0000_0000h	42.7.44/ 2500
402E_01D8	Endpoint Control 6 (USB_UOG1_ENDPTCTRL6)	32	R/W	0000_0000h	42.7.45/ 2503
402E_01DC	Endpoint Control 7 (USB_UOG1_ENDPTCTRL7)	32	R/W	0000_0000h	42.7.46/ 2506
402E_0200	Identification register (USB_UOG2_ID)	32	R	E4A1_FA05h	42.7.1/2438
402E_0204	Hardware General (USB_UOG2_HWGENERAL)	32	R	0000_0035h	42.7.2/2438

Table continues on the next page...

USB memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
402E_0208	Host Hardware Parameters (USB_UOG2_HWHOST)	32	R	1002_0001h	42.7.3/2440
402E_020C	Device Hardware Parameters (USB_UOG2_HWDEVICE)	32	R	0000_0011h	42.7.4/2440
402E_0210	TX Buffer Hardware Parameters (USB_UOG2_HWTXBUF)	32	R	8008_0B08h	42.7.5/2441
402E_0214	RX Buffer Hardware Parameters (USB_UOG2_HWRXBUF)	32	R	0000_0808h	42.7.6/2442
402E_0280	General Purpose Timer #0 Load (USB_UOG2_GPTIMER0LD)	32	R/W	0000_0000h	42.7.7/2442
402E_0284	General Purpose Timer #0 Controller (USB_UOG2_GPTIMER0CTRL)	32	R/W	0000_0000h	42.7.8/2443
402E_0288	General Purpose Timer #1 Load (USB_UOG2_GPTIMER1LD)	32	R/W	0000_0000h	42.7.9/2444
402E_028C	General Purpose Timer #1 Controller (USB_UOG2_GPTIMER1CTRL)	32	R/W	0000_0000h	42.7.10/ 2445
402E_0290	System Bus Config (USB_UOG2_SBUSCFG)	32	R/W	0000_0002h	42.7.11/ 2446
402E_0300	Capability Registers Length (USB_UOG2_CAPLENGTH)	8	R	40h	42.7.12/ 2447
402E_0302	Host Controller Interface Version (USB_UOG2_HCIVERSION)	16	R	0100h	42.7.13/ 2447
402E_0304	Host Controller Structural Parameters (USB_UOG2_HCSPARAMS)	32	R	0001_0011h	42.7.14/ 2448
402E_0308	Host Controller Capability Parameters (USB_UOG2_HCCPARAMS)	32	R	0000_0006h	42.7.15/ 2450
402E_0320	Device Controller Interface Version (USB_UOG2_DCIVERSION)	16	R	0001h	42.7.16/ 2452
402E_0324	Device Controller Capability Parameters (USB_UOG2_DCCPARAMS)	32	R	0000_0188h	42.7.17/ 2453
402E_0340	USB Command Register (USB_UOG2_USBCMD)	32	R/W	0008_0000h	42.7.18/ 2454
402E_0344	USB Status Register (USB_UOG2_USBSTS)	32	R/W	0000_0000h	42.7.19/ 2458
402E_0348	Interrupt Enable Register (USB_UOG2_USBINTR)	32	R/W	0000_0000h	42.7.20/ 2462
402E_034C	USB Frame Index (USB_UOG2_FRINDEX)	32	R/W	0000_0000h	42.7.21/ 2464
402E_0354	Frame List Base Address (USB_UOG2_PERIODICLISTBASE)	32	R/W	0000_0000h	42.7.22/ 2465
402E_0354	Device Address (USB_UOG2_DEVICEADDR)	32	R/W	0000_0000h	42.7.23/ 2465
402E_0358	Next Asynch. Address (USB_UOG2_ASYNCLISTADDR)	32	R/W	0000_0000h	42.7.24/ 2466
402E_0358	Endpoint List Address (USB_UOG2_ENDPTLISTADDR)	32	R/W	0000_0000h	42.7.25/ 2467
402E_0360	Programmable Burst Size (USB_UOG2_BURSTSIZE)	32	R/W	0000_0808h	42.7.26/ 2467

Table continues on the next page...

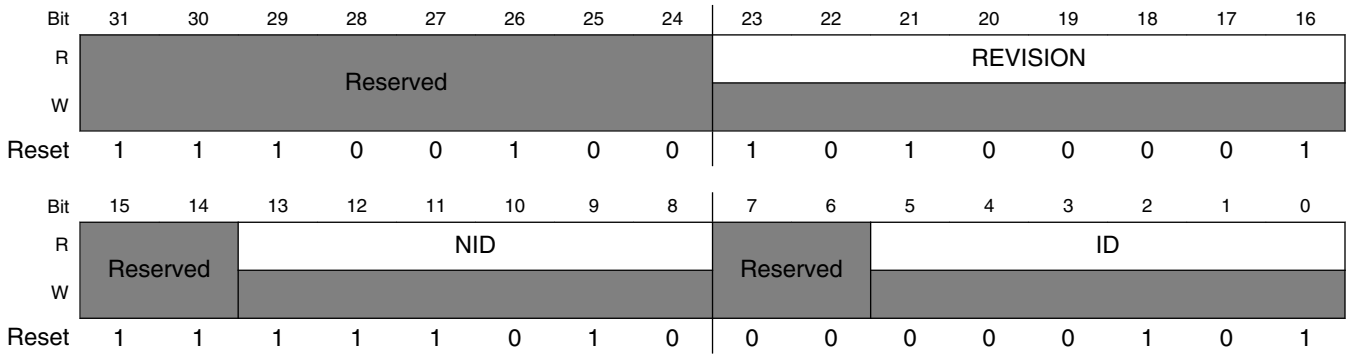
USB memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
402E_0364	TX FIFO Fill Tuning (USB_UOG2_TXFILLTUNING)	32	R/W	0000_0000h	42.7.27/ 2468
402E_0378	Endpoint NAK (USB_UOG2_ENDPTNAK)	32	R/W	0000_0000h	42.7.28/ 2470
402E_037C	Endpoint NAK Enable (USB_UOG2_ENDPTNAKEN)	32	R/W	0000_0000h	42.7.29/ 2470
402E_0380	Configure Flag Register (USB_UOG2_CONFIGFLAG)	32	R	0000_0001h	42.7.30/ 2471
402E_0384	Port Status & Control (USB_UOG2_PORTSC1)	32	R/W	1000_0000h	42.7.31/ 2471
402E_03A4	On-The-Go Status & control (USB_UOG2_OTGSC)	32	R/W	0000_1120h	42.7.32/ 2478
402E_03A8	USB Device Mode (USB_UOG2_USBMODE)	32	R/W	0000_5000h	42.7.33/ 2482
402E_03AC	Endpoint Setup Status (USB_UOG2_ENDPTSETUPSTAT)	32	R/W	0000_0000h	42.7.34/ 2483
402E_03B0	Endpoint Prime (USB_UOG2_ENDPTPRIME)	32	R/W	0000_0000h	42.7.35/ 2484
402E_03B4	Endpoint Flush (USB_UOG2_ENDPTFLUSH)	32	R/W	0000_0000h	42.7.36/ 2485
402E_03B8	Endpoint Status (USB_UOG2_ENDPTSTAT)	32	R	0000_0000h	42.7.37/ 2485
402E_03BC	Endpoint Complete (USB_UOG2_ENDPTCOMPLETE)	32	R/W	0000_0000h	42.7.38/ 2486
402E_03C0	Endpoint Control0 (USB_UOG2_ENDPTCTRL0)	32	R/W	0080_0080h	42.7.39/ 2487
402E_03C4	Endpoint Control 1 (USB_UOG2_ENDPTCTRL1)	32	R/W	0000_0000h	42.7.40/ 2489
402E_03C8	Endpoint Control 2 (USB_UOG2_ENDPTCTRL2)	32	R/W	0000_0000h	42.7.41/ 2492
402E_03CC	Endpoint Control 3 (USB_UOG2_ENDPTCTRL3)	32	R/W	0000_0000h	42.7.42/ 2494
402E_03D0	Endpoint Control 4 (USB_UOG2_ENDPTCTRL4)	32	R/W	0000_0000h	42.7.43/ 2497
402E_03D4	Endpoint Control 5 (USB_UOG2_ENDPTCTRL5)	32	R/W	0000_0000h	42.7.44/ 2500
402E_03D8	Endpoint Control 6 (USB_UOG2_ENDPTCTRL6)	32	R/W	0000_0000h	42.7.45/ 2503
402E_03DC	Endpoint Control 7 (USB_UOG2_ENDPTCTRL7)	32	R/W	0000_0000h	42.7.46/ 2506

42.7.1 Identification register (USB_*n*ID)

The ID register identifies the USB 2.0 High-Speed core and its revision.

Address: 402E_0000h base + 0h offset + (512d × i), where i=0d to 1d



USB_*n*ID field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23–16 REVISION	Revision number of the controller core.
15–14 -	This field is reserved. Reserved
13–8 NID	Complement version of ID
7–6 -	This field is reserved. Reserved
ID	Configuration number. This number is set to 0x05 and indicates that the peripheral is USB 2.0 High-Speed core.

42.7.2 Hardware General (USB_*n*HWGENERAL)

General hardware parameters as defined in System Level Issues and Core Configuration.

NOTE

The reset value could vary from instance to instance. Please see the detail in bit field description and ignore reset value in summary table in this case!

Address: 402E_0000h base + 4h offset + (512d × i), where i=0d to 1d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

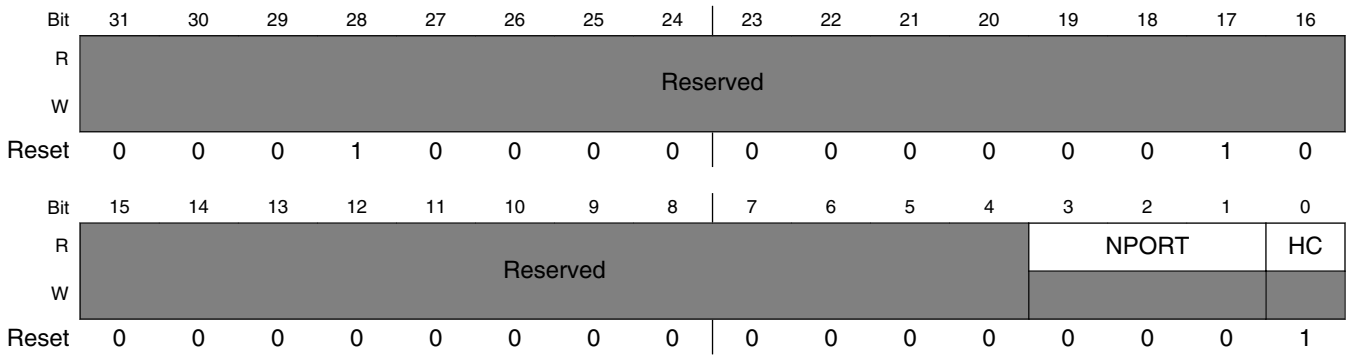
Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved					SM		PHYM			PHYW		Reserved			
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	1

USB_*n*HWGENERAL field descriptions

Field	Description
31–11 -	This field is reserved. Reserved
10–9 SM	Serial interface mode capability 00 No Serial Engine, always use parallel signalling. 01 Serial Engine present, always use serial signalling for FS/LS. 10 Software programmable - Reset to use parallel signalling for FS/LS 11 Software programmable - Reset to use serial signalling for FS/LS
8–6 PHYM	Transciever type 000 UTMI/UMTI+ 001 ULPI DDR 010 ULPI 011 Serial Only 100 Software programmable - reset to UTMI/UTMI+ 101 Software programmable - reset to ULPI DDR 110 Software programmable - reset to ULPI 111 Software programmable - reset to Serial 1000 IC-USB 1001 Software programmable - reset to IC-USB
5–4 PHYW	Data width of the transciever connected to the controller core. PHYW bit reset value is 00 8 bit wide data bus Software non-programmable 01 16 bit wide data bus Software non-programmable 10 Reset to 8 bit wide data bus Software programmable 11 Reset to 16 bit wide data bus Software programmable
-	This field is reserved. Reserved

42.7.3 Host Hardware Parameters (USB_*n*HWHOST)

Address: 402E_0000h base + 8h offset + (512d × i), where i=0d to 1d

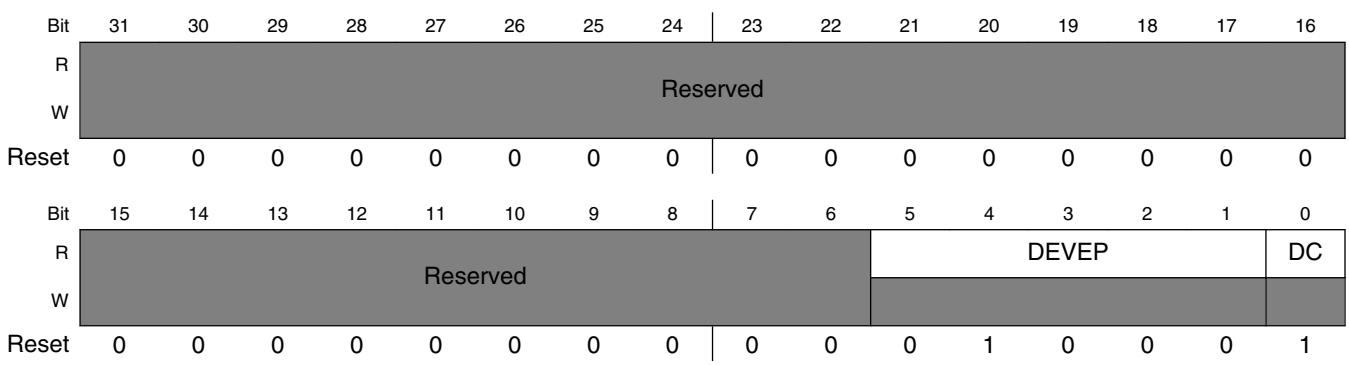


USB_*n*HWHOST field descriptions

Field	Description
31–4 -	This field is reserved. Reserved
3–1 NPORT	The Nmber of downstream ports supported by the host controller is (NPORT+1). **NOTE:** When these bits value is '000', it indicates a single-port host controller.
0 HC	Host Capable. Indicating whether host operation mode is supported or not. 1 Supported 0 Not supported

42.7.4 Device Hardware Parameters (USB_*n*HWDEVICE)

Address: 402E_0000h base + Ch offset + (512d × i), where i=0d to 1d



USB_*n*HWDEVICE field descriptions

Field	Description
31–6 -	This field is reserved. Reserved
5–1 DEVEP	Device Endpoint Number
0 DC	Device Capable. Indicating whether device operation mode is supported or not. 1 Supported 0 Not supported

42.7.5 TX Buffer Hardware Parameters (USB_*n*HWTXBUF)

Address: 402E_0000h base + 10h offset + (512d × i), where i=0d to 1d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								TXCHANADD								Reserved								TXBURST							
W																																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	1	0	0	0

USB_*n*HWTXBUF field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23–16 TXCHANADD	TX FIFO Buffer size is: (2^TXCHANADD) * 4 Bytes. These bits are set to '08h', so buffer size is 256*4 Bytes. For the OTG controller operating in device mode, this is the FIFO buffer size per endpoint. As the OTG controller has 8 TX endpoint, there are 8 of these buffers. For the OTG controller operating in host mode, or for Host-only controller, the entire buffer memory is used as a single TX buffer. Therefore, there is only 1 of this buffer
15–8 -	This field is reserved. Reserved
TXBURST	Default burst size for memory to TX buffer transfer. This is reset value of TXPBURST bits in USB core regsiter USB_n_BURSTSIZE. Please see Programmable Burst Size (USB_*n*BURSTSIZE) .

42.7.6 RX Buffer Hardware Parameters (USB_*n*HWRXBUF)

Address: 402E_0000h base + 14h offset + (512d × i), where i=0d to 1d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved																RXADD								RXBURST							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0

USB_*n*HWRXBUF field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
15–8 RXADD	Buffer total size for all receive endpoints is (2^RXADD). RX Buffer size is: (2^RXADD) * 4 Bytes. These bits are set to '08h', so buffer size is 256*4 Bytes. There is a single Receive FIFO buffer in the USB controller. The buffer is shared for all endpoints for the OTG controller in device mode.
RXBURST	Default burst size for memory to RX buffer transfer. This is reset value of RXPBURST bits in USB core regsiter USB_n_BURSTSIZE. Please see Programmable Burst Size (USB_*n*BURSTSIZE) .

42.7.7 General Purpose Timer #0 Load (USB_*n*GPTIMER0LD)

This register controls load value of the count timer in register n_GPTIMER0CTRL. Please see General Purpose Timer #0 Controller (USB_*n*GPTIMER0CTRL) .

Address: 402E_0000h base + 80h offset + (512d × i), where i=0d to 1d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								GPTLD																							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USB_*n*GPTIMER0LD field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
GPTLD	General Purpose Timer Load Value These bit fields are loaded to GPTCNT bits when GPTRST bit is set '1b'. This value represents the time in microseconds minus 1 for the timer duration.

Table continues on the next page...

USB_*n*GPTIMER0LD field descriptions (continued)

Field	Description
	Example: for a one millisecond timer, load 1000-1=999 or 0x0003E7. **NOTE:** Max value is 0xFFFFFF or 16.777215 seconds.

42.7.8 General Purpose Timer #0 Controller (USB_*n*GPTIMER0CTRL)

This register contains the control for this countdown timer and a data field can be queried to determine the running count value. This timer has granularity on 1 us and can be programmed to a little over 16 seconds. There are two counter modes which are described in the register table below. When the timer counter value transitions to zero, an interrupt could be generated if enable.

Interrupt status bit is TI0 bit in n_USBSTS register (See USB Status Register (USB_*n*USBSTS)), interrupt enable bit is TIE0 bit in n_USBINTR register. (See Interrupt Enable Register (USB_*n*USBINTR) .)

Address: 402E_0000h base + 84h offset + (512d × i), where i=0d to 1d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	GPTRUN	GPTRST	Reserved					GPTMODE	GPTCNT							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GPTCNT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USB_*n*GPTIMER0CTRL field descriptions

Field	Description
31 GPTRUN	General Purpose Timer Run GPTCNT bits are not effected when setting or clearing this bit. 0 Stop counting 1 Run
30 GPTRST	General Purpose Timer Reset

Table continues on the next page...

USB_*n*GPTIMER0CTRL field descriptions (continued)

Field	Description
	0 No action 1 Load counter value from GPTLD bits in n_GPTIMER0LD
29–25 -	This field is reserved. Reserved
24 GPTMODE	General Purpose Timer Mode In one shot mode, the timer will count down to zero, generate an interrupt, and stop until the counter is reset by software; In repeat mode, the timer will count down to zero, generate an interrupt and automatically reload the counter value from GPTLD bits to start again. 0 One Shot Mode 1 Repeat Mode
GPTCNT	General Purpose Timer Counter. This field is the count value of the countdown timer.

42.7.9 General Purpose Timer #1 Load (USB_*n*GPTIMER1LD)

This register controls load value of the count timer in register n_GPTIMER1CTRL. Please see General Purpose Timer #1 Controller (USB_*n*GPTIMER1CTRL) .

Address: 402E_0000h base + 88h offset + (512d × i), where i=0d to 1d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								GPTLD																							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USB_*n*GPTIMER1LD field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
GPTLD	General Purpose Timer Load Value These bit fields are loaded to GPTCNT bits when GPTRST bit is set '1b'. This value represents the time in microseconds minus 1 for the timer duration. Example: for a one millisecond timer, load 1000-1=999 or 0x0003E7. **NOTE:** Max value is 0xFFFFFF or 16.777215 seconds.

42.7.10 General Purpose Timer #1 Controller (USB_*n*GPTIMER1CTRL)

This register contains the control for this countdown timer and a data field can be queried to determine the running count value. This timer has granularity on 1 us and can be programmed to a little over 16 seconds. There are two counter modes which are described in the register table below. When the timer counter value transitions to zero, an interrupt could be generated if enable.

Interrupt status bit is TI1 bit in USB_n_USBSTS register (See USB Status Register (USB_*n*USBSTS)), interrupt enable bit is TIE1 bit in n_USBINTR register (See Interrupt Enable Register (USB_*n*USBINTR)).

Address: 402E_0000h base + 8Ch offset + (512d × i), where i=0d to 1d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	GPTRUN	GPTRST	Reserved					GPTMODE	GPTCNT							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	GPTCNT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USB_*n*GPTIMER1CTRL field descriptions

Field	Description
31 GPTRUN	General Purpose Timer Run GPTCNT bits are not effected when setting or clearing this bit. 0 Stop counting 1 Run
30 GPTRST	General Purpose Timer Reset 0 No action 1 Load counter value from GPTLD bits in USB_n_GPTIMER0LD
29–25 -	This field is reserved. Reserved
24 GPTMODE	General Purpose Timer Mode In one shot mode, the timer will count down to zero, generate an interrupt, and stop until the counter is reset by software. In repeat mode, the timer will count down to zero, generate an interrupt and automatically reload the counter value from GPTLD bits to start again.

Table continues on the next page...

USB_*n*GPTIMER1CTRL field descriptions (continued)

Field	Description
	0 One Shot Mode 1 Repeat Mode
GPTCNT	General Purpose Timer Counter. This field is the count value of the countdown timer.

42.7.11 System Bus Config (USB_*n*SBUSCFG)

Address: 402E_0000h base + 90h offset + (512d × i), where i=0d to 1d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																													AHBBRST		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

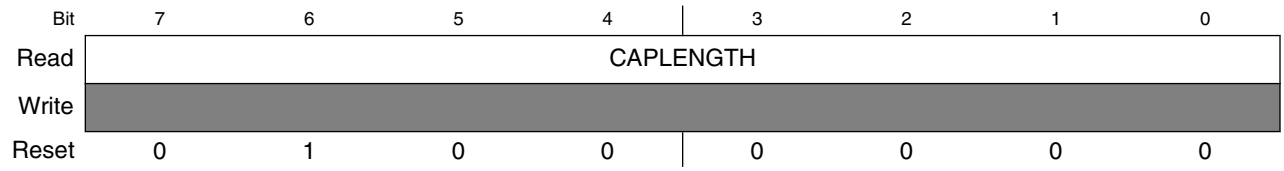
USB_*n*SBUSCFG field descriptions

Field	Description
31–3 -	This field is reserved. Reserved
AHBBRST	AHB master interface Burst configuration These bits control AHB master transfer type sequence (or priority). **NOTE:** This register overrides n_BURSTSIZE register when its value is not zero. 000 Incremental burst of unspecified length only 001 INCR4 burst, then single transfer 010 INCR8 burst, INCR4 burst, then single transfer 011 INCR16 burst, INCR8 burst, INCR4 burst, then single transfer 100 Reserved, don't use 101 INCR4 burst, then incremental burst of unspecified length 110 INCR8 burst, INCR4 burst, then incremental burst of unspecified length 111 INCR16 burst, INCR8 burst, INCR4 burst, then incremental burst of unspecified length

42.7.12 Capability Registers Length (USB_*n*CAPLENGTH)

The Capability Registers Length register contains the address offset to the Operational registers relative to the CAPLENGTH register.

Address: 402E_0000h base + 100h offset + (512d × i), where i=0d to 1d



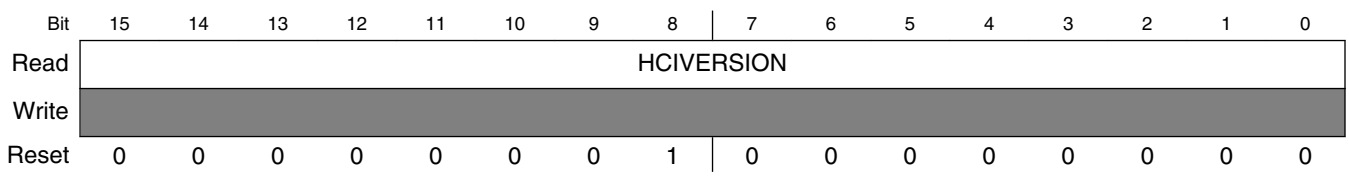
USB_*n*CAPLENGTH field descriptions

Field	Description
CAPLENGTH	These bits are used as an offset to add to register base to find the beginning of the Operational Register. Default value is '40h'.

42.7.13 Host Controller Interface Version (USB_*n*HCIVERSION)

This is a 2-byte register containing a BCD encoding of the EHCI revision number supported by this host controller. The most significant byte of this register represents a major revision and the least significant byte is the minor revision.

Address: 402E_0000h base + 102h offset + (512d × i), where i=0d to 1d



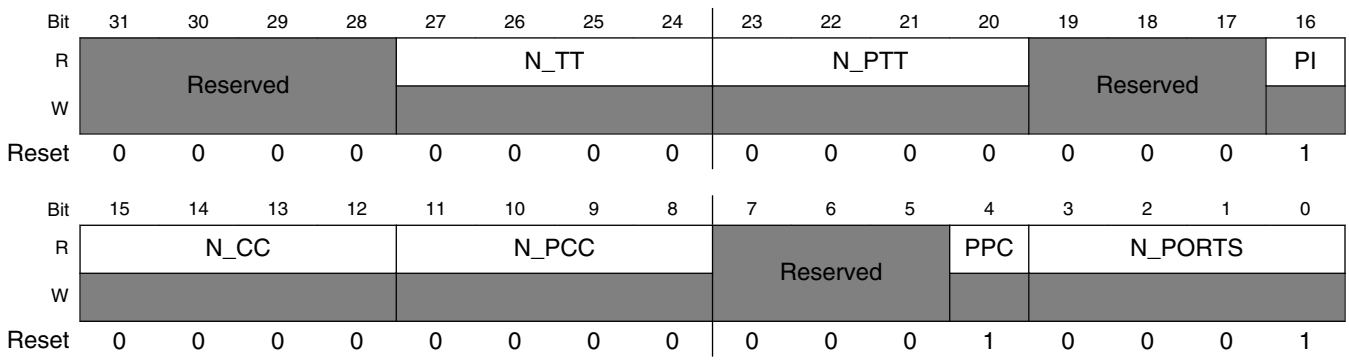
USB_*n*HCIVERSION field descriptions

Field	Description
HCIVERSION	Host Controller Interface Version Number Default value is '10h', which means EHCI rev1.0.

42.7.14 Host Controller Structural Parameters (USB_*n*HCSPARAMS)

The following figure shows the port steering logic capabilities of Host Control Structural Parameters (n_HCSPARAMS).

Address: 402E_0000h base + 104h offset + (512d × i), where i=0d to 1d



USB_*n*HCSPARAMS field descriptions

Field	Description
31–28 -	This field is reserved. Reserved
27–24 N_TT	Number of Transaction Translators (N_TT). Default value '0000b' This field indicates the number of embedded transaction translators associated with the USB2.0 host controller. These bits would be set to '0001b' for Multi-Port Host, and '0000b' for Single-Port Host.
23–20 N_PTT	Number of Ports per Transaction Translator (N_PTT). Default value '0000b' This field indicates the number of ports assigned to each transaction translator within the USB2.0 host controller. These bits would be set to equal N_PORTS for Multi-Port Host, and '0000b' for Single-Port Host.
19–17 -	This field is reserved. Reserved
16 PI	Port Indicators (P INDICATOR) This bit indicates whether the ports support port indicator control. When set to one, the port status and control registers include a read/writeable field for controlling the state of the port indicator This bit is "1b" in all controller core.
15–12 N_CC	Number of Companion Controller (N_CC). This field indicates the number of companion controllers associated with this USB2.0 host controller. These bits are '0000b' in all controller core. 0 There is no internal Companion Controller and port-ownership hand-off is not supported. 1 There are internal companion controller(s) and port-ownership hand-offs is supported.

Table continues on the next page...

USB_*n*HCSPARAMS field descriptions (continued)

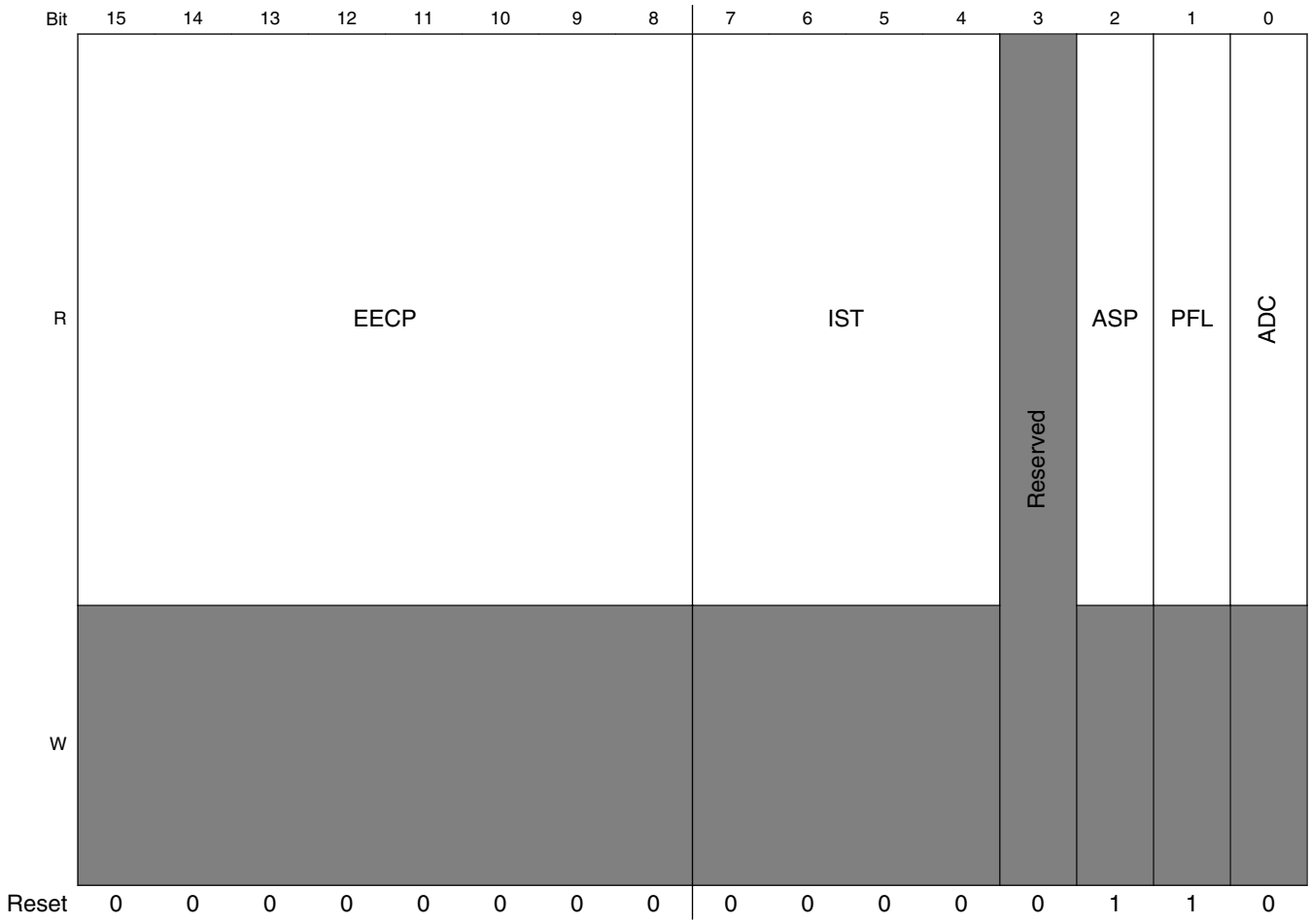
Field	Description
11–8 N_PCC	Number of Ports per Companion Controller This field indicates the number of ports supported per internal Companion Controller. It is used to indicate the port routing configuration to the system software. For example, if N_PORTS has a value of 6 and N_CC has a value of 2 then N_PCC could have a value of 3. The convention is that the first N_PCC ports are assumed to be routed to companion controller 1, the next N_PCC ports to companion controller 2, etc. In the previous example, the N_PCC could have been 4, where the first 4 are routed to companion controller 1 and the last two are routed to companion controller 2. The number in this field must be consistent with N_PORTS and N_CC. These bits are '0000b' in all controller core.
7–5 -	This field is reserved. Reserved
4 PPC	Port Power Control This field indicates whether the host controller implementation includes port power control. A one indicates the ports have port power switches. A zero indicates the ports do not have port power switches. The value of this field affects the functionality of the Port Power field in each port status and control register
N_PORTS	Number of downstream ports. This field specifies the number of physical downstream ports implemented on this host controller. The value of this field determines how many port registers are addressable in the Operational Register. Valid values are in the range of 1h to Fh. A zero in this field is undefined. These bits are always set to '0001b' because all controller cores are Single-Port Host.

42.7.15 Host Controller Capability Parameters (USB_*n*HCCPARAMS)

This register identifies multiple mode control (time-base bit functionality), addressing capability.

Address: 402E_0000h base + 108h offset + (512d × i), where i=0d to 1d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



USB_*n*HCCPARAMS field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
15–8 EECP	EHCI Extended Capabilities Pointer. This field indicates the existence of a capabilities list. A value of 00h indicates no extended capabilities are implemented. A non-zero value in this register indicates the offset in PCI configuration space of the first EHCI extended capability. The pointer value must be 40h or greater if implemented to maintain the consistency of the PCI header defined for this class of device. **NOTE:** These bits are set '00h' in all controller core.
7–4 IST	Isochronous Scheduling Threshold. This field indicates, relative to the current position of the executing host controller, where software can reliably update the isochronous schedule. When bit [7] is zero, the value of the least significant 3 bits indicates the number of micro-frames a host controller can hold a set of isochronous data structures (one or more) before flushing the state. When bit [7] is a one, then host software assumes the host controller may cache an isochronous data structure for an entire frame. These bits are set '00h' in all controller core.
3 -	This field is reserved. Reserved

Table continues on the next page...

USB_*n*HCCPARAMS field descriptions (continued)

Field	Description
2 ASP	Asynchronous Schedule Park Capability If this bit is set to a one, then the host controller supports the park feature for high-speed queue heads in the Asynchronous Schedule. The feature can be disabled or enabled and set to a specific level by using the *Asynchronous Schedule Park Mode Enable* and *Asynchronous Schedule Park Mode Count* fields in the USBCMD register. **NOTE:** ASP bit reset value: '00b' for OTG controller core, '11b' for Host-only controller core.
1 PFL	Programmable Frame List Flag If this bit is set to zero, then the system software must use a frame list length of 1024 elements with this host controller. The USBCMD register Frame List Size field is a read-only register and must be set to zero. If set to a one, then the system software can specify and use a smaller frame list and configure the host controller via the USBCMD register Frame List Size field. The frame list must always be aligned on a 4K-page boundary. This requirement ensures that the frame list is always physically contiguous. This bit is set '1b' in all controller core.
0 ADC	64-bit Addressing Capability This bit is set '0b' in all controller core, no 64-bit addressing capability is supported.

42.7.16 Device Controller Interface Version (USB_*n*DCIVERSION)

This register indicates the two-byte BCD encoding of the device controller interface version number.

Address: 402E_0000h base + 120h offset + (512d × i), where i=0d to 1d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	DCIVERSION															
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

USB_*n*DCIVERSION field descriptions

Field	Description
DCIVERSION	Device Controller Interface Version Number Default value is '01h', which means rev0.1.

42.7.17 Device Controller Capability Parameters (USB_*n*DCCPARAMS)

These fields describe the overall device capability of the controller.

Address: 402E_0000h base + 124h offset + (512d × i), where i=0d to 1d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved							HC	DC	Reserved		DEN				
W																
Reset	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0

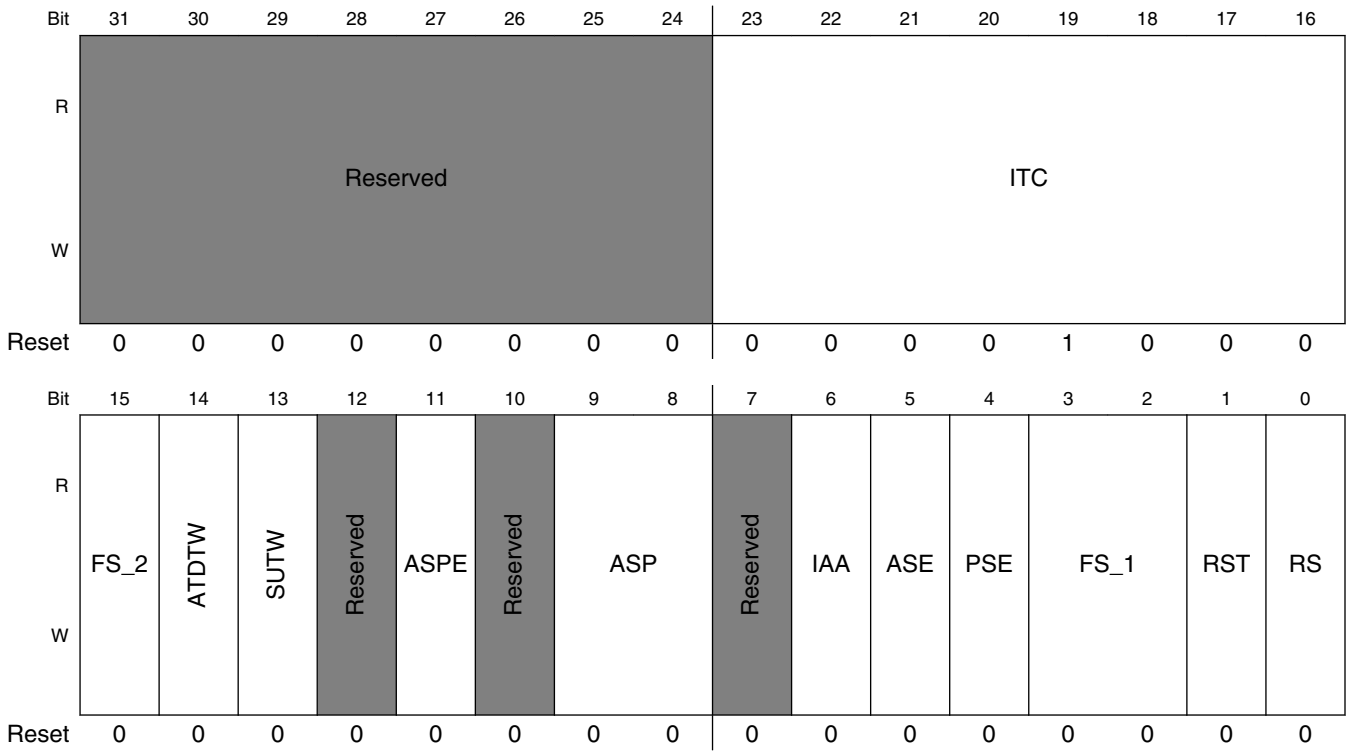
USB_*n*DCCPARAMS field descriptions

Field	Description
31–9 -	This field is reserved. Reserved
8 HC	Host Capable When this bit is 1, this controller is capable of operating as an EHCI compatible USB 2.0 host controller.
7 DC	Device Capable When this bit is 1, this controller is capable of operating as a USB 2.0 device.
6–5 -	This field is reserved. Reserved
DEN	Device Endpoint Number This field indicates the number of endpoints built into the device controller. If this controller is not device capable, then this field will be zero. Valid values are 0 - 15.

42.7.18 USB Command Register (USB_*n*USBCMD)

The Command Register indicates the command to be executed by the serial bus host/device controller. Writing to the register causes a command to be executed.

Address: 402E_0000h base + 140h offset + (512d × i), where i=0d to 1d



USB_*n*USBCMD field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23–16 ITC	Interrupt Threshold Control -Read/Write. The system software uses this field to set the maximum rate at which the host/device controller will issue interrupts. ITC contains the maximum interrupt interval measured in micro-frames. Valid values are shown below. Value Maximum Interrupt Interval 0x00 Immediate (no threshold) 0x01 1 micro-frame 0x02 2 micro-frames 0x04 4 micro-frames 0x08 8 micro-frames 0x10 16 micro-frames

Table continues on the next page...

USB_*n*USBCMD field descriptions (continued)

Field	Description
	0x20 32 micro-frames 0x40 64 micro-frames
15 FS_2	Frame List Size - (Read/Write or Read Only). [host mode only] This field is Read/Write only if Programmable Frame List Flag in the HCCPARAMS registers is set to one. This field specifies the size of the frame list that controls which bits in the Frame Index Register should be used for the Frame List Current index. **NOTE:** This field is made up from USBCMD bits 15, 3 and 2. Value Meaning 000 1024 elements (4096 bytes) Default value 001 512 elements (2048 bytes) 010 256 elements (1024 bytes) 011 128 elements (512 bytes) 100 64 elements (256 bytes) 101 32 elements (128 bytes) 110 16 elements (64 bytes) 111 8 elements (32 bytes)
14 ATDTW	Add dTD TripWire - Read/Write. [device mode only] This bit is used as a semaphore to ensure proper addition of a new dTD to an active (primed) endpoint's linked list. This bit is set and cleared by software. This bit would also be cleared by hardware when state machine is hazard region for which adding a dTD to a primed endpoint may go unrecognized.
13 SUTW	Setup TripWire - Read/Write. [device mode only] This bit is used as a semaphore to ensure that the setup data payload of 8 bytes is extracted from a QH by the DCD without being corrupted. If the setup lockout mode is off (SLOM bit in USB core register n_USBMODE, see USB Device Mode (USB_*n*USBMODE)) then there is a hazard when new setup data arrives while the DCD is copying the setup data payload from the QH for a previous setup packet. This bit is set and cleared by software. This bit would also be cleared by hardware when a hazard detected.
12 -	This field is reserved. Reserved
11 ASPE	Asynchronous Schedule Park Mode Enable - Read/Write. If the *Asynchronous Park Capability* bit in the HCCPARAMS register is a one, then this bit defaults to a 1h and is R/W. Otherwise the bit must be a zero and is RO. Software uses this bit to enable or disable Park mode. When this bit is one, Park mode is enabled. When this bit is a zero, Park mode is disabled. **NOTE:** ASPE bit reset value: '0b' for OTG controller .
10 -	This field is reserved. Reserved
9–8 ASP	Asynchronous Schedule Park Mode Count - Read/Write. If the *Asynchronous Park Capability* bit in the HCCPARAMS register is a one, then this field defaults to 3h and is R/W. Otherwise it defaults to zero and is Read-Only. It contains a count of the number of successive transactions the host controller is allowed to execute from a high-speed queue head on the

Table continues on the next page...

USB_*n*USBCMD field descriptions (continued)

Field	Description
	Asynchronous schedule before continuing traversal of the Asynchronous schedule. Valid values are 1h to 3h. Software must not write a zero to this bit when *Park Mode Enable* is a one as this will result in undefined behavior. This field is set to 3h in all controller core.
7 -	This field is reserved. Reserved
6 IAA	Interrupt on Async Advance Doorbell - Read/Write. This bit is used as a doorbell by software to tell the host controller to issue an interrupt the next time it advances asynchronous schedule. Software must write a 1 to this bit to ring the doorbell. When the host controller has evicted all appropriate cached schedule states, it sets the Interrupt on Async Advance status bit in the USBSTS register. If the Interrupt on Sync Advance Enable bit in the USBINTR register is one, then the host controller will assert an interrupt at the next interrupt threshold. The host controller sets this bit to zero after it has set the Interrupt on Sync Advance status bit in the USBSTS register to one. Software should not write a one to this bit when the asynchronous schedule is inactive. Doing so will yield undefined results. This bit is only used in host mode. Writing a one to this bit when device mode is selected will have undefined results.
5 ASE	Asynchronous Schedule Enable - Read/Write. Default 0b. This bit controls whether the host controller skips processing the Asynchronous Schedule. Only the host controller uses this bit. Values Meaning 0 Do not process the Asynchronous Schedule. 1 Use the ASYNCLISTADDR register to access the Asynchronous Schedule.
4 PSE	Periodic Schedule Enable- Read/Write. Default 0b. This bit controls whether the host controller skips processing the Periodic Schedule. Only the host controller uses this bit. Values Meaning 0 Do not process the Periodic Schedule 1 Use the PERIODICLISTBASE register to access the Periodic Schedule.
3–2 FS_1	See description at bit 15
1 RST	Controller Reset (RESET) - Read/Write. Software uses this bit to reset the controller. This bit is set to zero by the Host/Device Controller when the reset process is complete. Software cannot terminate the reset process early by writing a zero to this register. Host operation mode: When software writes a one to this bit, the Controller resets its internal pipelines, timers, counters, state machines etc. to their initial value. Any transaction currently in progress on USB is immediately terminated. A USB reset is not driven on downstream ports. Software should not set this bit to a one when the HCHalted bit in the USBSTS register is a zero. Attempting to reset an actively running host controller will result in undefined behavior. Device operation mode:

Table continues on the next page...

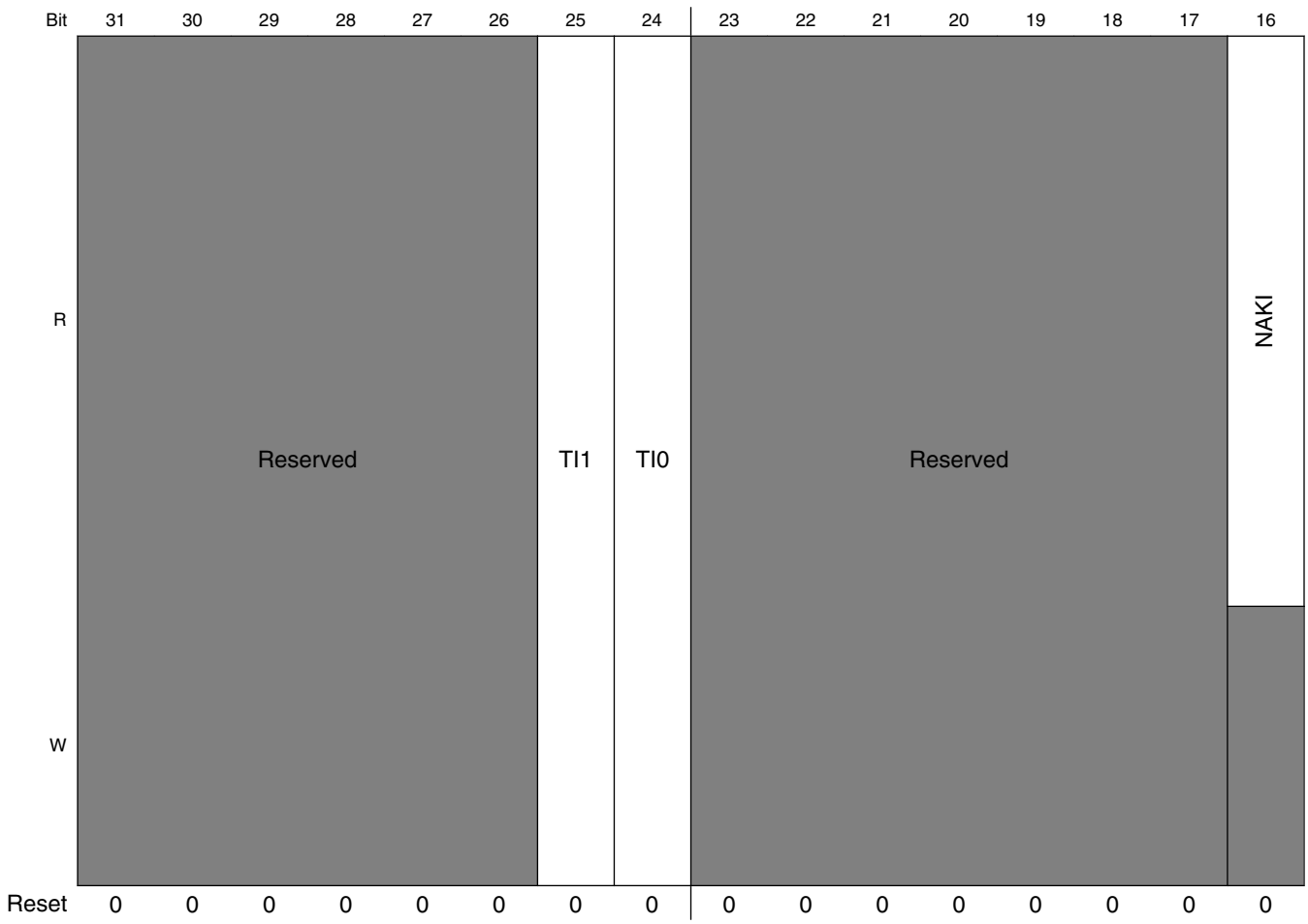
USB_*n*USBCMD field descriptions (continued)

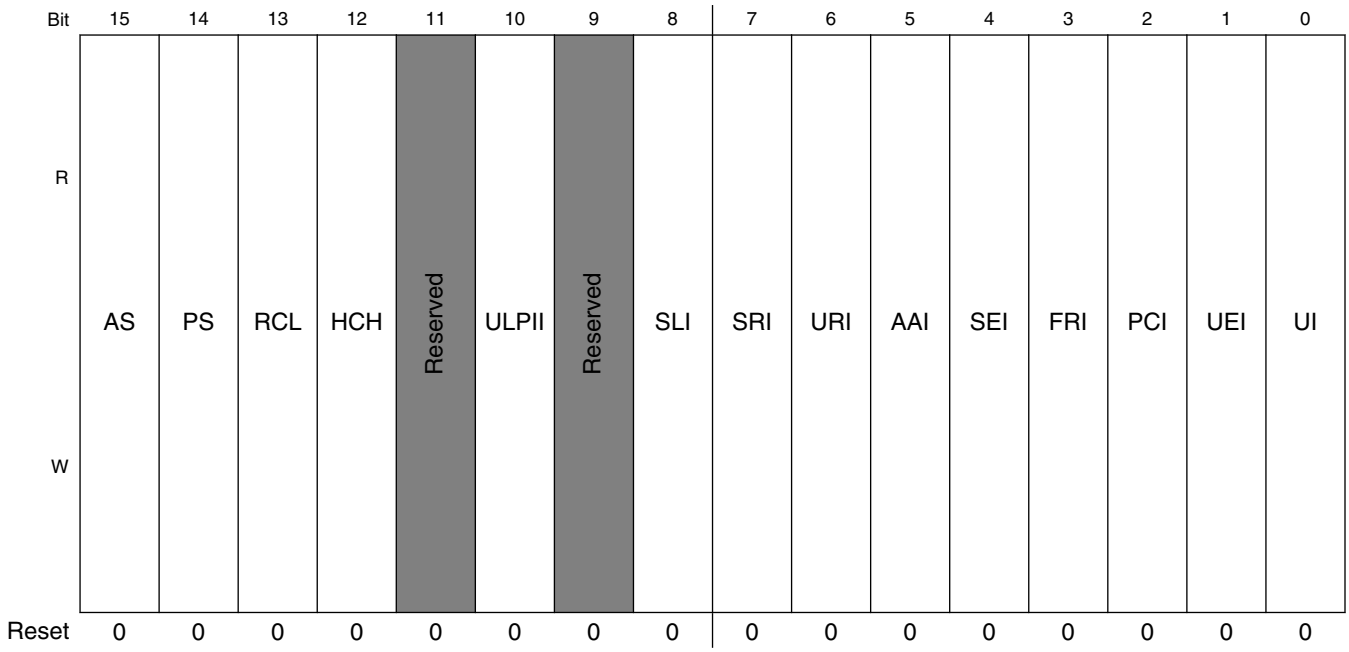
Field	Description
	When software writes a one to this bit, the Controller resets its internal pipelines, timers, counters, state machines etc. to their initial value. Writing a one to this bit when the device is in the attached state is not recommended, because the effect on an attached host is undefined. In order to ensure that the device is not in an attached state before initiating a device controller reset, all primed endpoints should be flushed and the USBCMD Run/Stop bit should be set to 0.
0 RS	Run/Stop (RS) - Read/Write. Default 0b. 1=Run. 0=Stop. Host operation mode: When set to '1b', the Controller proceeds with the execution of the schedule. The Controller continues execution as long as this bit is set to a one. When this bit is set to 0, the Host Controller completes the current transaction on the USB and then halts. The HC Halted bit in the status register indicates when the Controller has finished the transaction and has entered the stopped state. Software should not write a one to this field unless the controller is in the Halted state (that is, HCHalted in the USBSTS register is a one). Device operation mode: Writing a one to this bit will cause the controller to enable a pull-up on D+ and initiate an attach event. This control bit is not directly connected to the pull-up enable, as the pull-up will become disabled upon transitioning into high-speed mode. Software should use this bit to prevent an attach event before the controller has been properly initialized. Writing a 0 to this will cause a detach event.

42.7.19 USB Status Register (USB_*n*USBSTS)

This register indicates various states of the Host/Device Controller and any pending interrupts. This register does not indicate status resulting from a transaction on the serial bus.

Address: 402E_0000h base + 144h offset + (512d × i), where i=0d to 1d





USB_*n*USBSTS field descriptions

Field	Description
31–26 -	This field is reserved. Reserved
25 TI1	General Purpose Timer Interrupt 1(GPTINT1)--R/WC. This bit is set when the counter in the GPTIMER1CTRL register transitions to zero, writing a one to this bit will clear it.
24 TI0	General Purpose Timer Interrupt 0(GPTINT0)--R/WC. This bit is set when the counter in the GPTIMER0CTRL register transitions to zero, writing a one to this bit clears it.
23–17 -	This field is reserved. Reserved
16 NAKI	NAK Interrupt Bit--RO. This bit is set by hardware when for a particular endpoint both the TX/RX Endpoint NAK bit and corresponding TX/RX Endpoint NAK Enable bit are set. This bit is automatically cleared by hardware when all Enabled TX/RX Endpoint NAK bits are cleared.
15 AS	Asynchronous Schedule Status - Read Only. This bit reports the current real status of the Asynchronous Schedule. When set to zero the asynchronous schedule status is disabled and if set to one the status is enabled. The Host Controller is not required to immediately disable or enable the Asynchronous Schedule when software transitions the Asynchronous Schedule Enable bit in the USBCMD register. When this bit and the Asynchronous Schedule Enable bit are the same value, the Asynchronous Schedule is either enabled (1) or disabled (0). Only used in the host operation mode.
14 PS	Periodic Schedule Status - Read Only.

Table continues on the next page...

USB_*n*USBSTS field descriptions (continued)

Field	Description
	This bit reports the current real status of the Periodic Schedule. When set to zero the periodic schedule is disabled, and if set to one the status is enabled. The Host Controller is not required to immediately disable or enable the Periodic Schedule when software transitions the Periodic Schedule Enable bit in the USBCMD register. When this bit and the Periodic Schedule Enable bit are the same value, the Periodic Schedule is either enabled (1) or disabled (0). Only used in the host operation mode.
13 RCL	Reclamation - Read Only. This is a read-only status bit used to detect an empty asynchronous schedule. Only used in the host operation mode.
12 HCH	HCHalted - Read Only. This bit is a zero whenever the Run/Stop bit is a one. The Controller sets this bit to one after it has stopped executing because of the Run/Stop bit being set to 0, either by software or by the Controller hardware (for example, an internal error). Only used in the host operation mode. Default value is '0b' for OTG core. This is because OTG core is not operating as host in default. Please see CM bit in USB_n_USBMODE register. **NOTE:** HCH bit reset value: '0b' for OTG controller core.
11 -	This field is reserved. Reserved
10 ULPII	ULPI Interrupt - R/WC. This bit will be set '1b' by hardware when there is an event completion in ULPI viewport. This bit is usable only if the controller support UPLI interface mode.
9 -	This field is reserved. Reserved
8 SLI	DCSuspend - R/WC. When a controller enters a suspend state from an active state, this bit will be set to a one. The device controller clears the bit upon exiting from a suspend state. Only used in device operation mode.
7 SRI	SOF Received - R/WC. When the device controller detects a Start Of (micro) Frame, this bit will be set to a one. When a SOF is extremely late, the device controller will automatically set this bit to indicate that an SOF was expected. Therefore, this bit will be set roughly every 1ms in device FS mode and every 125ms in HS mode and will be synchronized to the actual SOF that is received. Because the device controller is initialized to FS before connect, this bit will be set at an interval of 1ms during the prelude to connect and chirp. In host mode, this bit will be set every 125us and can be used by host controller driver as a time base. Software writes a 1 to this bit to clear it.
6 URI	USB Reset Received - R/WC. When the device controller detects a USB Reset and enters the default state, this bit will be set to a one. Software can write a 1 to this bit to clear the USB Reset Received status bit. Only used in device operation mode.

Table continues on the next page...

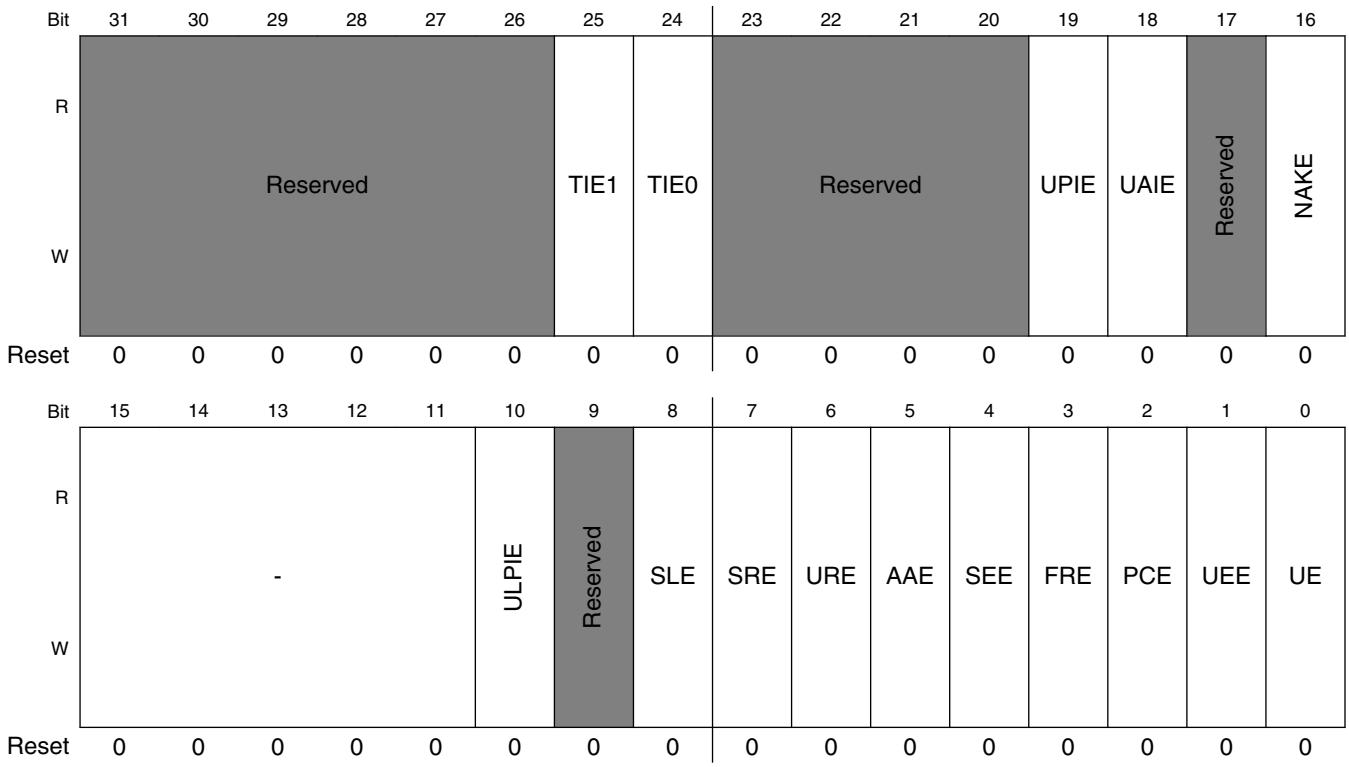
USB_*n*USBSTS field descriptions (continued)

Field	Description
5 AAI	Interrupt on Async Advance - R/WC. System software can force the host controller to issue an interrupt the next time the host controller advances the asynchronous schedule by writing a one to the Interrupt on Async Advance Doorbell bit in the n_USBCMD register. This status bit indicates the assertion of that interrupt source. Only used in host operation mode.
4 SEI	System Error- R/WC. This bit is will be set to '1b' when an Error response is seen to a read on the system interface.
3 FRI	Frame List Rollover - R/WC. The Host Controller sets this bit to a one when the Frame List Index rolls over from its maximum value to zero. The exact value at which the rollover occurs depends on the frame list size. For example. If the frame list size (as programmed in the Frame List Size field of the USB_n_USBCMD register) is 1024, the Frame Index Register rolls over every time FRINDEX [13] toggles. Similarly, if the size is 512, the Host Controller sets this bit to a one every time FHINDEX [12] toggles. Only used in host operation mode.
2 PCI	Port Change Detect - R/WC. The Host Controller sets this bit to a one when on any port a Connect Status occurs, a Port Enable/ Disable Change occurs, or the Force Port Resume bit is set as the result of a J-K transition on the suspended port. The Device Controller sets this bit to a one when the port controller enters the full or high-speed operational state. When the port controller exits the full or high-speed operation states due to Reset or Suspend events, the notification mechanisms are the USB Reset Received bit and the DCSuspend bits respectively.
1 UEI	USB Error Interrupt (USBERRINT) - R/WC. When completion of a USB transaction results in an error condition, this bit is set by the Host/Device Controller. This bit is set along with the USBINT bit, if the TD on which the error interrupt occurred also had its interrupt on complete (IOC) bit set. The device controller detects resume signaling only.
0 UI	USB Interrupt (USBINT) - R/WC. This bit is set by the Host/Device Controller when the cause of an interrupt is a completion of a USB transaction where the Transfer Descriptor (TD) has an interrupt on complete (IOC) bit set. This bit is also set by the Host/Device Controller when a short packet is detected. A short packet is when the actual number of bytes received was less than the expected number of bytes.

42.7.20 Interrupt Enable Register (USB_*n*USBINTR)

The interrupts to software are enabled with this register. An interrupt is generated when a bit is set and the corresponding interrupt source is active. The USB Status register (n_USBSTS) still shows interrupt sources even if they are disabled by the n_USBINTR register, allowing polling of interrupt events by the software.

Address: 402E_0000h base + 148h offset + (512d × i), where i=0d to 1d



USB_*n*USBINTR field descriptions

Field	Description
31–26 -	This field is reserved. Reserved
25 TIE1	General Purpose Timer #1 Interrupt Enable When this bit is one and the TI1 bit in n_USBSTS register is a one the controller will issue an interrupt.
24 TIE0	General Purpose Timer #0 Interrupt Enable When this bit is one and the TI0 bit in n_USBSTS register is a one the controller will issue an interrupt.
23–20 -	This field is reserved. Reserved
19 UPIE	USB Host Periodic Interrupt Enable

Table continues on the next page...

USB_*n*USBINTR field descriptions (continued)

Field	Description
	When this bit is one, and the UPI bit in the n_USBSTS register is one, host controller will issue an interrupt at the next interrupt threshold.
18 UAIE	USB Host Asynchronous Interrupt Enable When this bit is one, and the UAI bit in the n_USBSTS register is one, host controller will issue an interrupt at the next interrupt threshold.
17 -	This field is reserved. Reserved
16 NAKE	NAK Interrupt Enable When this bit is one and the NAKI bit in n_USBSTS register is a one the controller will issue an interrupt.
15–11 -	These bits are reserved and should be set to zero.
10 ULPIE	ULPI Interrupt Enable When this bit is one and the UPLII bit in n_USBSTS register is a one the controller will issue an interrupt. This bit is usable only if the controller support UPLI interface mode.
9 -	This field is reserved. Reserved
8 SLE	Sleep Interrupt Enable When this bit is one and the SLI bit in n_n_USBSTS register is a one the controller will issue an interrupt. Only used in device operation mode.
7 SRE	SOF Received Interrupt Enable When this bit is one and the SRI bit in n_USBSTS register is a one the controller will issue an interrupt.
6 URE	USB Reset Interrupt Enable When this bit is one and the URI bit in n_USBSTS register is a one the controller will issue an interrupt. Only used in device operation mode.
5 AAE	Async Advance Interrupt Enable When this bit is one and the AAI bit in n_USBSTS register is a one the controller will issue an interrupt. Only used in host operation mode.
4 SEE	System Error Interrupt Enable When this bit is one and the SEI bit in n_USBSTS register is a one the controller will issue an interrupt. Only used in host operation mode.
3 FRE	Frame List Rollover Interrupt Enable When this bit is one and the FRI bit in n_USBSTS register is a one the controller will issue an interrupt. Only used in host operation mode.
2 PCE	Port Change Detect Interrupt Enable When this bit is one and the PCI bit in n_USBSTS register is a one the controller will issue an interrupt.
1 UEE	USB Error Interrupt Enable When this bit is one and the UEI bit in n_USBSTS register is a one the controller will issue an interrupt.
0 UE	USB Interrupt Enable When this bit is one and the UI bit in n_USBSTS register is a one the controller will issue an interrupt.

42.7.21 USB Frame Index (USB_*n*FRINDEX)

This register is used by the host controller to index the periodic frame list. The register updates every 125 microseconds (once each micro-frame). Bits [N: 3] are used to select a particular entry in the Periodic Frame List during periodic schedule execution. The number of bits used for the index depends on the size of the frame list as set by system software in the Frame List Size field in the n_USBCMD register.

This register must be written as a DWord. Byte writes produce undefined results. This register cannot be written unless the Host Controller is in the 'Halted' state as indicated by the HCHalted bit. A write to this register while the Run/Stop hit is set to a one produces undefined results. Writes to this register also affect the SOF value.

In device mode this register is read only and, the device controller updates the FRINDEX [13:3] register from the frame number indicated by the SOF marker. Whenever a SOF is received by the USB bus, FRINDEX [13:3] will be checked against the SOF marker. If FRINDEX [13:3] is different from the SOF marker, FRINDEX [13:3] will be set to the SOF value and FRINDEX [2:0] will be set to zero (that is, SOF for 1 ms frame). If FRINDEX [13:3] is equal to the SOF value, FRINDEX [2:0] will be increment (that is, SOF for 125 us micro-frame.).

Address: 402E_0000h base + 14Ch offset + (512d × i), where i=0d to 1d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																FRINDEX															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USB_*n*FRINDEX field descriptions

Field	Description
31–14 -	This field is reserved. Reserved
FRINDEX	Frame Index. The value, in this register, increments at the end of each time frame (micro-frame). Bits [N: 3] are used for the Frame List current index. This means that each location of the frame list is accessed 8 times (frames or micro-frames) before moving to the next index. The following illustrates values of N based on the value of the Frame List Size field in the USBCMD register, when used in host mode. USBCMD [Frame List Size] Number Elements N In device mode the value is the current frame number of the last frame transmitted. It is not used as an index. In either mode bits 2:0 indicate the current microframe. 000 (1024) 12 001 (512) 11

Table continues on the next page...

USB_*n*FRINDEX field descriptions (continued)

Field	Description
	010 (256) 10 011 (128) 9 100 (64) 8 101 (32) 7 110 (16) 6 111 (8) 5

42.7.22 Frame List Base Address (USB_*n*PERIODICLISTBASE)

Host Controller only

This 32-bit register contains the beginning address of the Periodic Frame List in the system memory. HCD loads this register prior to starting the schedule execution by the Host Controller. The memory structure referenced by this physical memory pointer is assumed to be 4-Kbyte aligned. The contents of this register are combined with the Frame Index Register (USB_n_FRINDEX) to enable the Host Controller to step through the Periodic Frame List in sequence.

Address: 402E_0000h base + 154h offset + (512d × i), where i=0d to 1d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BASEADR																				Reserved											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USB_*n*PERIODICLISTBASE field descriptions

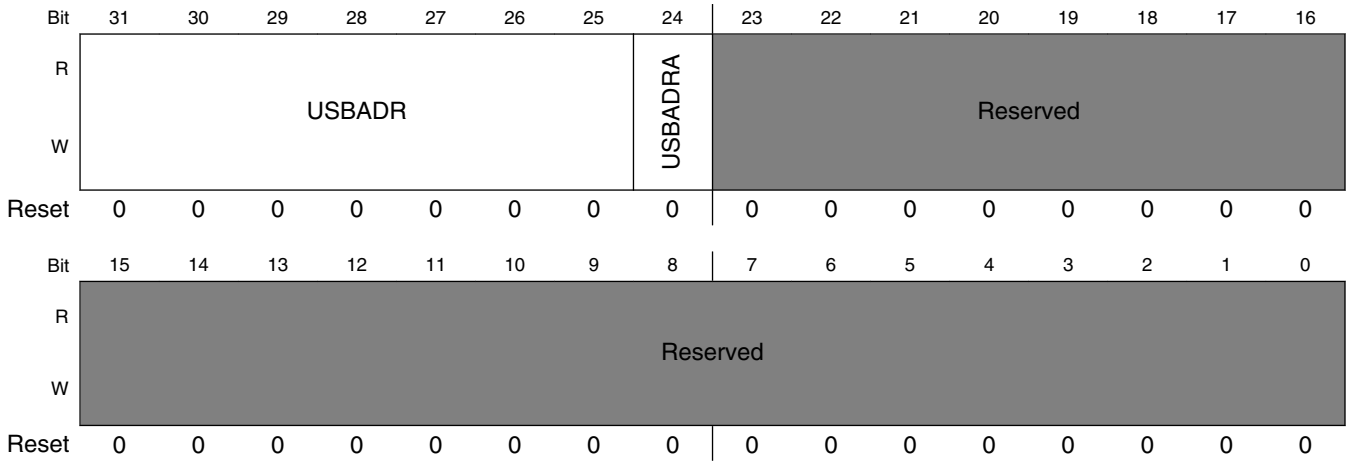
Field	Description
31–12 BASEADR	Base Address (Low). These bits correspond to memory address signals [31:12], respectively. Only used by the host controller.
-	This field is reserved. Reserved

42.7.23 Device Address (USB_*n*DEVICEADDR)

Device Controller only

The upper seven bits of this register represent the device address. After any controller reset or a USB reset, the device address is set to the default address (0). The default address will match all incoming addresses. Software shall reprogram the address after receiving a SET_ADDRESS descriptor.

Address: 402E_0000h base + 154h offset + (512d × i), where i=0d to 1d



USB_*n*DEVICEADDR field descriptions

Field	Description
31–25 USBADR	Device Address. These bits correspond to the USB device address
24 USBADRA	Device Address Advance. Default=0. When this bit is '0', any writes to USBADR are instantaneous. When this bit is written to a '1' at the same time or before USBADR is written, the write to the USBADR field is staged and held in a hidden register. After an IN occurs on endpoint 0 and is ACKed, USBADR will be loaded from the holding register. Hardware will automatically clear this bit on the following conditions: 1) IN is ACKed to endpoint 0. (USBADR is updated from staging register). 2) OUT/SETUP occur to endpoint 0. (USBADR is not updated). 3) Device Reset occurs (USBADR is reset to 0). **NOTE:** After the status phase of the SET_ADDRESS descriptor, the DCD has 2 ms to program the USBADR field. This mechanism will ensure this specification is met when the DCD can not write of the device address within 2ms from the SET_ADDRESS status phase. If the DCD writes the USBADR with USBADRA=1 after the SET_ADDRESS data phase (before the prime of the status phase), the USBADR will be programmed instantly at the correct time and meet the 2ms USB requirement.
-	This field is reserved. Reserved

42.7.24 Next Asynch. Address (USB_*n*ASYNCLISTADDR)

Host Controller only

This 32-bit register contains the address of the next asynchronous queue head to be executed by the host. Bits [4:0] of this register cannot be modified by the system software and will always return a zero when read.

Address: 402E_0000h base + 158h offset + (512d × i), where i=0d to 1d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	ASYBASE																											Reserved				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USB_*n*ASYNCLISTADDR field descriptions

Field	Description
31–5 ASYBASE	Link Pointer Low (LPL). These bits correspond to memory address signals [31:5], respectively. This field may only reference a Queue Head (QH). Only used by the host controller.
-	This field is reserved. Reserved

42.7.25 Endpoint List Address (USB_*n*ENDPTLISTADDR)

Device Controller only

In device mode, this register contains the address of the top of the endpoint list in system memory. Bits [10:0] of this register cannot be modified by the system software and will always return a zero when read.

The memory structure referenced by this physical memory pointer is assumed 64-byte.

Address: 402E_0000h base + 158h offset + (512d × i), where i=0d to 1d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	EPBASE																					Reserved										
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USB_*n*ENDPTLISTADDR field descriptions

Field	Description
31–11 EPBASE	Endpoint List Pointer(Low). These bits correspond to memory address signals [31:11], respectively. This field will reference a list of up to 32 Queue Head (QH) (that is, one queue head per endpoint & direction).
-	This field is reserved. Reserved

42.7.26 Programmable Burst Size (USB_*n*BURSTSIZE)

This register is used to control the burst size used during data movement on the AHB master interface. This register is ignored if AHBBRST bits in SBUSCFG register is non-zero value.

Address: 402E_0000h base + 160h offset + (512d × i), where i=0d to 1d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																TXPBURST								RXPBURST							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0

USB_*n*BURSTSIZE field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16–8 TXPBURST	Programmable TX Burst Size. Default value is determined by TXBURST bits in n_HWTXBUF. This register represents the maximum length of a the burst in 32-bit words while moving data from system memory to the USB bus.
RXPBURST	Programmable RX Burst Size. Default value is determined by TXBURST bits in n_HWRXBUF. This register represents the maximum length of a the burst in 32-bit words while moving data from the USB bus to system memory.

42.7.27 TX FIFO Fill Tuning (USB_*n*TXFILLTUNING)

The fields in this register control performance tuning associated with how the host controller posts data to the TX latency FIFO before moving the data onto the USB bus. The specific areas of performance include the how much data to post into the FIFO and an estimate for how long that operation should take in the target system.

Definitions:

T_0 = Standard packet overhead

T_1 = Time to send data payload

T_{ff} = Time to fetch packet into TX FIFO up to specified level.

T_s = Total Packet Flight Time (send-only) packet

$T_s = T_0 + T_1$

T_p = Total Packet Time (fetch and send) packet

$T_p = T_{ff} + T_0 + T_1$

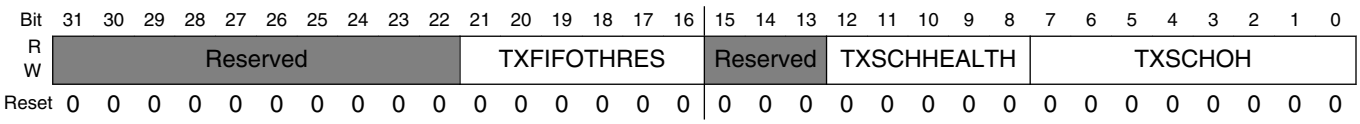
Upon discovery of a transmit (OUT/SETUP) packet in the data structures, host controller checks to ensure T_p remains before the end of the [micro]frame. If so it proceeds to pre-fill the TX FIFO. If at anytime during the pre-fill operation the time remaining the [micro]frame is < T_s then the packet attempt ceases and the packet is tried at a later time. Although this is not an error condition and the host controller will eventually recover, a

mark will be made the scheduler health counter to note the occurrence of a "back-off" event. When a back-off event is detected, the partial packet fetched may need to be discarded from the latency buffer to make room for periodic traffic that will begin after the next SOF. Too many back-off events can waste bandwidth and power on the system bus and thus should be minimized (not necessarily eliminated). Back-offs can be minimized with use of the n_TSCHHEALTH (T_{ff}) described below.

NOTE

The reset value could vary from instance to instance. Please see the detail in bit field description and ignore reset value in summary table in this case!

Address: 402E_0000h base + 164h offset + (512d × i), where i=0d to 1d

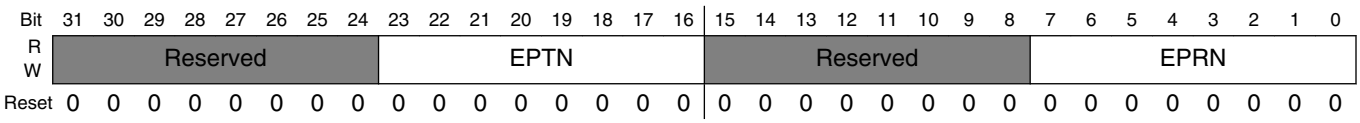


USB_*n*TXFILLTUNING field descriptions

Field	Description
31–22 -	This field is reserved. Reserved
21–16 TXFIFOTHRES	FIFO Burst Threshold. (Read/Write) This register controls the number of data bursts that are posted to the TX latency FIFO in host mode before the packet begins on to the bus. The minimum value is 2 and this value should be a low as possible to maximize USB performance. A higher value can be used in systems with unpredictable latency and/or insufficient bandwidth where the FIFO may underrun because the data transferred from the latency FIFO to USB occurs before it can be replenished from system memory. This value is ignored if the Stream Disable bit in USB_n_USBMODE register is set. Default value is '0Ah for OTG controller core.
15–13 -	This field is reserved. Reserved
12–8 TXSCHHEALTH	Scheduler Health Counter. (Read/Write To Clear) This register increments when the host controller fails to fill the TX latency FIFO to the level programmed by TXFIFOTHRES before running out of time to send the packet before the next Start-Of-Frame. This health counter measures the number of times this occurs to provide feedback to selecting a proper TXSCHOH. Writing to this register will clear the counter and this counter will max. at 31. Default value is '00h' for OTG controller core.
TXSCHOH	Scheduler Overhead. (Read/Write) [Default = 0] This register adds an additional fixed offset to the schedule time estimator described above as Tff. As an approximation, the value chosen for this register should limit the number of back-off events captured in the TXSCHHEALTH to less than 10 per second in a highly utilized bus. Choosing a value that is too high for this register is not desired as it can needlessly reduce USB utilization. The time unit represented in this register is 1.267us when a device is connected in High-Speed Mode. The time unit represented in this register is 6.333us when a device is connected in Low/Full Speed Mode. Default value is '08h' for OTG controller core.

42.7.28 Endpoint NAK (USB_*n*ENDPTNAK)

Address: 402E_0000h base + 178h offset + (512d × i), where i=0d to 1d

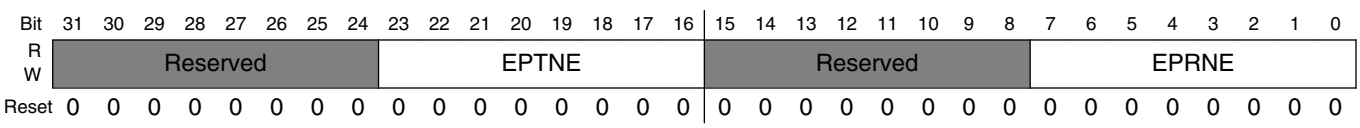


USB_*n*ENDPTNAK field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23–16 EPTN	TX Endpoint NAK - R/WC. Each TX endpoint has 1 bit in this field. The bit is set when the device sends a NAK handshake on a received IN token for the corresponding endpoint. Bit [N] - Endpoint #[N], N is 0-7
15–8 -	This field is reserved. Reserved
EPRN	RX Endpoint NAK - R/WC. Each RX endpoint has 1 bit in this field. The bit is set when the device sends a NAK handshake on a received OUT or PING token for the corresponding endpoint. Bit [N] - Endpoint #[N], N is 0-7

42.7.29 Endpoint NAK Enable (USB_*n*ENDPTNAKEN)

Address: 402E_0000h base + 17Ch offset + (512d × i), where i=0d to 1d



USB_*n*ENDPTNAKEN field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23–16 EPTNE	TX Endpoint NAK Enable - R/W. Each bit is an enable bit for the corresponding TX Endpoint NAK bit. If this bit is set and the corresponding TX Endpoint NAK bit is set, the NAK Interrupt bit is set. Bit [N] - Endpoint #[N], N is 0-7
15–8 -	This field is reserved. Reserved
EPRNE	RX Endpoint NAK Enable - R/W.

Table continues on the next page...

USB_*n*ENDPTNAKEN field descriptions (continued)

Field	Description
	Each bit is an enable bit for the corresponding RX Endpoint NAK bit. If this bit is set and the corresponding RX Endpoint NAK bit is set, the NAK Interrupt bit is set. Bit [N] - Endpoint #[N], N is 0-7

42.7.30 Configure Flag Register (USB_*n*CONFIGFLAG)

Address: 402E_0000h base + 180h offset + (512d × i), where i=0d to 1d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved															CF
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

USB_*n*CONFIGFLAG field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 CF	Configure Flag Host software sets this bit as the last action in its process of configuring the Host Controller. This bit controls the default port-routing control logic. 0 Port routing control logic default-routes each port to an implementation dependent classic host controller. 1 Port routing control logic default-routes all ports to this host controller.

42.7.31 Port Status & Control (USB_*n*PORTSC1)

Host Controller

A host controller could implement one to eight port status and control registers. The number is determined by N_PORTs bits in HWSPARAMs register (please see Host Controller Structural Parameters (USB_*n*HCSPARAMS)). Software could read this parameter register to determine how many ports need service.

All controller cores on this product are Single-Port, so there is only one port status and control register for each controller core.

This register is only reset by power on reset or controller core reset. The initial conditions of a port are:

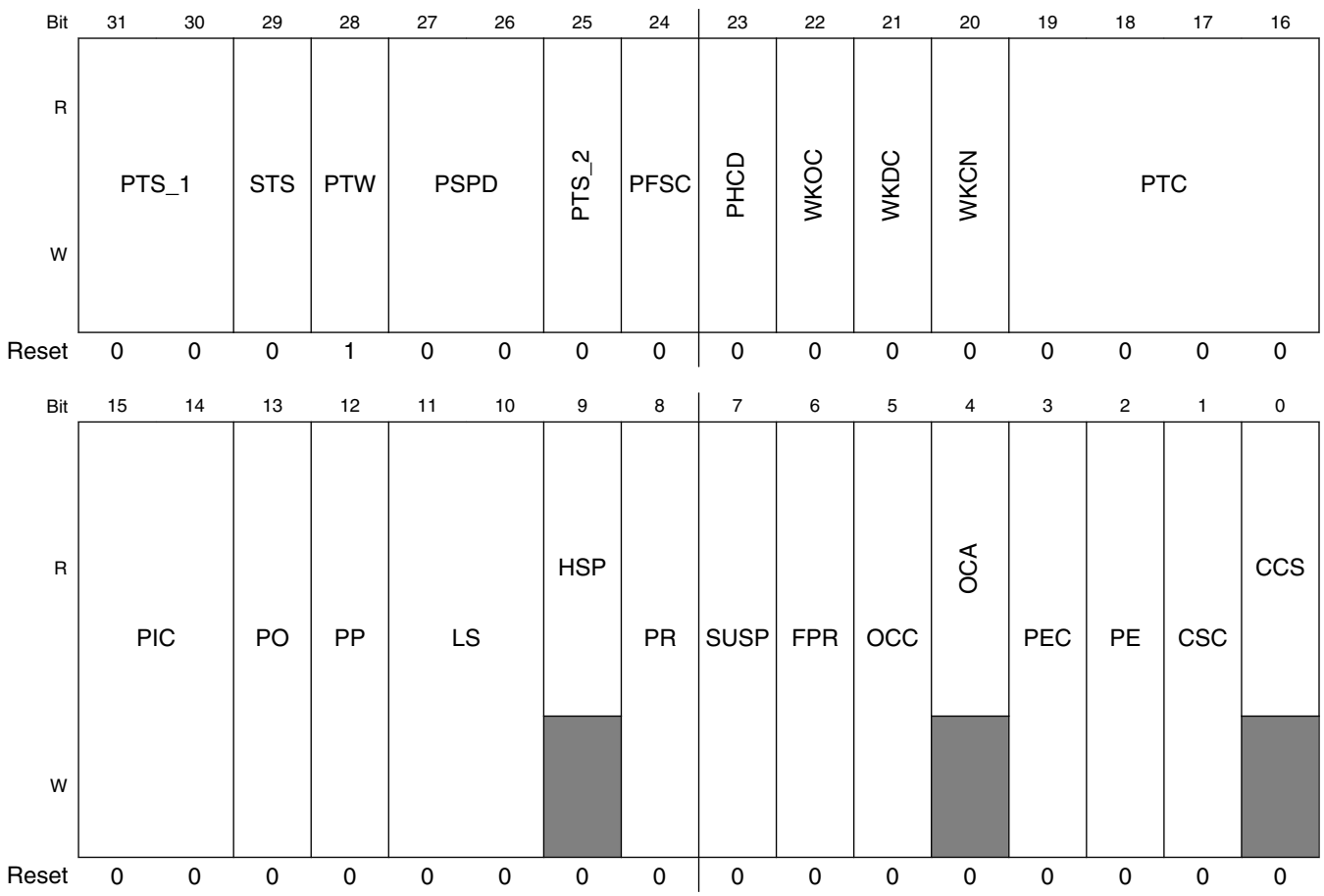
- No device connected
- Port disabled

If the port supports power control, this state remains until port power is supplied (by software).

Device Controller

A controller operating in device mode has only port register one (PORTSC1) and it does not support power control in that mode. Port control in device mode is only used for status port reset, suspend, and current connect status. It is also used to initiate test mode or force signaling and allows software to put the PHY into low power suspend mode and disable the PHY clock.

Address: 402E_0000h base + 184h offset + (512d × i), where i=0d to 1d



USB_*n*PORTSC1 field descriptions

Field	Description
31–30 PTS_1	**NOTE:** All USB port interface modes are listed in this field description, but not all are supported. For detail feature of each controller core, please see Features . The behaviour is unknown when unsupported interface mode is selected.
29 STS	Serial Transceiver Select 1 Serial Interface Engine is selected 0 Parallel Interface signals is selected Serial Interface Engine can be used in combination with UTMI+/ULPI physical interface to provide FS/LS signaling instead of the parallel interface signals. When this bit is set '1b', serial interface engine will be used instead of parallel interface signals. This bit has no effect unless PTS bits is set to select UTMI+/ULPI interface. The Serial/USB1.1 PHY/IC-USB will use the serial interface engine for FS/LS signaling regardless of this bit value.
28 PTW	Parallel Transceiver Width This bit has no effect if serial interface engine is used. For OTG1/OTG2 core, it is Read-Only. Reset value is '1b'. 0 Select the 8-bit UTMI interface [60MHz] 1 Select the 16-bit UTMI interface [30MHz]
27–26 PSPD	Port Speed - Read Only. This register field indicates the speed at which the port is operating. 00 Full Speed 01 Low Speed 10 High Speed 11 Undefined
25 PTS_2	See description at bits 31-30
24 PFSC	Port Force Full Speed Connect - Read/Write. Default = 0b. When this bit is set to '1b', the port will be forced to only connect at Full Speed, It disables the chirp sequence that allows the port to identify itself as High Speed. 1 Forced to full speed 0 Normal operation
23 PHCD	PHY Low Power Suspend - Clock Disable (PLPSCD) - Read/Write. Default = 0b. When this bit is set to '1b', the PHY clock is disabled. Reading this bit will indicate the status of the PHY clock. **NOTE:** The PHY clock cannot be disabled if it is being used as the system clock. In device mode, The PHY can be put into Low Power Suspend when the device is not running (USBCMD Run/Stop=0b) or the host has signalled suspend (PORTSC1 SUSPEND=1b). PHY Low power suspend will be cleared automatically when the host initials resume. Before forcing a resume from the device, the device controller driver must clear this bit. In host mode, the PHY can be put into Low Power Suspend when the downstream device has been put into suspend mode or when no downstream device is connected. Low power suspend is completely under the control of software.

Table continues on the next page...

USB_*n*PORTSC1 field descriptions (continued)

Field	Description
	1 Disable PHY clock 0 Enable PHY clock
22 WKOC	Wake on Over-current Enable (WKOC_E) - Read/Write. Default = 0b. Writing this bit to a one enables the port to be sensitive to over-current conditions as wake-up events. This field is zero if *Port Power(*Port Status & Control (USB_*n*PORTSC1)*)* is zero.
21 WKDC	Wake on Disconnect Enable (WKDSCNNT_E) - Read/Write. Default=0b. Writing this bit to a one enables the port to be sensitive to device disconnects as wake-up events. This field is zero if *Port Power(*Port Status & Control (USB_*n*PORTSC1)*)* is zero or in device mode.
20 WKCN	Wake on Connect Enable (WKCNNT_E) - Read/Write. Default=0b. Writing this bit to a one enables the port to be sensitive to device connects as wake-up events. This field is zero if *Port Power(*Port Status & Control (USB_*n*PORTSC1)*)* is zero or in device mode.
19–16 PTC	Port Test Control - Read/Write. Default = 0000b. Refer to Port Test Mode for the operational model for using these test modes and the USB Specification Revision 2.0, Chapter 7 for details on each test mode. The FORCE_ENABLE_FS and FORCE ENABLE_LS are extensions to the test mode support specified in the EHCI specification. Writing the PTC field to any of the FORCE_ENABLE_{HS/FS/LS} values will force the port into the connected and enabled state at the selected speed. Writing the PTC field back to TEST_MODE_DISABLE will allow the port state machines to progress normally from that point. **NOTE:** *Low speed operations are not supported as a peripheral device.* Any other value than zero indicates that the port is operating in test mode. Value Specific Test 0000 TEST_MODE_DISABLE 0001 J_STATE 0010 K_STATE 0011 SE0 (host) / NAK (device) 0100 Packet 0101 FORCE_ENABLE_HS 0110 FORCE_ENABLE_FS 0111 FORCE_ENABLE_LS 1000-1111 Reserved
15–14 PIC	Port Indicator Control - Read/Write. Default = Ob. Writing to this field has no effect if the P_INDICATOR bit in the HCSPARAMS register is a zero. Refer to the USB Specification Revision 2.0 for a description on how these bits are to be used. This field is zero if *Port Power* is zero. Bit Value Meaning 00 Port indicators are off 01 Amber 10 Green 11 Undefined
13 PO	Port Owner-Read/Write. Default = 0.

Table continues on the next page...

USB_*n*PORTSC1 field descriptions (continued)

Field	Description
	This bit unconditionally goes to a 0 when the configured bit in the CONFIGFLAG register makes a 0 to 1 transition. This bit unconditionally goes to 1 whenever the Configured bit is zero System software uses this field to release ownership of the port to a selected host controller (in the event that the attached device is not a high-speed device). Software writes a one to this bit when the attached device is not a high-speed device. A one in this bit means that an internal companion controller owns and controls the port. Port owner handoff is not supported in all controller cores, therefore this bit will always be 0.
12 PP	Port Power (PP)-Read/Write or Read Only. The function of this bit depends on the value of the Port Power Switching (PPC) field in the HCSPARAMS register. The behavior is as follows: PPC PP Operation 0 1b *Read Only - Host controller does not have port power control switches. Each port is hard-wired to power.* 1 1b/0b - *Read/Write. OTG controller requires port power control switches. This bit represents the current setting of the switch (0=off, 1=on). When power is not available on a port (that is, PP equals a 0), the port is non-functional and will not report attaches, detaches, etc.* When an over-current condition is detected on a powered port and PPC is a one, the PP bit in each affected port may be transitional by the host controller driver from a one to a zero (removing power from the port). This feature is implemented in all controller cores (PPC = 1).
11–10 LS	Line Status-Read Only. These bits reflect the current logical levels of the D+ (bit 11) and D- (bit 10) signal lines. In host mode, the use of linestate by the host controller driver is not necessary (unlike EHCI), because the port controller state machine and the port routing manage the connection of LS and FS. In device mode, the use of linestate by the device controller driver is not necessary. The encoding of the bits are: Bits [11:10] Meaning 00 SE0 10 J-state 01 K-state 11 Undefined
9 HSP	High-Speed Port - Read Only. Default = 0b. When the bit is one, the host/device connected to the port is in high-speed mode and if set to zero, the host/device connected to the port is not in a high-speed mode. **NOTE:** HSP is redundant with PSPD(bit 27, 26) but remained for compatibility.
8 PR	Port Reset - Read/Write or Read Only. Default = 0b. In Host Mode: Read/Write. 1=Port is in Reset. 0=Port is not in Reset. Default 0. When software writes a one to this bit the bus-reset sequence as defined in the USB Specification Revision 2.0 is started. *This bit will automatically change to zero after the reset sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a zero after the reset duration is timed in the driver.*

Table continues on the next page...

USB_*n*PORTSC1 field descriptions (continued)

Field	Description
	In Device Mode: This bit is a read only status bit. Device reset from the USB bus is also indicated in the USBSTS register. This field is zero if *Port Power(*Port Status & Control (USB_*n*PORTSC1)*)* is zero.
7 SUSP	Suspend - Read/Write or Read Only. Default = 0b. 1=Port in suspend state. 0=Port not in suspend state. In Host Mode: Read/Write. Port Enabled Bit and Suspend bit of this register define the port states as follows: Bits [Port Enabled, Suspend] Port State 0x Disable 10 Enable 11 Suspend When in suspend state, downstream propagation of data is blocked on this port, except for port reset. The blocking occurs at the end of the current transaction if a transaction was in progress when this bit was written to 1. In the suspend state, the port is sensitive to resume detection. Note that the bit status does not change until the port is suspended and that there may be a delay in suspending a port if there is a transaction currently in progress on the USB. The host controller will unconditionally set this bit to zero when software sets the *Force Port Resume* bit to zero. The host controller ignores a write of zero to this bit. If host software sets this bit to a one when the port is not enabled (that is, *Port enabled* bit is a zero) the results are undefined. This field is zero if *Port Power(*Port Status & Control (USB_*n*PORTSC1)*)* is zero in host mode. In Device Mode: Read Only. In device mode this bit is a read only status bit.
6 FPR	Force Port Resume -Read/Write. 1= Resume detected/driven on port. 0=No resume (K-state) detected/driven on port. Default = 0. In Host Mode: Software sets this bit to one to drive resume signaling. The Host Controller sets this bit to one if a J-to-K transition is detected while the port is in the Suspend state. When this bit transitions to a one because a J-to-K transition is detected, the *Port Change Detect* bit in the USBSTS register is also set to one. *This bit will automatically change to zero after the resume sequence is complete. This behavior is different from EHCI where the host controller driver is required to set this bit to a zero after the resume duration is timed in the driver.* Note that when the Host controller owns the port, the resume sequence follows the defined sequence documented in the USB Specification Revision 2.0. The resume signaling (Full-speed 'K') is driven on the port as long as this bit remains a one. This bit will remain a one until the port has switched to the high-speed idle. Writing a zero has no effect because the port controller will time the resume operation, clear the bit the port control state switches to HS or FS idle. This field is zero if *Port Power(*Port Status & Control (USB_*n*PORTSC1)*)* is zero in host mode. This bit is not-EHCI compatible. In Device mode:

Table continues on the next page...

USB_*n*PORTSC1 field descriptions (continued)

Field	Description
	After the device has been in Suspend State for 5ms or more, software must set this bit to one to drive resume signaling before clearing. The Device Controller will set this bit to one if a J-to-K transition is detected while the port is in the Suspend state. The bit will be cleared when the device returns to normal operation. Also, when this bit wil be cleared because a K-to-J transition detected, the *Port Change Detect* bit in the USBSTS register is also set to one.
5 OCC	Over-current Change-R/WC. Default=0. This bit is set '1b' by hardware when there is a change to Over-current Active. Software can clear this bit by writing a one to this bit position.
4 OCA	Over-current Active-Read Only. Default 0. This bit will automatically transition from one to zero when the over current condition is removed. 1 This port currently has an over-current condition 0 This port does not have an over-current condition.
3 PEC	Port Enable/Disable Change-R/WC. 1=Port enabled/disabled status has changed. 0=No change. Default = 0. In Host Mode: For the root hub, this bit is set to a one only when a port is disabled due to disconnect on the port or due to the appropriate conditions existing at the EOF2 point (See Chapter 11 of the USB Specification). Software clears this by writing a one to it. This field is zero if *Port Power(*Port Status & Control (USB_*n*PORTSC1)*)* is zero. In Device mode: The device port is always enabled, so this bit is always '0b'.
2 PE	Port Enabled/Disabled-Read/Write. 1=Enable. 0=Disable. Default 0. In Host Mode: Ports can only be enabled by the host controller as a part of the reset and enable. Software cannot enable a port by writing a one to this field. Ports can be disabled by either a fault condition (disconnect event or other fault condition) or by the host software. Note that the bit status does not change until the port state actually changes. There may be a delay in disabling or enabling a port due to other host controller and bus events. When the port is disabled, (0b) downstream propagation of data is blocked except for reset. This field is zero if *Port Power(*Port Status & Control (USB_*n*PORTSC1)*)* is zero in host mode. In Device Mode: The device port is always enabled, so this bit is always '1b'.
1 CSC	Connect Status Change-R/WC. 1 =Change in Current Connect Status. 0=No change. Default 0. In Host Mode: Indicates a change has occurred in the port's Current Connect Status. The host/device controller sets this bit for all changes to the port device connect status, even if system software has not cleared an existing connect status change. For example, the insertion status changes twice before system software has cleared the changed condition, hub hardware will be 'setting' an already-set bit (that is, the bit will remain set). Software clears this bit by writing a one to it. This field is zero if *Port Power(*Port Status & Control (USB_*n*PORTSC1)*)* is zero in host mode. In Device Mode: This bit is undefined in device controller mode.

Table continues on the next page...

USB_*n*PORTSC1 field descriptions (continued)

Field	Description
0 CCS	Current Connect Status-Read Only. In Host Mode: 1=Device is present on port. 0=No device is present. Default = 0. This value reflects the current state of the port, and may not correspond directly to the event that caused the *Connect Status Change* bit (Bit 1) to be set. This field is zero if *Port Power(*Port Status & Control (USB_*n*PORTSC1)*)* is zero in host mode. In Device Mode: 1=Attached. 0=Not Attached. Default=0. A one indicates that the device successfully attached and is operating in either high speed or full speed as indicated by the High Speed Port bit in this register. A zero indicates that the device did not attach successfully or was forcibly disconnected by the software writing a zero to the Run bit in the USBCMD register. It does not state the device being disconnected or suspended.

42.7.32 On-The-Go Status & control (USB_*n*OTGSC)

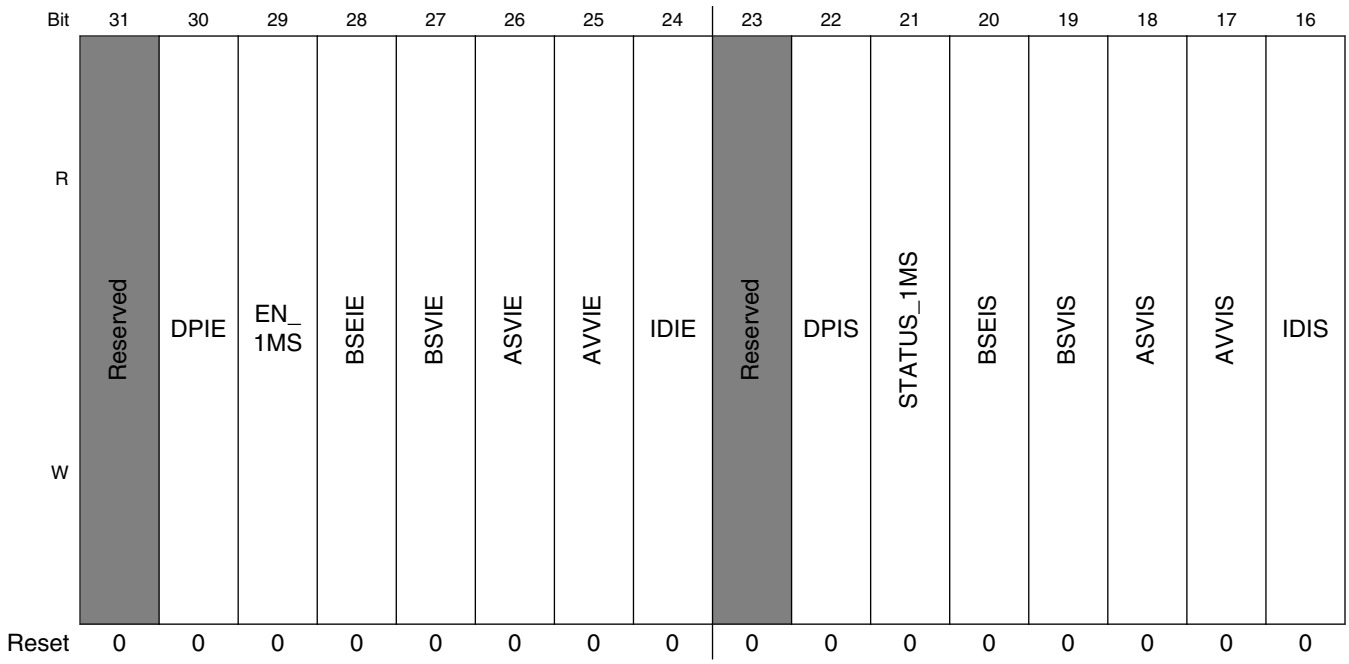
This register is availabe only in OTG controller core. It has four sections:

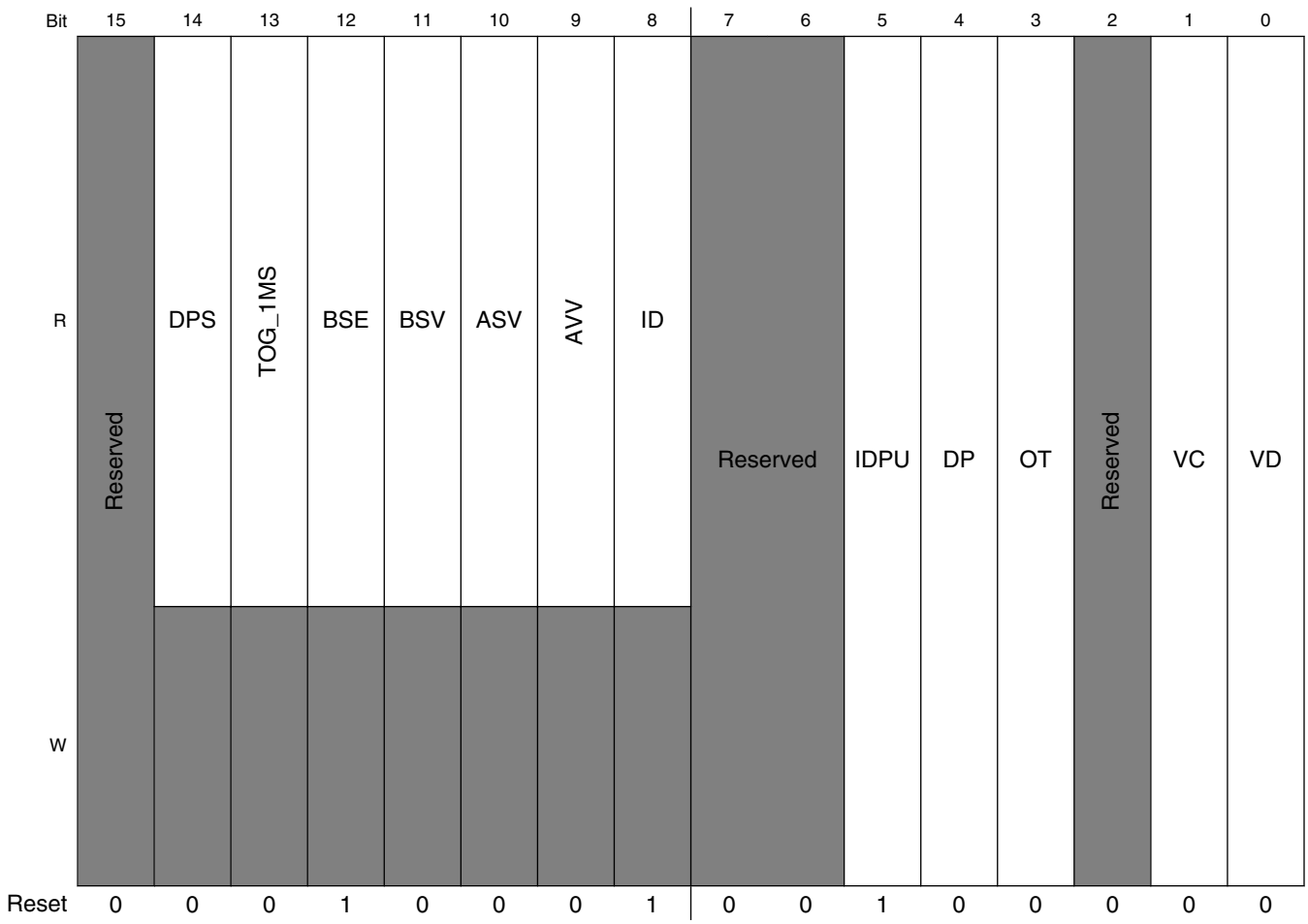
- OTG Interrupt enables (Read/Write)
- OTG Interrupt status (Read/Write to Clear)
- OTG Status inputs (Read Only)
- OTG Controls (Read/Write)

The status inputs are debounced using a 1 ms time constant. Values on the status inputs that do not persist for more than 1 ms does not cause an update of the status input register, or cause an OTG interrupt.

See also USB Device Mode (USB_*n*USBMODE) register.

Address: 402E_0000h base + 1A4h offset + (512d × i), where i=0d to 1d





USB_*n*OTGSC field descriptions

Field	Description
31 -	This field is reserved. Reserved
30 DPIE	Data Pulse Interrupt Enable
29 EN_1MS	1 millisecond timer Interrupt Enable - Read/Write
28 BSEIE	B Session End Interrupt Enable - Read/Write. Setting this bit enables the B session end interrupt.
27 BSVIE	B Session Valid Interrupt Enable - Read/Write. Setting this bit enables the B session valid interrupt.
26 ASVIE	A Session Valid Interrupt Enable - Read/Write. Setting this bit enables the A session valid interrupt.
25 AVVIE	A VBus Valid Interrupt Enable - Read/Write. Setting this bit enables the A VBus valid interrupt.
24 IDIE	USB ID Interrupt Enable - Read/Write. Setting this bit enables the USB ID interrupt.
23 -	This field is reserved. Reserved
22 DPIS	Data Pulse Interrupt Status - Read/Write to Clear. This bit is set when data bus pulsing occurs on DP or DM. Data bus pulsing is only detected when USBMODE.CM = Host (11) and PORTSC1(0)[PP] = 0. Software must write a one to clear this bit.
21 STATUS_1MS	1 millisecond timer Interrupt Status - Read/Write to Clear. This bit is set once every millisecond. Software must write a one to clear this bit.
20 BSEIS	B Session End Interrupt Status - Read/Write to Clear. This bit is set when VBus has fallen below the B session end threshold. Software must write a one to clear this bit.
19 BSVIS	B Session Valid Interrupt Status - Read/Write to Clear. This bit is set when VBus has either risen above or fallen below the B session valid threshold. Software must write a one to clear this bit.
18 ASVIS	A Session Valid Interrupt Status - Read/Write to Clear. This bit is set when VBus has either risen above or fallen below the A session valid threshold. Software must write a one to clear this bit.
17 AVVIS	A VBus Valid Interrupt Status - Read/Write to Clear. This bit is set when VBus has either risen above or fallen below the VBus valid threshold on an A device. Software must write a one to clear this bit.
16 IDIS	USB ID Interrupt Status - Read/Write. This bit is set when a change on the ID input has been detected. Software must write a one to clear this bit.

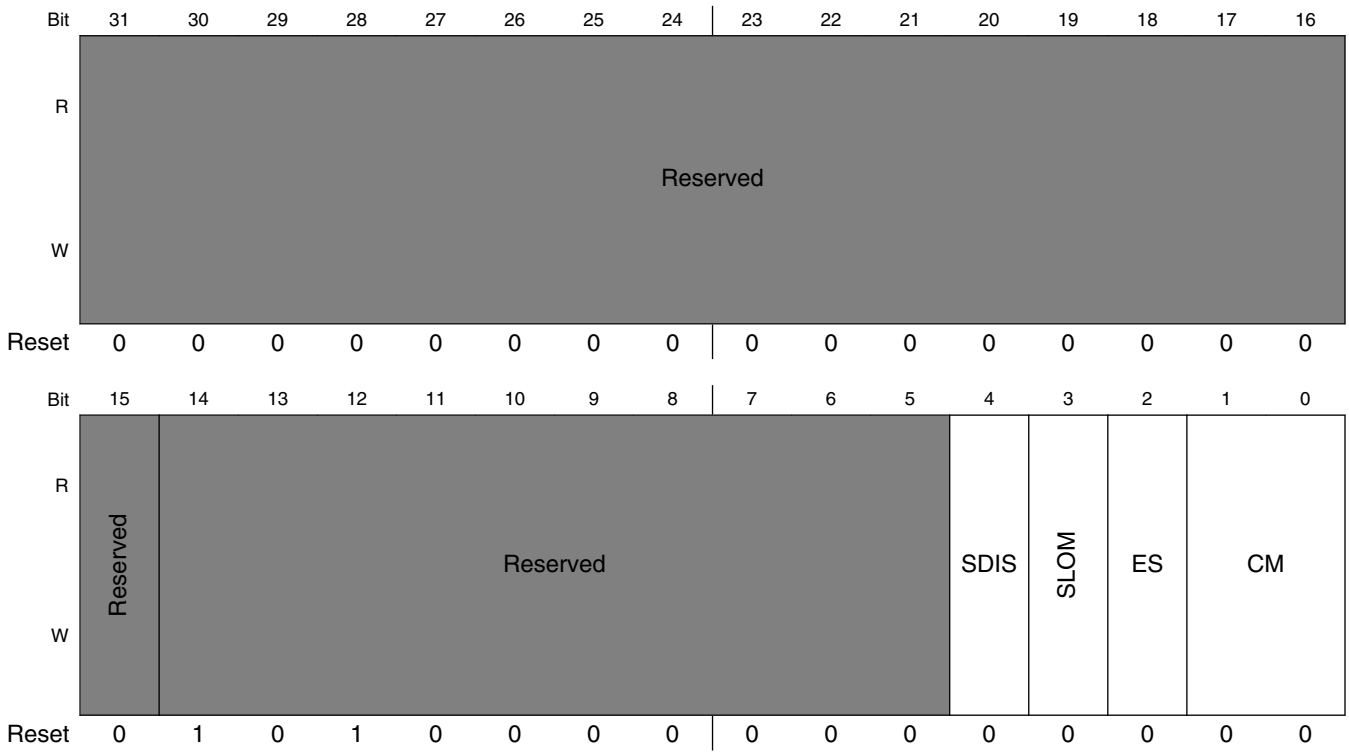
Table continues on the next page...

USB_*n*OTGSC field descriptions (continued)

Field	Description
15 -	This field is reserved. Reserved
14 DPS	Data Bus Pulsing Status - Read Only. A '1' indicates data bus pulsing is being detected on the port.
13 TOG_1MS	1 millisecond timer toggle - Read Only. This bit toggles once per millisecond.
12 BSE	B Session End - Read Only. Indicates VBus is below the B session end threshold.
11 BSV	B Session Valid - Read Only. Indicates VBus is above the B session valid threshold.
10 ASV	A Session Valid - Read Only. Indicates VBus is above the A session valid threshold.
9 AVV	A VBus Valid - Read Only. Indicates VBus is above the A VBus valid threshold.
8 ID	USB ID - Read Only. 0 = A device, 1 = B device
7–6 -	This field is reserved. Reserved
5 IDPU	ID Pullup - Read/Write This bit provide control over the ID pull-up resistor; 0 = off, 1 = on [default]. When this bit is 0, the ID input will not be sampled.
4 DP	Data Pulsing - Read/Write. Setting this bit causes the pullup on DP to be asserted for data pulsing during SRP.
3 OT	OTG Termination - Read/Write. This bit must be set when the OTG device is in device mode, this controls the pulldown on DM.
2 -	This field is reserved. Reserved
1 VC	VBUS Charge - Read/Write. Setting this bit causes the VBus line to be charged. This is used for VBus pulsing during SRP.
0 VD	VBUS_Discharge - Read/Write. Setting this bit causes VBus to discharge through a resistor.

42.7.33 USB Device Mode (USB_*n*USBMODE)

Address: 402E_0000h base + 1A8h offset + (512d × i), where i=0d to 1d



USB_*n*USBMODE field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
15 -	This field is reserved. Reserved
14–5 -	This field is reserved. Reserved
4 SDIS	Stream Disable Mode. (0 - Inactive [default]; 1 - Active) Device Mode: Setting to a '1' disables double priming on both RX and TX for low bandwidth systems. This mode ensures that when the RX and TX buffers are sufficient to contain an entire packet that the standard double buffering scheme is disabled to prevent overruns/underruns in bandwidth limited systems. Note: In High Speed Mode, all packets received are responded to with a NYET handshake when stream disable is active. Host Mode: Setting to a '1' ensures that overruns/underruns of the latency FIFO are eliminated for low bandwidth systems where the RX and TX buffers are sufficient to contain the entire packet. Enabling stream disable also has the effect of ensuring the TX latency is filled to capacity before the packet is launched onto the USB. **NOTE:** Time duration to pre-fill the FIFO becomes significant when stream disable is active. See TX FIFO Fill Tuning (USB_*n*TXFILLTUNING) and TXTTFILLTUNING [MPH Only] to characterize the adjustments needed for the scheduler when using this feature.

Table continues on the next page...

USB_*n*USBMODE field descriptions (continued)

Field	Description
	NOTE: The use of this feature substantially limits of the overall USB performance that can be achieved.
3 SLOM	Setup Lockout Mode. In device mode, this bit controls behavior of the setup lock mechanism. See Control Endpoint Operation Model . 0 Setup Lockouts On (default); 1 Setup Lockouts Off (DCD requires use of Setup Data Buffer Tripwire in USB Command Register (USB_*n*USBCMD) .
2 ES	Endian Select - Read/Write. This bit can change the byte alignment of the transfer buffers to match the host microprocessor. The bit fields in the microprocessor interface and the data structures are unaffected by the value of this bit because they are based upon the 32-bit word. Bit Meaning 0 Little Endian [Default] 1 Big Endian
CM	Controller Mode - R/WO. Controller mode is defaulted to the proper mode for host only and device only implementations. For those designs that contain both host & device capability, the controller defaults to an idle state and needs to be initialized to the desired operating mode after reset. For combination host/device controllers, this register can only be written once after reset. If it is necessary to switch modes, software must reset the controller by writing to the *RESET* bit in the USBCMD register before reprogramming this register. For OTG controller core, reset value is '00b'. 00 Idle [Default for combination host/device] 01 Reserved 10 Device Controller [Default for device only controller] 11 Host Controller [Default for host only controller]

42.7.34 Endpoint Setup Status (USB_*n*ENDPTSETUPSTAT)

Address: 402E_0000h base + 1ACh offset + (512d × i), where i=0d to 1d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																ENDPTSETUPSTAT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USB_*n*ENDPTSETUPSTAT field descriptions

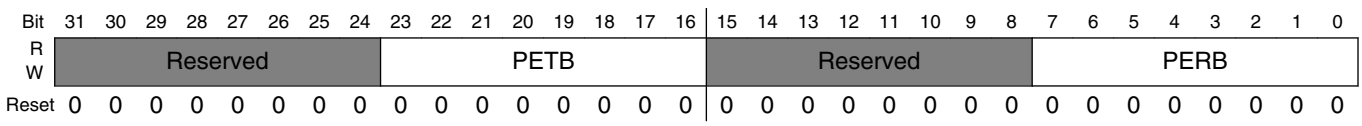
Field	Description
31–16 -	This field is reserved. Reserved
ENDPTSETUPSTAT	Setup Endpoint Status. For every setup transaction that is received, a corresponding bit in this register is set to one. Software must clear or acknowledge the setup transfer by writing a one to a respective bit after it has read the setup data from Queue head. The response to a setup packet as in the order of operations and total response time is crucial to limit bus time outs while the setup lock our mechanism is engaged. See Managing Endpoints in the Device Operational Model. This register is only used in device mode.

42.7.35 Endpoint Prime (USB_*n*ENDPTPRIME)

This register is only used in device mode.

When software sets the prime bit for a given endpoint, the device controller loads the transfer descriptor, pointed to by the queue head, such that the endpoint is ready to transmit or receive when the host sends a request (IN/OUT token). The endpoint will NAK all requests from the host until the endpoint is primed. The controller will automatically re-prime the endpoint with a new transfer descriptor when one is found via the next_dtd pointer of the current transfer descriptor. Hence, the prime bit must only be set by software when a descriptor is added to the queue head.

Address: 402E_0000h base + 1B0h offset + (512d × i), where i=0d to 1d



USB_*n*ENDPTPRIME field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23–16 PETB	Prime Endpoint Transmit Buffer - R/WS. For each endpoint a corresponding bit is used to request that a buffer is prepared for a transmit operation in order to respond to a USB IN/INTERRUPT transaction. Software should write a one to the corresponding bit when posting a new transfer descriptor to an endpoint queue head. Hardware automatically uses this bit to begin parsing for a new transfer descriptor from the queue head and prepare a transmit buffer. Hardware clears this bit when the associated endpoint(s) is (are) successfully primed. **NOTE:** These bits are momentarily set by hardware during hardware re-priming operations when a dTD is retired, and the dQH is updated. PETB[N] - Endpoint #N, N is in 0..7
15–8 -	This field is reserved. Reserved
PERB	Prime Endpoint Receive Buffer - R/WS. For each endpoint, a corresponding bit is used to request a buffer prepare for a receive operation for when a USB host initiates a USB OUT transaction. Software should write a one to the corresponding bit whenever posting a new transfer descriptor to an endpoint queue head. Hardware automatically uses this bit to begin parsing for a new transfer descriptor from the queue head and prepare a receive buffer. Hardware clears this bit when the associated endpoint(s) is (are) successfully primed. **NOTE:** These bits are momentarily set by hardware during hardware re-priming operations when a dTD is retired, and the dQH is updated. PERB[N] - Endpoint #N, N is in 0..7

42.7.36 Endpoint Flush (USB_*n*ENDPTFLUSH)

This register is only used in device mode.

Address: 402E_0000h base + 1B4h offset + (512d × i), where i=0d to 1d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								FETB								Reserved								FERB							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USB_*n*ENDPTFLUSH field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23–16 FETB	Flush Endpoint Transmit Buffer - R/WS. Writing one to a bit(s) in this register causes the associated endpoint(s) to clear any primed buffers. If a packet is in progress for one of the associated endpoints, then that transfer continues until completion. Hardware clears this register after the endpoint flush operation is successful. FETB[N] - Endpoint #N, N is in 0..7
15–8 -	This field is reserved. Reserved
FERB	Flush Endpoint Receive Buffer - R/WS. Writing one to a bit(s) causes the assocUOGiated endpoint(s) to clear any primed buffers. If a packet is in progress for one of the associated endpoints, then that transfer continues until completion. Hardware clears this register after the endpoint flush operation is successful. FERB[N] - Endpoint #N, N is in 0..7\

42.7.37 Endpoint Status (USB_*n*ENDPTSTAT)

This register is only used in device mode.

Address: 402E_0000h base + 1B8h offset + (512d × i), where i=0d to 1d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								ETBR								Reserved								ERBR							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USB_*n*ENDPTSTAT field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23–16 ETBR	Endpoint Transmit Buffer Ready -- Read Only. One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to one by the hardware as a response to receiving a command from a corresponding bit in the ENDPTPRIME register. There is always a delay between setting a bit in the ENDPTPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. **NOTE:** These bits are momentarily cleared by hardware during hardware endpoint re-priming operations when a dTD is retired, and the dQH is updated. ETBR[N] - Endpoint #N, N is in 0..7
15–8 -	This field is reserved. Reserved
ERBR	Endpoint Receive Buffer Ready -- Read Only. One bit for each endpoint indicates status of the respective endpoint buffer. This bit is set to a one by the hardware as a response to receiving a command from a corresponding bit in the ENDPRIME register. There is always a delay between setting a bit in the ENDPRIME register and endpoint indicating ready. This delay time varies based upon the current USB traffic and the number of bits set in the ENDPRIME register. Buffer ready is cleared by USB reset, by the USB DMA system, or through the ENDPTFLUSH register. **NOTE:** These bits are momentarily cleared by hardware during hardware endpoint re-priming operations when a dTD is retired, and the dQH is updated. ERBR[N] - Endpoint #N, N is in 0..7

42.7.38 Endpoint Complete (USB_*n*ENDPTCOMPLETE)

This register is only used in device mode.

Address: 402E_0000h base + 1BCh offset + (512d × i), where i=0d to 1d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								ETCE								Reserved								ERCE							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USB_*n*ENDPTCOMPLETE field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23–16 ETCE	Endpoint Transmit Complete Event - R/WC. Each bit indicates a transmit event (IN/INTERRUPT) occurred and software should read the corresponding endpoint queue to determine the endpoint status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit is set simultaneously with the *USBINT* . Writing one clears the corresponding bit in this register. ETCE[N] - Endpoint #N, N is in 0..7

Table continues on the next page...

USB_*n*ENDPTCOMPLETE field descriptions (continued)

Field	Description
15–8 -	This field is reserved. Reserved
ERCE	Endpoint Receive Complete Event - RW/C. Each bit indicates a received event (OUT/SETUP) occurred and software should read the corresponding endpoint queue to determine the transfer status. If the corresponding IOC bit is set in the Transfer Descriptor, then this bit is set simultaneously with the *USBINT* . Writing one clears the corresponding bit in this register. ERCE[N] - Endpoint #N, N is in 0..7

42.7.39 Endpoint Control0 (USB_*n*ENDPTCTRL0)

Every Device implements Endpoint 0 as a control endpoint.

Address: 402E_0000h base + 1C0h offset + (512d × i), where i=0d to 1d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved								TXE	Reserved			TXT		Reserved	TXS
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								RXE	Reserved			RXT		Reserved	RXS
Reset	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

USB_*n*ENDPTCTRL0 field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23 TXE	TX Endpoint Enable 1 Enabled

Table continues on the next page...

USB_*n*ENDPTCTRL0 field descriptions (continued)

Field	Description
	Endpoint0 is always enabled.
22–20 -	This field is reserved. Reserved
19–18 TXT	TX Endpoint Type - Read/Write 00 - Control Endpoint0 is fixed as a Control End Point.
17 -	This field is reserved. Reserved
16 TXS	TX Endpoint Stall - Read/Write 0 End Point OK [Default] 1 End Point Stalled Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It continues returning STALL until the bit is cleared by software or it is automatically cleared upon receipt of a new SETUP request. After receiving a SETUP request, this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. **NOTE:** There is a slight delay (50 clocks max.) between the endptsetupstat being cleared and hardware continuing to clear this bit. In most systems it is unlikely the DCD software will observe this delay. However, should the dcd observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a newsetup has been received by checking the associated endptsetupstat bit.
15–8 -	This field is reserved. Reserved
7 RXE	RX Endpoint Enable 1 Enabled Endpoint0 is always enabled.
6–4 -	This field is reserved. Reserved
3–2 RXT	RX Endpoint Type - Read/Write 00 Control Endpoint0 is fixed as a Control End Point.
1 -	This field is reserved. Reserved
0 RXS	RX Endpoint Stall - Read/Write 0 End Point OK. [Default] 1 End Point Stalled Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. It continues returning STALL until the bit is cleared by software or it is automatically cleared upon receipt of a new SETUP request. After receiving a SETUP request, this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared.

Table continues on the next page...

USB_*n*ENDPTCTRL0 field descriptions (continued)

Field	Description
	NOTE: There is a slight delay (50 clocks max.) between the endptsetupstat being cleared and hardware continuing to clear this bit. In most systems it is unlikely the dcd software will observe this delay. However, should the dcd observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a newsetup has been received by checking the associated endptsetupstat bit.

42.7.40 Endpoint Control 1 (USB_*n*ENDPTCTRL1)

This is endpoint control register for endpoint 1 in device operation mode.

NOTE

If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled then the unused direction type must be changed from the default control-type to any other type (that is Bulk-type). leaving an unconfigured endpoint control causes undefined behavior for the data pid tracking on the active endpoint/direction.

Address: 402E_0000h base + 1C4h offset + (512d × i), where i=0d to 1d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved								TXE	TXR	TXI	Reserved	TXT		TXD	TXS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								RXE	RXR	RXI	Reserved	RXT		RXD	RXS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USB_*n*ENDPTCTRL1 field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23 TXE	TX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
22 TXR	TX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the Host and device.
21 TXI	TX Data Toggle Inhibit 0 PID Sequencing Enabled. [Default] 1 PID Sequencing Disabled. This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet.
20 -	This field is reserved. Reserved
19–18 TXT	TX Endpoint Type - Read/Write 00 Control 01 Isochronous 10 Bulk 11 Interrupt
17 TXD	TX Endpoint Data Source - Read/Write 0 Dual Port Memory Buffer/DMA Engine [DEFAULT] Should always be written as 0.
16 TXS	TX Endpoint Stall - Read/Write 0 End Point OK 1 End Point Stalled This bit will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint and this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE:** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit.
15–8 -	This field is reserved. Reserved

Table continues on the next page...

USB_*n*ENDPTCTRL1 field descriptions (continued)

Field	Description
7 RXE	RX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
6 RXR	RX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the host and device.
5 RXI	RX Data Toggle Inhibit 0 Disabled [Default] 1 Enabled This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID.
4 -	This field is reserved. Reserved.
3–2 RXT	RX Endpoint Type - Read/Write 00 Control 01 Isochronous 10 Bulk 11 Interrupt
1 RXD	RX Endpoint Data Sink - Read/Write 0 Dual Port Memory Buffer/DMA Engine [Default] Should always be written as zero.
0 RXS	RX Endpoint Stall - Read/Write 0 End Point OK. [Default] 1 End Point Stalled This bit is set automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpointand this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE:** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit.

42.7.41 Endpoint Control 2 (USB_*n*ENDPTCTRL2)

This is endpoint control register for endpoint 2 in device operation mode.

NOTE

If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled then the unused direction type must be changed from the default control-type to any other type (that is Bulk-type). leaving an unconfigured endpoint control causes undefined behavior for the data pid tracking on the active endpoint/direction.

Address: 402E_0000h base + 1C8h offset + (512d × i), where i=0d to 1d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved								TXE	TXR	TXI	Reserved	TXT		TXD	TXS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved								RXE	RXR	RXI	Reserved	RXT		RXD	RXS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USB_*n*ENDPTCTRL2 field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23 TXE	TX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.

Table continues on the next page...

USB_*n*ENDPTCTRL2 field descriptions (continued)

Field	Description
22 TXR	TX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the Host and device.
21 TXI	TX Data Toggle Inhibit 0 PID Sequencing Enabled. [Default] 1 PID Sequencing Disabled. This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet.
20 -	This field is reserved. Reserved
19–18 TXT	TX Endpoint Type - Read/Write 00 Control 01 Isochronous 10 Bulk 11 Interrupt
17 TXD	TX Endpoint Data Source - Read/Write 0 Dual Port Memory Buffer/DMA Engine [DEFAULT] Should always be written as 0.
16 TXS	TX Endpoint Stall - Read/Write 0 End Point OK 1 End Point Stalled This bit will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint and this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE:** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit.
15–8 -	This field is reserved. Reserved
7 RXE	RX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
6 RXR	RX Data Toggle Reset (WS) Write 1 - Reset PID Sequence

Table continues on the next page...

USB_*n*ENDPTCTRL2 field descriptions (continued)

Field	Description
	Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the host and device.
5 RXI	RX Data Toggle Inhibit 0 Disabled [Default] 1 Enabled This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID.
4 -	This field is reserved. Reserved.
3–2 RXT	RX Endpoint Type - Read/Write 00 Control 01 Isochronous 10 Bulk 11 Interrupt
1 RXD	RX Endpoint Data Sink - Read/Write 0 Dual Port Memory Buffer/DMA Engine [Default] Should always be written as zero.
0 RXS	RX Endpoint Stall - Read/Write 0 End Point OK. [Default] 1 End Point Stalled This bit is set automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpointand this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE:** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit.

42.7.42 Endpoint Control 3 (USB_*n*ENDPTCTRL3)

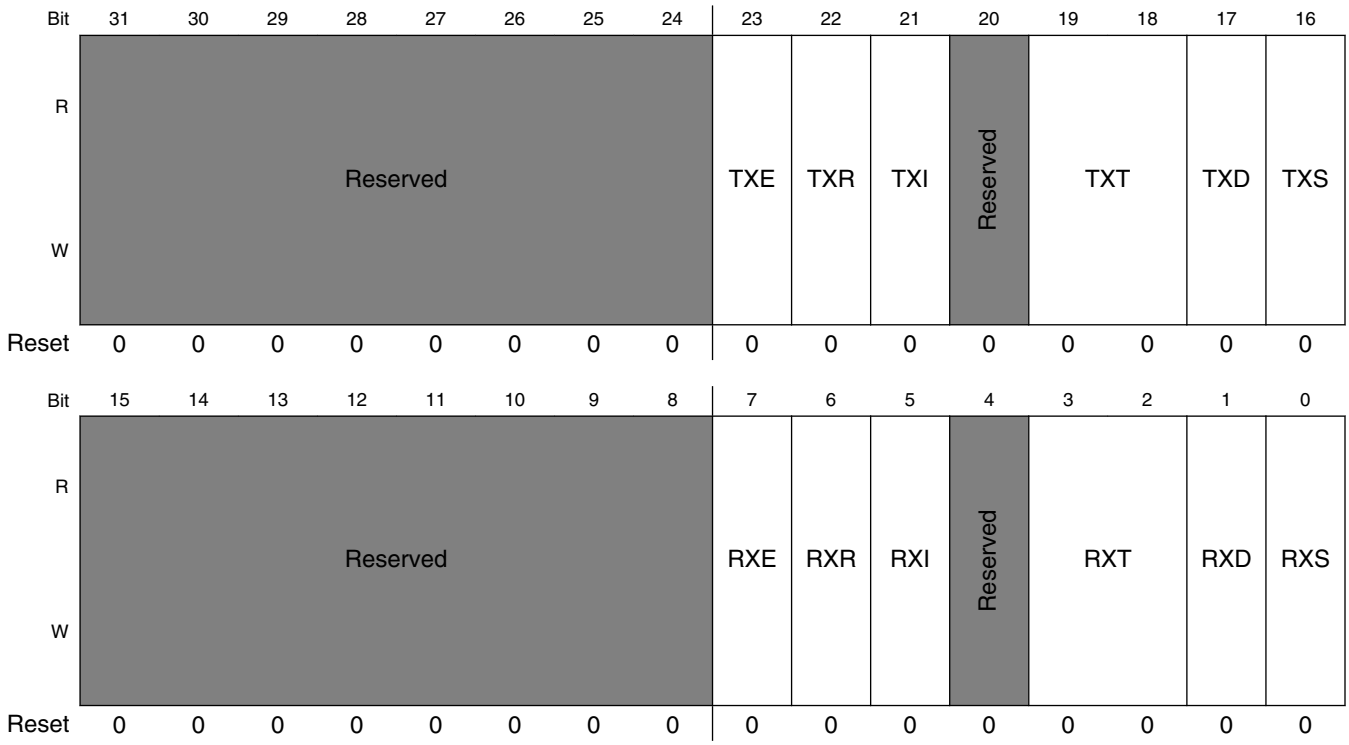
This is endpoint control register for endpoint 3 in device operation mode.

NOTE

If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled then the unused direction type must be changed from the default control-type to any other type (that is Bulk-type). leaving an unconfigured endpoint control

causes undefined behavior for the data pid tracking on the active endpoint/direction.

Address: 402E_0000h base + 1CCh offset + (512d × i), where i=0d to 1d



USB_*n*ENDPTCTRL3 field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23 TXE	TX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
22 TXR	TX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the Host and device.
21 TXI	TX Data Toggle Inhibit 0 PID Sequencing Enabled. [Default] 1 PID Sequencing Disabled. This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet.
20 -	This field is reserved. Reserved

Table continues on the next page...

USB_*n*ENDPTCTRL3 field descriptions (continued)

Field	Description
19–18 TXT	TX Endpoint Type - Read/Write 00 Control 01 Isochronous 10 Bulk 11 Interrupt
17 TXD	TX Endpoint Data Source - Read/Write 0 Dual Port Memory Buffer/DMA Engine [DEFAULT] Should always be written as 0.
16 TXS	TX Endpoint Stall - Read/Write 0 End Point OK 1 End Point Stalled This bit will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint and this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE:** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit.
15–8 -	This field is reserved. Reserved
7 RXE	RX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
6 RXR	RX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the host and device.
5 RXI	RX Data Toggle Inhibit 0 Disabled [Default] 1 Enabled This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID.
4 -	This field is reserved. Reserved.
3–2 RXT	RX Endpoint Type - Read/Write 00 Control

Table continues on the next page...

USB_*n*ENDPTCTRL3 field descriptions (continued)

Field	Description
	01 Isochronous 10 Bulk 11 Interrupt
1 RXD	RX Endpoint Data Sink - Read/Write 0 Dual Port Memory Buffer/DMA Engine [Default] Should always be written as zero.
0 RXS	RX Endpoint Stall - Read/Write 0 End Point OK. [Default] 1 End Point Stalled This bit is set automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpointand this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE:** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit.

42.7.43 Endpoint Control 4 (USB_*n*ENDPTCTRL4)

This is endpoint control register for endpoint 4 in device operation mode.

NOTE

If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled then the unused direction type must be changed from the default control-type to any other type (that is Bulk-type). leaving an unconfigured endpoint control causes undefined behavior for the data pid tracking on the active endpoint/direction.

Address: 402E_0000h base + 1D0h offset + (512d × i), where i=0d to 1d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved								TXE	TXR	TXI	Reserved	TXT		TXD	TXS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved								RXE	RXR	RXI	Reserved	RXT		RXD	RXS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USB_*n*ENDPTCTRL4 field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23 TXE	TX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
22 TXR	TX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the Host and device.
21 TXI	TX Data Toggle Inhibit 0 PID Sequencing Enabled. [Default] 1 PID Sequencing Disabled. This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet.
20 -	This field is reserved. Reserved
19–18 TXT	TX Endpoint Type - Read/Write 00 Control 01 Isochronous

Table continues on the next page...

USB_*n*ENDPTCTRL4 field descriptions (continued)

Field	Description
	10 Bulk 11 Interrupt
17 TXD	TX Endpoint Data Source - Read/Write 0 Dual Port Memory Buffer/DMA Engine [DEFAULT] Should always be written as 0.
16 TXS	TX Endpoint Stall - Read/Write 0 End Point OK 1 End Point Stalled This bit will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint and this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE:** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit.
15–8 -	This field is reserved. Reserved
7 RXE	RX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
6 RXR	RX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the host and device.
5 RXI	RX Data Toggle Inhibit 0 Disabled [Default] 1 Enabled This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID.
4 -	This field is reserved. Reserved.
3–2 RXT	RX Endpoint Type - Read/Write 00 Control 01 Isochronous 10 Bulk 11 Interrupt

Table continues on the next page...

USB_*n*ENDPTCTRL4 field descriptions (continued)

Field	Description
1 RXD	RX Endpoint Data Sink - Read/Write 0 Dual Port Memory Buffer/DMA Engine [Default] Should always be written as zero.
0 RXS	RX Endpoint Stall - Read/Write 0 End Point OK. [Default] 1 End Point Stalled This bit is set automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpointand this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE:** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit.

42.7.44 Endpoint Control 5 (USB_*n*ENDPTCTRL5)

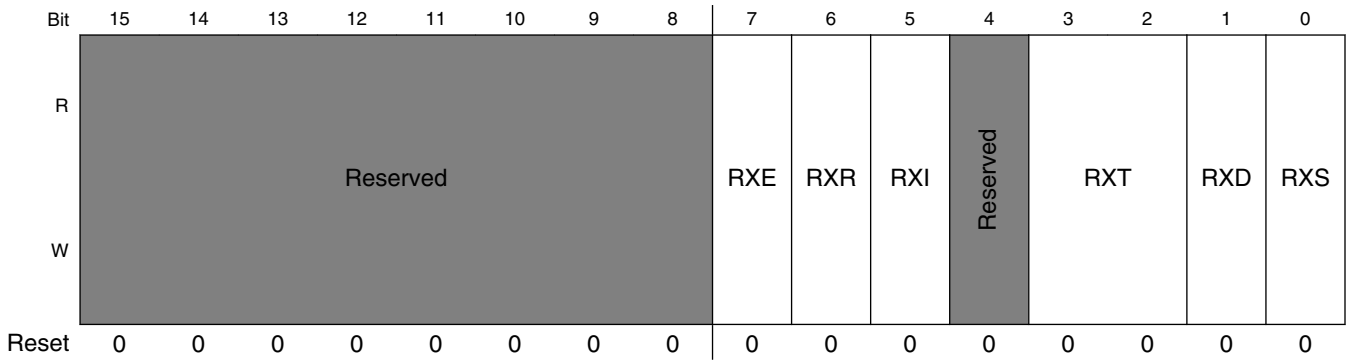
This is endpoint control register for endpoint 5 in device operation mode.

NOTE

If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled then the unused direction type must be changed from the default control-type to any other type (that is Bulk-type). leaving an unconfigured endpoint control causes undefined behavior for the data pid tracking on the active endpoint/direction.

Address: 402E_0000h base + 1D4h offset + (512d × i), where i=0d to 1d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved								TXE	TXR	TXI	Reserved	TXT		TXD	TXS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



USB_*n*ENDPTCTRL5 field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23 TXE	TX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
22 TXR	TX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the Host and device.
21 TXI	TX Data Toggle Inhibit 0 PID Sequencing Enabled. [Default] 1 PID Sequencing Disabled. This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet.
20 -	This field is reserved. Reserved
19–18 TXT	TX Endpoint Type - Read/Write 00 Control 01 Isochronous 10 Bulk 11 Interrupt
17 TXD	TX Endpoint Data Source - Read/Write 0 Dual Port Memory Buffer/DMA Engine [DEFAULT] Should always be written as 0.
16 TXS	TX Endpoint Stall - Read/Write 0 End Point OK 1 End Point Stalled

Table continues on the next page...

USB_*n*ENDPTCTRL5 field descriptions (continued)

Field	Description
	This bit will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint and this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE:** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit.
15–8 -	This field is reserved. Reserved
7 RXE	RX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
6 RXR	RX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the host and device.
5 RXI	RX Data Toggle Inhibit 0 Disabled [Default] 1 Enabled This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID.
4 -	This field is reserved. Reserved.
3–2 RXT	RX Endpoint Type - Read/Write 00 Control 01 Isochronous 10 Bulk 11 Interrupt
1 RXD	RX Endpoint Data Sink - Read/Write 0 Dual Port Memory Buffer/DMA Engine [Default] Should always be written as zero.
0 RXS	RX Endpoint Stall - Read/Write 0 End Point OK. [Default] 1 End Point Stalled

Table continues on the next page...

USB_*n*ENDPTCTRL5 field descriptions (continued)

Field	Description
	This bit is set automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpointand this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE:** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit.

42.7.45 Endpoint Control 6 (USB_*n*ENDPTCTRL6)

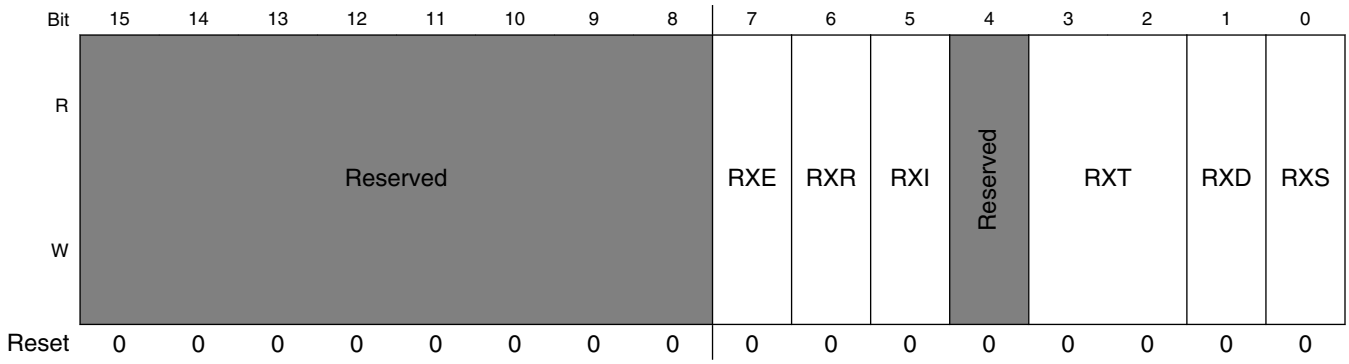
This is endpoint control register for endpoint 6 in device operation mode.

NOTE

If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled then the unused direction type must be changed from the default control-type to any other type (that is Bulk-type). leaving an unconfigured endpoint control causes undefined behavior for the data pid tracking on the active endpoint/direction.

Address: 402E_0000h base + 1D8h offset + (512d × i), where i=0d to 1d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved								TXE	TXR	TXI	Reserved	TXT		TXD	TXS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



USB_*n*ENDPTCTRL6 field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23 TXE	TX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
22 TXR	TX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the Host and device.
21 TXI	TX Data Toggle Inhibit 0 PID Sequencing Enabled. [Default] 1 PID Sequencing Disabled. This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet.
20 -	This field is reserved. Reserved
19–18 TXT	TX Endpoint Type - Read/Write 00 Control 01 Isochronous 10 Bulk 11 Interrupt
17 TXD	TX Endpoint Data Source - Read/Write 0 Dual Port Memory Buffer/DMA Engine [DEFAULT] Should always be written as 0.
16 TXS	TX Endpoint Stall - Read/Write 0 End Point OK 1 End Point Stalled

Table continues on the next page...

USB_*n*ENDPTCTRL6 field descriptions (continued)

Field	Description
	This bit will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint and this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE:** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit.
15–8 -	This field is reserved. Reserved
7 RXE	RX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
6 RXR	RX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the host and device.
5 RXI	RX Data Toggle Inhibit 0 Disabled [Default] 1 Enabled This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID.
4 -	This field is reserved. Reserved.
3–2 RXT	RX Endpoint Type - Read/Write 00 Control 01 Isochronous 10 Bulk 11 Interrupt
1 RXD	RX Endpoint Data Sink - Read/Write 0 Dual Port Memory Buffer/DMA Engine [Default] Should always be written as zero.
0 RXS	RX Endpoint Stall - Read/Write 0 End Point OK. [Default] 1 End Point Stalled

Table continues on the next page...

USB_*n*ENDPTCTRL6 field descriptions (continued)

Field	Description
	This bit is set automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpointand this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE:** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit.

42.7.46 Endpoint Control 7 (USB_*n*ENDPTCTRL7)

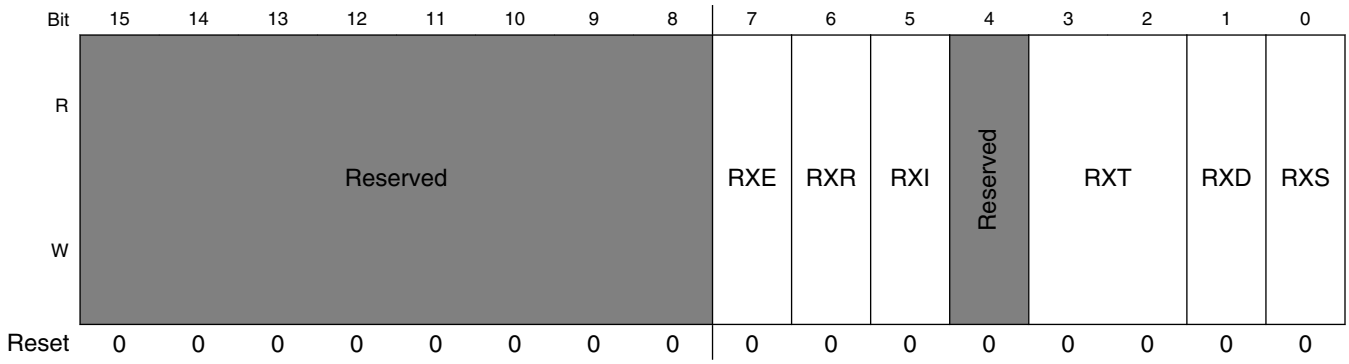
This is endpoint control register for endpoint 7 in device operation mode.

NOTE

If one endpoint direction is enabled and the paired endpoint of opposite direction is disabled then the unused direction type must be changed from the default control-type to any other type (that is Bulk-type). leaving an unconfigured endpoint control causes undefined behavior for the data pid tracking on the active endpoint/direction.

Address: 402E_0000h base + 1DCh offset + (512d × i), where i=0d to 1d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved								TXE	TXR	TXI	Reserved	TXT		TXD	TXS
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



USB_*n*ENDPTCTRL7 field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23 TXE	TX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
22 TXR	TX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the Host and device.
21 TXI	TX Data Toggle Inhibit 0 PID Sequencing Enabled. [Default] 1 PID Sequencing Disabled. This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always transmit DATA0 for a data packet.
20 -	This field is reserved. Reserved
19–18 TXT	TX Endpoint Type - Read/Write 00 Control 01 Isochronous 10 Bulk 11 Interrupt
17 TXD	TX Endpoint Data Source - Read/Write 0 Dual Port Memory Buffer/DMA Engine [DEFAULT] Should always be written as 0.
16 TXS	TX Endpoint Stall - Read/Write 0 End Point OK 1 End Point Stalled

Table continues on the next page...

USB_*n*ENDPTCTRL7 field descriptions (continued)

Field	Description
	This bit will be cleared automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpoint and this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE:** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit.
15–8 -	This field is reserved. Reserved
7 RXE	RX Endpoint Enable 0 Disabled [Default] 1 Enabled An Endpoint should be enabled only after it has been configured.
6 RXR	RX Data Toggle Reset (WS) Write 1 - Reset PID Sequence Whenever a configuration event is received for this Endpoint, software must write a one to this bit in order to synchronize the data PID's between the host and device.
5 RXI	RX Data Toggle Inhibit 0 Disabled [Default] 1 Enabled This bit is only used for test and should always be written as zero. Writing a one to this bit causes this endpoint to ignore the data toggle sequence and always accept data packet regardless of their data PID.
4 -	This field is reserved. Reserved.
3–2 RXT	RX Endpoint Type - Read/Write 00 Control 01 Isochronous 10 Bulk 11 Interrupt
1 RXD	RX Endpoint Data Sink - Read/Write 0 Dual Port Memory Buffer/DMA Engine [Default] Should always be written as zero.
0 RXS	RX Endpoint Stall - Read/Write 0 End Point OK. [Default] 1 End Point Stalled

Table continues on the next page...

USB_*n*ENDPTCTRL7 field descriptions (continued)

Field	Description
	This bit is set automatically upon receipt of a SETUP request if this Endpoint is configured as a Control Endpointand this bit will continue to be cleared by hardware until the associated ENDPTSETUPSTAT bit is cleared. Software can write a one to this bit to force the endpoint to return a STALL handshake to the Host. This control will continue to STALL until this bit is either cleared by software or automatically cleared as above for control endpoints. **NOTE:** [CONTROL ENDPOINT TYPES ONLY]: there is a slight delay (50 clocks max) between the ENDPTSETUPSTAT begin cleared and hardware continuing to clear this bit. In most systems, it is unlikely the DCD software will observe this delay. However, should the DCD observe that the stall bit is not set after writing a one to it then follow this procedure: continually write this stall bit until it is set or until a new setup has been received by checking the associated endptsetupstat bit.

Chapter 43
Universal Serial Bus 2.0 Integrated PHY (USB-PHY)

43.1 Chip-specific USB-PHY information

Table 43-1. Reference links to related information

Topic	Related module(s)	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
Interrupts, DMA Events and XBAR Assignments	-	Interrupts, DMA Events and XBAR Assignments

43.2 USB PHY Overview

The chip contains 2 integrated USB 2.0 PHY macrocells capable of connecting to USB host/device systems at the USB low-speed (LS) rate of 1.5 Mbits/s, full-speed (FS) rate of 12 Mbits/s or at the USB 2.0 high-speed (HS) rate of 480 Mbits/s.

The integrated PHY provides a standard UTM interface. The USB_*n*_DN and USB_*n*_DP pins connect directly to a USB connector.

The following subsections describe the external interfaces, internal interfaces, major blocks, and programmable registers that comprise the integrated USB 2.0 PHY.

43.3 Operation

The UTM provides a 16-bit interface to the USB controller. This interface is clocked at 30 MHz.

- The digital portions of the USBPHY block include the UTMI, digital transmitter, digital receiver, and the programmable registers.
- The analog transceiver section comprises an analog receiver and an analog transmitter, as shown in Figure 43-1.

43.3.1 UTMI

The UTMI block handles the line_state bits, reset buffering, suspend distribution, transceiver speed selection, and transceiver termination selection.

The PLL supplies a 120 MHz signal to all of the digital logic. The UTMI block does a final divide-by-four to develop the 30 MHz clock used in the interface.

43.3.2 Digital Transmitter

The digital transmitter receives the 16-bit transmit data from the USB controller and handles the tx_valid, tx_validh and tx_ready handshake.

In addition, it contains the transmit serializer that converts the 16-bit parallel words at 30 MHz to a single bitstream at 480 Mbit for high-speed or 12 Mbit for full-speed or 1.5 Mbit for low-speed. It does this while implementing the bit-stuffing algorithm and the NRZI encoder that are used to remove the DC component from the serial bitstream. The output of this encoder is sent to the low-speed (LS), full-speed (FS) or high-speed (HS) drivers in the analog transceiver section's transmitter block.

43.3.3 Digital Receiver

The digital receiver receives the raw serial bitstream from the low speed (LS) differential transceiver, full speed (FS) differential transceiver, and a 9X, 480 MHz sampled data from the high speed (HS) differential transceiver.

As the phase of the USB host transmitter shifts relative to the local PLL, the receiver section's HS DLL tracks these changes to give a reliable sample of the incoming 480 Mbit/s bitstream. Since this sample point shifts relative to the PLL phase used by the digital logic, a rate-matching elastic buffer is provided to cross this clock domain

boundary. Once the bitstream is in the local clock domain, an NRZI decoder and bit unstuffer restore the original payload data bitstream and pass it to a deserializer and holding register. The receive state machine handles the rx_valid, rx_validh, and handshake with the USB controller. The handshake is not interlocked, in that there is no rx_ready signal coming from the controller. The controller must take each 16-bit value as presented by the PHY. The receive state machine provides an rx_active signal to the controller that indicates when it is inside a valid packet (SYNC detected, and so on).

43.3.4 Analog Receiver

The analog receiver comprises five differential receivers, two single-ended receivers, and a 9X, 480 MHz HS data sampling module

, as shown in the figure below and described further in this section.

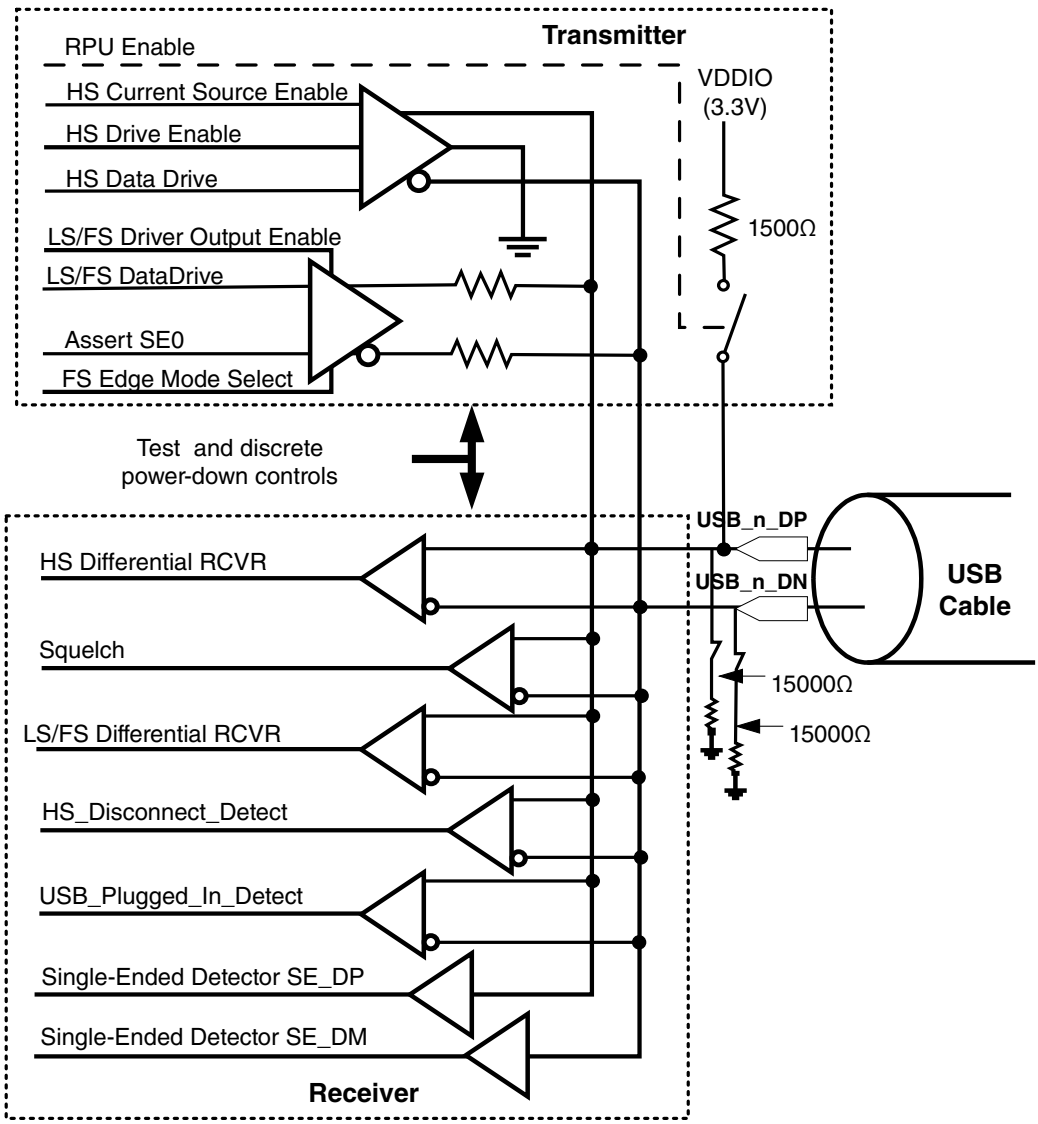


Figure 43-1. USB 2.0 PHY Analog Transceiver Block Diagram

43.3.4.1 HS Differential Receiver

The high-speed differential receiver is both a differential analog receiver and threshold comparator. Its output is a one if the differential signal is greater than a 0-V threshold.

Otherwise, its output is 0. Its purpose is to discriminate the ± 400-mV differential voltage resulting from the high-speed drivers current flow into the dual 45Ω terminations found on each pin of the differential pair. The envelope or squelch detector, described below, ensures that the differential signal has sufficient magnitude to be valid. The HS differential receiver tolerates up to 500 mV of common mode offset.

43.3.4.2 Squelch Detector

The squelch detector is a differential analog receiver and threshold comparator.

Its output is 1, if the differential magnitude is less than a nominal 100 mV threshold. Otherwise, its output is 0.

Its purpose is to invalidate the HS differential receiver when the incoming signal is simply too low to receive reliably.

43.3.4.3 LS/FS Differential Receiver

The low-speed/full-speed differential receiver is both a differential analog receiver and threshold comparator.

The crossover voltage falls between 1.3 V and 2.0 V. Its output is 1, when the USB_*n*_DP line is above the crossover point and the USB_*n*_DN line is below the crossover point. The digital receiver section decodes the receiver data into J or K state according to the speed.

43.3.4.4 HS Disconnect Detector

It is a differential analog receiver and threshold comparator. It outputs high when differential magnitude is greater than a nominal 575-mV threshold. Otherwise, it outputs low.

43.3.4.5 USB Plugged-In Detector

The USB plugged-in detector looks for both USB_*n*_DP and USB_*n*_DN to be high. There is a pair of large on-chip pullup resistors (200 KΩ) that hold both USB_*n*_DP and USB_*n*_DN high when the USB cable is not attached. The USB plugged-in detector signals a 0 in this case.

When operating in device mode, the upstream port in host/hub interface contains a 15 KΩ pulldown resistor which could easily override the 200 KΩ pullup resistor. When plugged in, at least one signal in the pair will be low, which will force the plugged-in detector's output high.

43.3.4.6 Single-Ended USB_DP Receiver

The single-ended USB_*n*_DP receiver output is high whenever the USB_*n*_DP input is above its nominal 1.8 V threshold.

43.3.4.7 Single-Ended USB_DN Receiver

The single-ended USB_*n*_DN receiver output is high whenever the USB_*n*_DN input is above its nominal 1.8 V threshold.

43.3.4.8 9X Oversample Module

The 9X oversample module uses nine identically spaced phases of the 480 MHz clock to sample a high speed bit data. The squelch signal is sampled only 1X.

43.3.5 Analog Transmitter

The analog transmitter comprises two differential drivers: one for high-speed signaling and one for full-speed signaling. It also contains the switchable 1.5 KΩ pullup resistor.

See Figure 43-1.

43.3.5.1 Switchable High-Speed 45Ω Termination Resistors

High-speed current mode differential signaling requires good 90 Ω differential termination at each end of the USB cable. This results from switching in 45 Ω terminating resistors from each signal line to ground at each end of the cable.

Because each signal is parallel terminated with 45 Ω at each end, each driver sees a 22.5 Ω load. This load impedance is much too low for full-speed signaling levels—hence the need for switchable high-speed terminating resistors. Switchable trimming resistors are provided to tune the actual termination resistance of each device, as shown in Figure

43-2. The HW_USBPHY_TX_TXCAL45DP bit field, for example, allows one of 16 trimming resistor values to be placed in parallel with the 45Ω terminator on the USB_*n*_DP signal.

43.3.5.2 Low-Speed/Full-Speed Differential Driver

The low-speed/full-speed differential drivers are essentially low-impedance pulldown devices that are switched in a differential mode for low-speed or full-speed signaling, that is, either one or the other device is turned on to signal the "J" state or the "K" state.

43.3.5.3 High-Speed Differential Driver

The high-speed differential driver receives a 17.78 mA current from the constant current source (Iref) and essentially steers it down either the USB_DP signal or the USB_DN signal or alternatively to ground.

This current will produce approximately a 400 mV drop across the 22.5 Ω termination seen by the driver when it is steered onto one of the signal lines. The approximately 17.78 mA current source is referenced back to the integrated voltage-band-gap (Vbg) circuit. The Iref, Ibias, and V to I circuits are shared with the integrated battery charger.

43.3.5.4 Switchable 1.5KΩ USB_DP Pullup Resistor

This product contains a switchable 1.5 KΩ pullup resistor on the USB_*n*_DP signal.

This resistor is switched on to indicate to the host/hub controller that a full-speed-capable device is on the USB cable, powered on, and ready. This resistor is switched off at power-on reset so the host does not recognize a USB device until the processor software enables the announcement of a full-speed device.

43.3.5.5 Switchable 15KΩ USB_DP Pulldown Resistor

This product contains a switchable 15 KΩ pulldown resistor on both USB_*n*_DP and USB_*n*_DN signals. This is used in host mode to indicate to the device controller that a host is present.

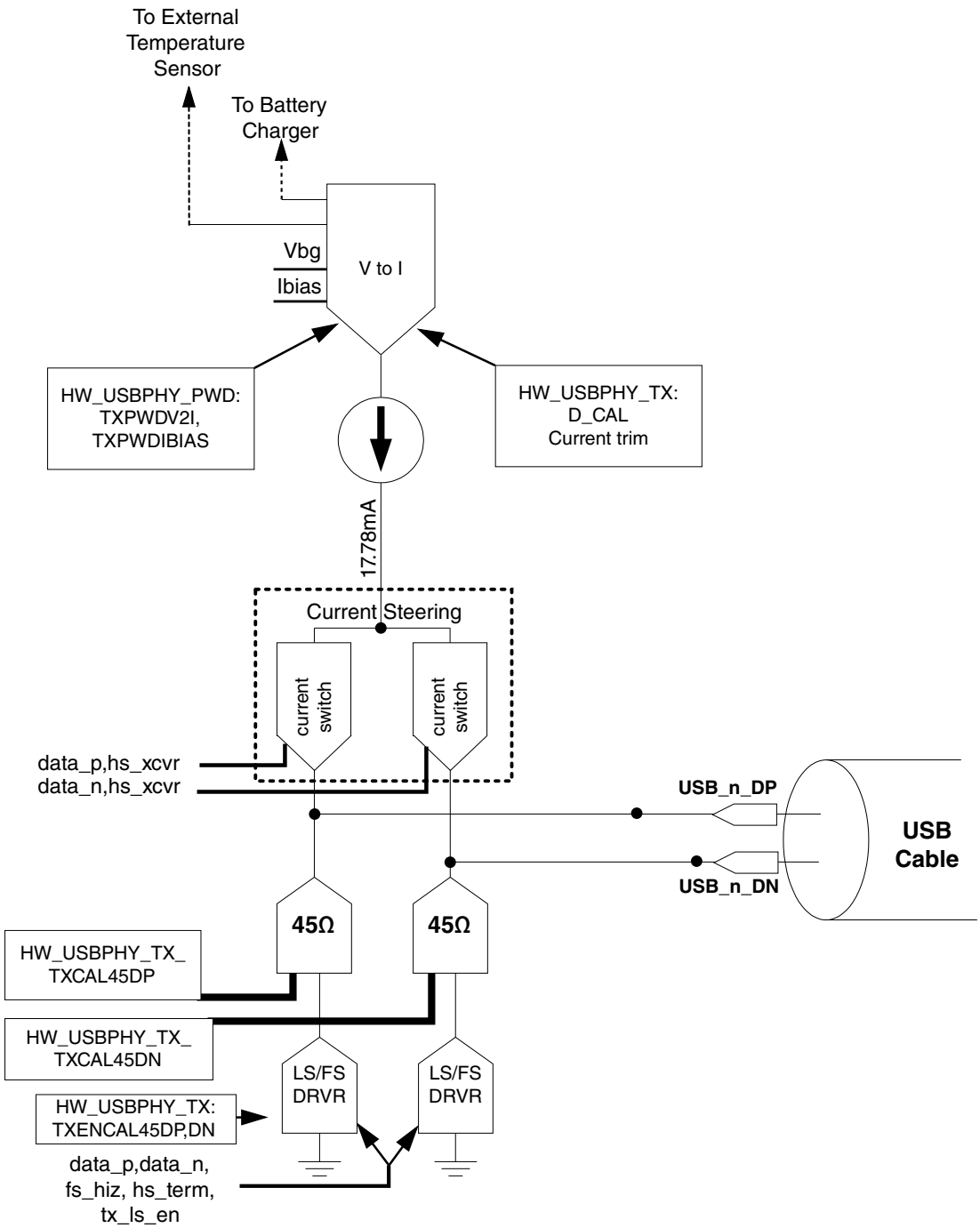


Figure 43-2. USB 2.0 PHY Transmitter Block Diagram

43.3.6 Recommended Register Configuration for USB Certification

The register settings in this section are recommended for passing USB certification.

The following settings lower the J/K levels to certifiable limits:

HW_USBPHY_TX_TXCAL45DP = 2
HW_USBPHY_TX_TXCAL45DN = 2
HW_USBPHY_TX_D_CAL = 8

43.3.7 Charger detection

The USB charger detector is a block that detects whether the upstream-facing device is connected to a down-stream facing charger, either a dedicated USB charger or a host charger.

The USB charger detector is comprised of two sub-blocks, namely the USB data-pin contact detector and the charger detector.

This section details those two sub-blocks and gives the software flow of USB charger detection. Finally, this chapter discusses the detection of a USB charger in case of a dead battery.

43.3.7.1 Charger detect control table

Before we dive into the details of the detectors, we show the logic table of the control signals to give the user an overall picture of the charger detector.

Table 43-2. Charger detection control table

EN_B	CHK_CHRG_B	CHK_CONTACT	Data pin contact detector	Charger detector
0	1	1	Enabled	Disabled
0	0	x(don't care)	Disabled	Enabled
1	x	x(don't care)	Disabled	Disabled

43.3.7.2 Data pin contact detector

According to Battery Charging Specification (rev 1.2), USB plugs and receptacles are designed such that when the plug is inserted into the receptacle, the power pins make contact before the data pins make contact. Therefore, there is inevitably a time interval during which USB_*n*_VBUS has been observed by the device while the USB_*n*_DP and USB_*n*_DN pins are not still pending for contact. The USB data pin contact detector is designed to give the software an indication of the contact of the data pins.

To enable the USB data pin contact detector, the user should set the CHK_CONTACT bit of the USB1_CHRG_DETECT register to 1 and monitor the PLUG_CONTACT bit status of the USB1_CHRG_DETECT_STAT register. If PLUG_CONTACT is 1, then it indicates that the data pins have make good contacts, otherwise the user should continue to wait until this bit is set.

According to Table 1, it should be noted that the data pin contact detector only works when EN_B=0 and CHK_CHRG_B=1, both bit being of the USB1_CHRG_DETECT register.

43.3.7.3 Charger detector

Once the data pins make contact, the user should enable the charger detector by clearing the CHK_CHRG_B bit that is low-active. Then the user should wait for 40ms and then check the status bit of CHRG_DETECTED in register hw_anadig_usb1_chrg_det_stat. CHRG_DETECTED=1 means that the device is connected to a charger, either a dedicated charger or a charging downstream port (or equivalently called a host charger, or charging host). To further differentiate between a host charger and a dedicated charger, the user is suggested to pull up USB_*n*_DP signaling a connect event to the host. Then the user should monitor the USB_*n*_DN line status. If USB_*n*_DN=1, then the charger is a dedicated charger; if USB_*n*_DN=0, then it is a host charger.

43.3.7.4 Charger detection software flow

Upon seeing VBUS, the software should follow the software flow for the charger detection process. The flow chart mentions the "enable the vdd3p0 current limiter". Please refer to the power chapter for details.

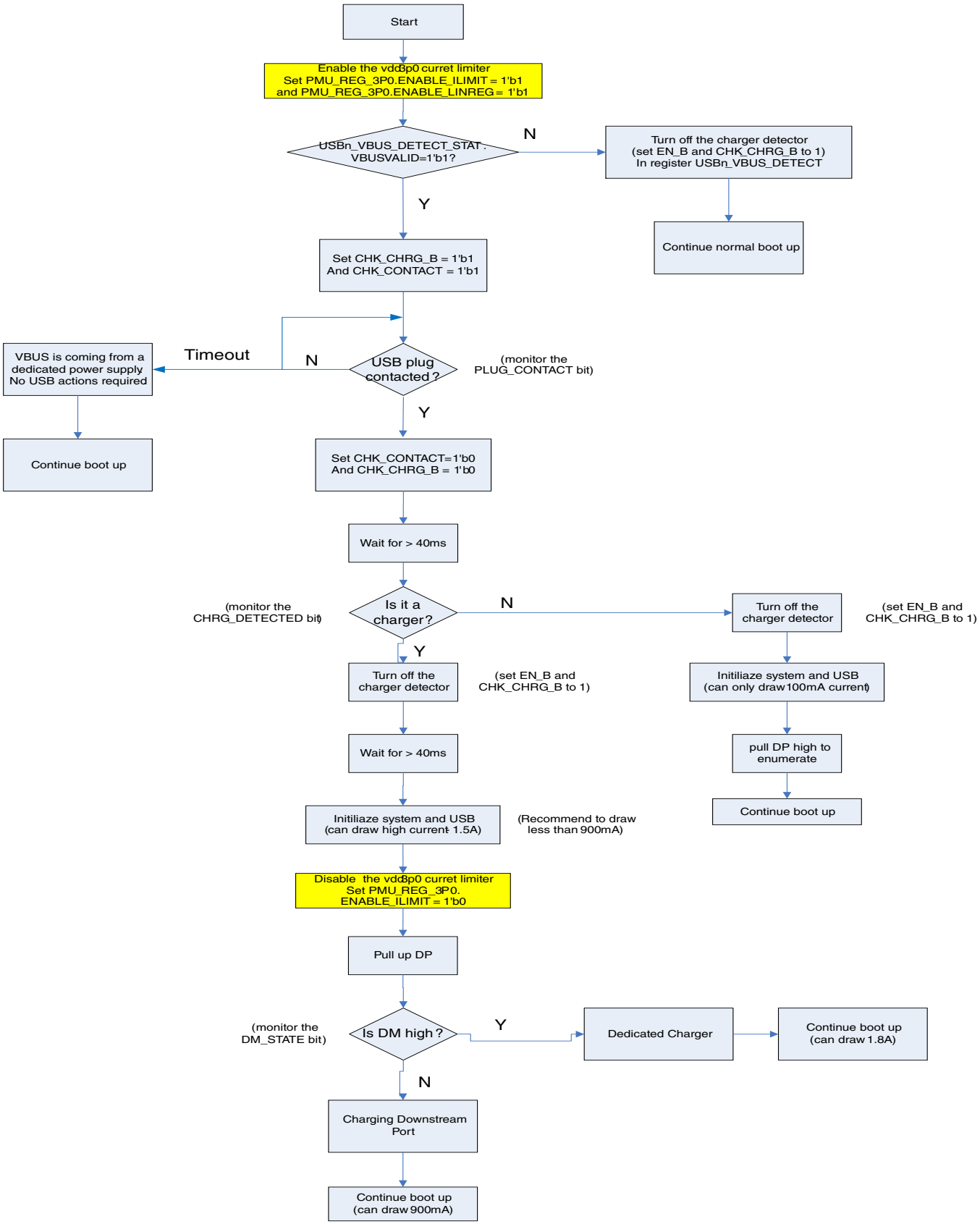


Figure 43-3. USBPHY Charger Detection Software Flow

43.3.7.5 Dead Battery Protect

All the descriptions above are based on the assumption that all the power supplies have been on when the device is plugged into a remote host (or charger). However, there are cases when the local battery of the portable device has been so depleted that the system could not be turned on. In such scenarios the user may prefer a method of signaling the external power management unit (PMIC) the existence of the USB charger to draw a current larger than 100mA from the remote host to speed up system boot up or battery charging. The charger detector indeed supports this function.

When we have a fully depleted battery, all the power supplies might be off. Upon insertion of the 5V, the supplies are brought up by the external PMIC and the internal regulators. Due to the 100mA inrush current limit of the USB spec, we cannot draw larger than 100mA current which might be a limit for system boot-up. Since by default, EN_B=0, CHK_CHRG_B=0 and CHK_CONTACT=1, the usb charger detector is automatically enabled without any software operation needed and it can signal the external PMIC the existence of a USB charger through the open-drain output pin USB_OTG_CHD_B. This pin should be pulled up to an external voltage that is acceptable to the PMIC. If this signal is low, then the PMIC can get that the device is connected to a charger. In this case, the PMIC can draw more than 100mA current from the USB.

It should be noted that this function requires cooperation between the chip and the external PMIC. It is suggested that the user consult NXP for such use cases.

43.4 USB PHY Memory Map/Register Definition

USBPHY Hardware Register Format Summary

USBPHY memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400D_9000	USB PHY Power-Down Register (USBPHY1_PWD)	32	R/W	001E_1C00h	43.4.1/2525
400D_9004	USB PHY Power-Down Register (USBPHY1_PWD_SET)	32	R/W	001E_1C00h	43.4.1/2525
400D_9008	USB PHY Power-Down Register (USBPHY1_PWD_CLR)	32	R/W	001E_1C00h	43.4.1/2525
400D_900C	USB PHY Power-Down Register (USBPHY1_PWD_TOG)	32	R/W	001E_1C00h	43.4.1/2525
400D_9010	USB PHY Transmitter Control Register (USBPHY1_TX)	32	R/W	1006_0607h	43.4.2/2527

Table continues on the next page...

USBPHY memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400D_9014	USB PHY Transmitter Control Register (USBPHY1_TX_SET)	32	R/W	1006_0607h	43.4.2/2527
400D_9018	USB PHY Transmitter Control Register (USBPHY1_TX_CLR)	32	R/W	1006_0607h	43.4.2/2527
400D_901C	USB PHY Transmitter Control Register (USBPHY1_TX_TOG)	32	R/W	1006_0607h	43.4.2/2527
400D_9020	USB PHY Receiver Control Register (USBPHY1_RX)	32	R/W	0000_0000h	43.4.3/2528
400D_9024	USB PHY Receiver Control Register (USBPHY1_RX_SET)	32	R/W	0000_0000h	43.4.3/2528
400D_9028	USB PHY Receiver Control Register (USBPHY1_RX_CLR)	32	R/W	0000_0000h	43.4.3/2528
400D_902C	USB PHY Receiver Control Register (USBPHY1_RX_TOG)	32	R/W	0000_0000h	43.4.3/2528
400D_9030	USB PHY General Control Register (USBPHY1_CTRL)	32	R/W	C020_0000h	43.4.4/2530
400D_9034	USB PHY General Control Register (USBPHY1_CTRL_SET)	32	R/W	C020_0000h	43.4.4/2530
400D_9038	USB PHY General Control Register (USBPHY1_CTRL_CLR)	32	R/W	C020_0000h	43.4.4/2530
400D_903C	USB PHY General Control Register (USBPHY1_CTRL_TOG)	32	R/W	C020_0000h	43.4.4/2530
400D_9040	USB PHY Status Register (USBPHY1_STATUS)	32	R/W	0000_0000h	43.4.5/2533
400D_9050	USB PHY Debug Register (USBPHY1_DEBUG)	32	R/W	7F18_0000h	43.4.6/2535
400D_9054	USB PHY Debug Register (USBPHY1_DEBUG_SET)	32	R/W	7F18_0000h	43.4.6/2535
400D_9058	USB PHY Debug Register (USBPHY1_DEBUG_CLR)	32	R/W	7F18_0000h	43.4.6/2535
400D_905C	USB PHY Debug Register (USBPHY1_DEBUG_TOG)	32	R/W	7F18_0000h	43.4.6/2535
400D_9060	UTMI Debug Status Register 0 (USBPHY1_DEBUG0_STATUS)	32	R	0000_0000h	43.4.7/2537
400D_9070	UTMI Debug Status Register 1 (USBPHY1_DEBUG1)	32	R/W	0000_1000h	43.4.8/2538
400D_9074	UTMI Debug Status Register 1 (USBPHY1_DEBUG1_SET)	32	R/W	0000_1000h	43.4.8/2538
400D_9078	UTMI Debug Status Register 1 (USBPHY1_DEBUG1_CLR)	32	R/W	0000_1000h	43.4.8/2538
400D_907C	UTMI Debug Status Register 1 (USBPHY1_DEBUG1_TOG)	32	R/W	0000_1000h	43.4.8/2538
400D_9080	UTMI RTL Version (USBPHY1_VERSION)	32	R	0403_0000h	43.4.9/2539
400D_A000	USB PHY Power-Down Register (USBPHY2_PWD)	32	R/W	001E_1C00h	43.4.1/2525
400D_A004	USB PHY Power-Down Register (USBPHY2_PWD_SET)	32	R/W	001E_1C00h	43.4.1/2525
400D_A008	USB PHY Power-Down Register (USBPHY2_PWD_CLR)	32	R/W	001E_1C00h	43.4.1/2525
400D_A00C	USB PHY Power-Down Register (USBPHY2_PWD_TOG)	32	R/W	001E_1C00h	43.4.1/2525
400D_A010	USB PHY Transmitter Control Register (USBPHY2_TX)	32	R/W	1006_0607h	43.4.2/2527
400D_A014	USB PHY Transmitter Control Register (USBPHY2_TX_SET)	32	R/W	1006_0607h	43.4.2/2527
400D_A018	USB PHY Transmitter Control Register (USBPHY2_TX_CLR)	32	R/W	1006_0607h	43.4.2/2527
400D_A01C	USB PHY Transmitter Control Register (USBPHY2_TX_TOG)	32	R/W	1006_0607h	43.4.2/2527
400D_A020	USB PHY Receiver Control Register (USBPHY2_RX)	32	R/W	0000_0000h	43.4.3/2528

Table continues on the next page...

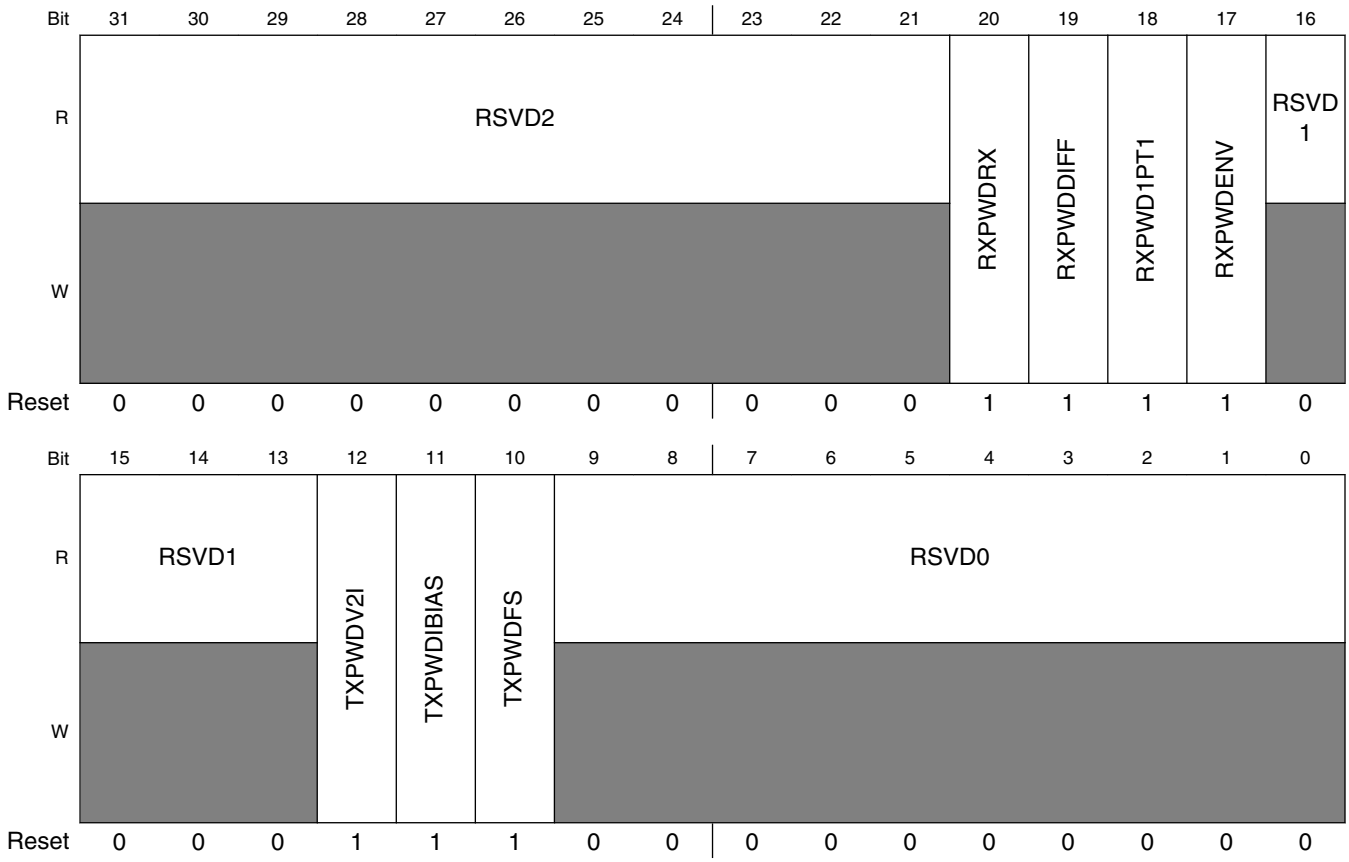
USBPHY memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400D_A024	USB PHY Receiver Control Register (USBPHY2_RX_SET)	32	R/W	0000_0000h	43.4.3/2528
400D_A028	USB PHY Receiver Control Register (USBPHY2_RX_CLR)	32	R/W	0000_0000h	43.4.3/2528
400D_A02C	USB PHY Receiver Control Register (USBPHY2_RX_TOG)	32	R/W	0000_0000h	43.4.3/2528
400D_A030	USB PHY General Control Register (USBPHY2_CTRL)	32	R/W	C020_0000h	43.4.4/2530
400D_A034	USB PHY General Control Register (USBPHY2_CTRL_SET)	32	R/W	C020_0000h	43.4.4/2530
400D_A038	USB PHY General Control Register (USBPHY2_CTRL_CLR)	32	R/W	C020_0000h	43.4.4/2530
400D_A03C	USB PHY General Control Register (USBPHY2_CTRL_TOG)	32	R/W	C020_0000h	43.4.4/2530
400D_A040	USB PHY Status Register (USBPHY2_STATUS)	32	R/W	0000_0000h	43.4.5/2533
400D_A050	USB PHY Debug Register (USBPHY2_DEBUG)	32	R/W	7F18_0000h	43.4.6/2535
400D_A054	USB PHY Debug Register (USBPHY2_DEBUG_SET)	32	R/W	7F18_0000h	43.4.6/2535
400D_A058	USB PHY Debug Register (USBPHY2_DEBUG_CLR)	32	R/W	7F18_0000h	43.4.6/2535
400D_A05C	USB PHY Debug Register (USBPHY2_DEBUG_TOG)	32	R/W	7F18_0000h	43.4.6/2535
400D_A060	UTMI Debug Status Register 0 (USBPHY2_DEBUG0_STATUS)	32	R	0000_0000h	43.4.7/2537
400D_A070	UTMI Debug Status Register 1 (USBPHY2_DEBUG1)	32	R/W	0000_1000h	43.4.8/2538
400D_A074	UTMI Debug Status Register 1 (USBPHY2_DEBUG1_SET)	32	R/W	0000_1000h	43.4.8/2538
400D_A078	UTMI Debug Status Register 1 (USBPHY2_DEBUG1_CLR)	32	R/W	0000_1000h	43.4.8/2538
400D_A07C	UTMI Debug Status Register 1 (USBPHY2_DEBUG1_TOG)	32	R/W	0000_1000h	43.4.8/2538
400D_A080	UTMI RTL Version (USBPHY2_VERSION)	32	R	0403_0000h	43.4.9/2539

43.4.1 USB PHY Power-Down Register (USBPHYx_PWD*n*)

The USB PHY Power-Down Register provides overall control of the PHY power state. Before programming this register, the PHY clocks must be enabled in registers USBPHYx_CTRLn and CCM_ANALOG_USBPHYx_PLL_480_CTRLn.

Address: Base address + 0h offset + (4d × i), where i=0d to 3d



USBPHYx_PWD*n* field descriptions

Field	Description
31–21 RSVD2	Reserved.
20 RXPWDRX	0 = Normal operation. 1 = Power-down the entire USB PHY receiver block except for the full-speed differential receiver. Note that this bit will be auto cleared if there is USB wakeup event while ENAUTOCLR_PHY_PWD bit of USBPHYx_CTRL is enabled.
19 RXPWDDIFF	0 = Normal operation. 1 = Power-down the USB high-speed differential receiver.

Table continues on the next page...

USBPHYx_PWD*n* field descriptions (continued)

Field	Description
	Note that this bit will be auto cleared if there is USB wakeup event while ENAUTOCLR_PHY_PWD bit of USBPHYx_CTRL is enabled.
18 RXPWD1PT1	0 = Normal operation. 1 = Power-down the USB full-speed differential receiver. Note that this bit will be auto cleared if there is USB wakeup event while ENAUTOCLR_PHY_PWD bit of USBPHYx_CTRL is enabled.
17 RXPWDENV	0 = Normal operation. 1 = Power-down the USB high-speed receiver envelope detector (squelch signal). Note that this bit will be auto cleared if there is USB wakeup event while ENAUTOCLR_PHY_PWD bit of USBPHYx_CTRL is enabled.
16–13 RSVD1	Reserved.
12 TXPWDV2I	0 = Normal operation. 1 = Power-down the USB PHY transmit V-to-I converter and the current mirror. Note that this bit will be auto cleared if there is USB wakeup event while ENAUTOCLR_PHY_PWD bit of USBPHYx_CTRL is enabled. Note that these circuits are shared with the battery charge circuit. Setting this to 1 does not power-down these circuits, unless the corresponding bit in the battery charger is also set for power-down.
11 TXPWDIBIAS	0 = Normal operation. 1 = Power-down the USB PHY current bias block for the transmitter. This bit should be set only when the USB is in suspend mode. This effectively powers down the entire USB transmit path. Note that this bit will be auto cleared if there is USB wakeup event while ENAUTOCLR_PHY_PWD bit of USBPHYx_CTRL is enabled. Note that these circuits are shared with the battery charge circuit. Setting this bit to 1 does not power-down these circuits, unless the corresponding bit in the battery charger is also set for power-down.
10 TXPWDFS	0 = Normal operation. 1 = Power-down the USB full-speed drivers. This turns off the current starvation sources and puts the drivers into high-impedance output. Note that this bit will be auto cleared if there is USB wakeup event while ENAUTOCLR_PHY_PWD bit of USBPHYx_CTRL is enabled.
RSVD0	Reserved.

43.4.2 USB PHY Transmitter Control Register (USBPHY*x*_TX*n*)

The USB PHY Transmitter Control Register handles the transmit controls.

Address: Base address + 10h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD5			USBPHY_TX_ EDGECTRL			RSVD2						TXCAL45DP			
W																
Reset	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RSVD1				TXCAL45DN				RSVD0				D_CAL			
W																
Reset	0	0	0	0	0	1	1	0	0	0	0	0	0	1	1	1

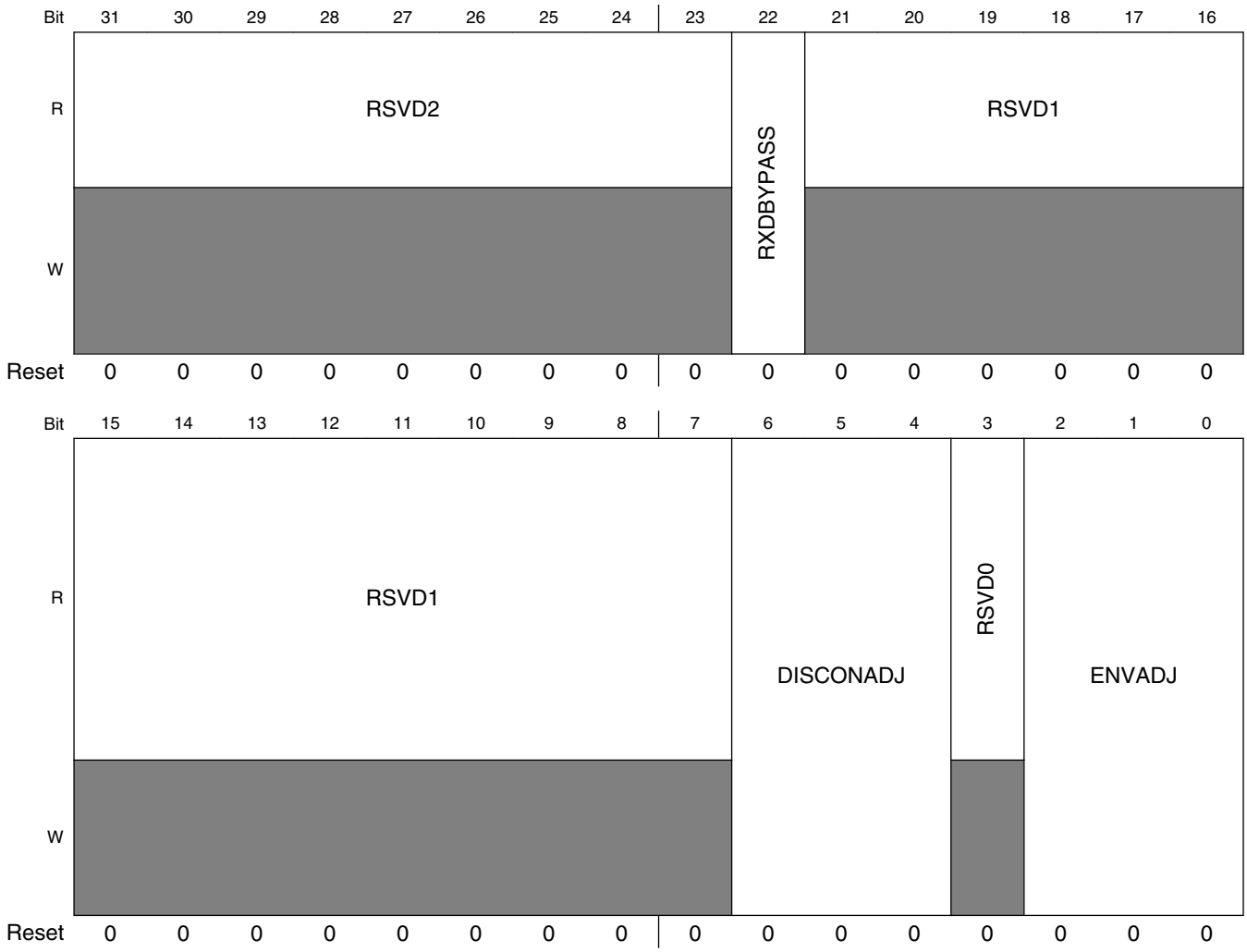
USBPHY*x*_TX*n* field descriptions

Field	Description
31–29 RSVD5	Reserved.
28–26 USBPHY_TX_ EDGECTRL	Controls the edge-rate of the current sensing transistors used in HS transmit. NOT FOR CUSTOMER USE.
25–20 RSVD2	Reserved.
19–16 TXCAL45DP	Decode to select a 45-Ohm resistance to the USB_DP output pin. Maximum resistance = 0000. Resistance is centered by design at 0110.
15–12 RSVD1	Reserved. **Note:** This bit should remain clear.
11–8 TXCAL45DN	Decode to select a 45-Ohm resistance to the USB_DN output pin. Maximum resistance = 0000. Resistance is centered by design at 0110.
7–4 RSVD0	Reserved. **Note:** This bit should remain clear.
D_CAL	Resistor Trimming Code: 0000 = 0.16% 0111 = Nominal 1111 = +25%

43.4.3 USB PHY Receiver Control Register (USBPHY*x*_RX*n*)

The USB PHY Receiver Control Register handles receive path controls.

Address: Base address + 20h offset + (4d × i), where i=0d to 3d



USBPHY*x*_RX*n* field descriptions

Field	Description
31–23 RSVD2	Reserved.
22 RXDBYPASS	0 = Normal operation. 1 = Use the output of the USB_DP single-ended receiver in place of the full-speed differential receiver. This test mode is intended for lab use only.
21–7 RSVD1	Reserved.

Table continues on the next page...

USBPHYx_RXn field descriptions (continued)

Field	Description
6–4 DISCONADJ	The DISCONADJ field adjusts the trip point for the disconnect detector: 000 = Trip-Level Voltage is 0.57500 V 001 = Trip-Level Voltage is 0.56875 V 010 = Trip-Level Voltage is 0.58125 V 011 = Trip-Level Voltage is 0.58750 V 1XX = Reserved
3 RSVD0	Reserved.
ENVADJ	The ENVADJ field adjusts the trip point for the envelope detector. 000 = Trip-Level Voltage is 0.12500 V 001 = Trip-Level Voltage is 0.10000 V 010 = Trip-Level Voltage is 0.13750 V 011 = Trip-Level Voltage is 0.15000 V 1XX = Reserved

43.4.4 USB PHY General Control Register (USBPHYx_CTRLn)

The USB PHY General Control Register handles OTG and Host controls. This register also includes interrupt enables and connectivity detect enables and results.

Address: Base address + 30h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	SFTRST	CLKGATE	UTMI_SUSPENDM	HOST_FORCE_LS_SE0	OTG_ID_VALUE	RSVD1		FSDLL_RST_EN	ENVBUSCHG_WKUP	ENIDCHG_WKUP	ENDPDMCHG_WKUP	ENAUTOCLR_PHY_PWD	ENAUTOCLR_CLKGATE	ENAUTO_PWRON_PLL	WAKEUP_IRQ	ENIRQWAKEUP
W																
Reset	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	ENUTMILEVEL3	ENUTMILEVEL2	DATA_ON_LRADC	DEVPLUGIN_IRQ	ENIRQDEVPLUGIN	RESUME_IRQ	ENIRQRESUMEDETECT	RESUMEIRQSTICKY	ENOTGIDDETECT	OTG_ID_CHG_IRQ	DEVPLUGIN_POLARITY	ENDEVPLUGINDETECT	HOSTDISCONDETECT_IRQ	ENIRQHOSTDISCON	ENHOSTDISCONDETECT	ENOTG_ID_CHG_IRQ
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USBPHYx_CTRL*n* field descriptions

Field	Description
31 SFTRST	Writing a 1 to this bit will soft-reset the USBPHYx_PWD, USBPHYx_TX, USBPHYx_RX, and USBPHYx_CTRL registers. Set to 0 to release the PHY from reset.
30 CLKGATE	Gate UTMI Clocks. Clear to 0 to run clocks. Set to 1 to gate clocks. Set this to save power while the USB is not actively being used. Configuration state is kept while the clock is gated. Note this bit can be auto-cleared if there is any wakeup event when USB is suspended while ENAUTOCLR_CLKGATE bit of USBPHYx_CTRL is enabled.
29 UTMI_SUSPENDM	Used by the PHY to indicate a powered-down state. If all the power-down bits in the USBPHYx_PWD are enabled, UTMI_SUSPENDM will be 0, otherwise 1. UTMI_SUSPENDM is negative logic, as required by the UTMI specification.
28 HOST_FORCE_LS_SE0	Forces the next FS packet that is transmitted to have a EOP with LS timing. This bit is used in host mode for the resume sequence. After the packet is transferred, this bit is cleared. The design can use this function to force the LS SE0 or use the USBPHYx_CTRL_UTMI_SUSPENDM to trigger this event when leaving suspend. This bit is used in conjunction with USBPHYx_DEBUG_HOST_RESUME_DEBUG.
27 OTG_ID_VALUE	Almost same as OTGID_STATUS in USBPHYx_STATUS Register. The only difference is that OTG_ID_VALUE has debounce logic to filter the glitches on ID Pad.
26–25 RSVD1	Reserved.
24 FSDLL_RST_EN	Enables the feature to reset the FSDLL lock detection logic at the end of each TX packet.
23 ENVBUSCHG_WKUP	Enables the feature to wakeup USB if VBUS is toggled when USB is suspended.
22 ENIDCHG_WKUP	Enables the feature to wakeup USB if ID is toggled when USB is suspended.
21 ENDPDMCHG_WKUP	Enables the feature to wakeup USB if DP/DM is toggled when USB is suspended. This bit is enabled by default.
20 ENAUTOCLR_PHY_PWD	Enables the feature to auto-clear the PWD register bits in USBPHYx_PWD if there is wakeup event while USB is suspended. This should be enabled if needs to support auto wakeup without S/W's interaction.
19 ENAUTOCLR_CLKGATE	Enables the feature to auto-clear the CLKGATE bit if there is wakeup event while USB is suspended. This should be enabled if needs to support auto wakeup without S/W's interaction.
18 ENAUTO_PWRON_PLL	Enables the feature to auto-enable the POWER bit of HW_CLKCTRL_PLLxCTRL0 if there is wakeup event if USB is suspended. This should be enabled if needs to support auto wakeup without S/W's interaction.
17 WAKEUP_IRQ	Indicates that there is a wakeup event. Reset this bit by writing a 1 to the clear address space and not by a general write.
16 ENIRQWAKEUP	Enables interrupt for the wakeup events.
15 ENUTMILEVEL3	Enables UTMI+ Level3. This should be enabled if needs to support external FS Hub with LS device connected
14 ENUTMILEVEL2	Enables UTMI+ Level2. This should be enabled if needs to support LS device
13 DATA_ON_LRADC	Enables the LRADC to monitor USB_DP and USB_DM. This is for use in non-USB modes only.
12 DEVPLUGIN_IRQ	Indicates that the device is connected. Reset this bit by writing a 1 to the clear address space and not by a general write.

Table continues on the next page...

USBPHY*x*_CTRL*n* field descriptions (continued)

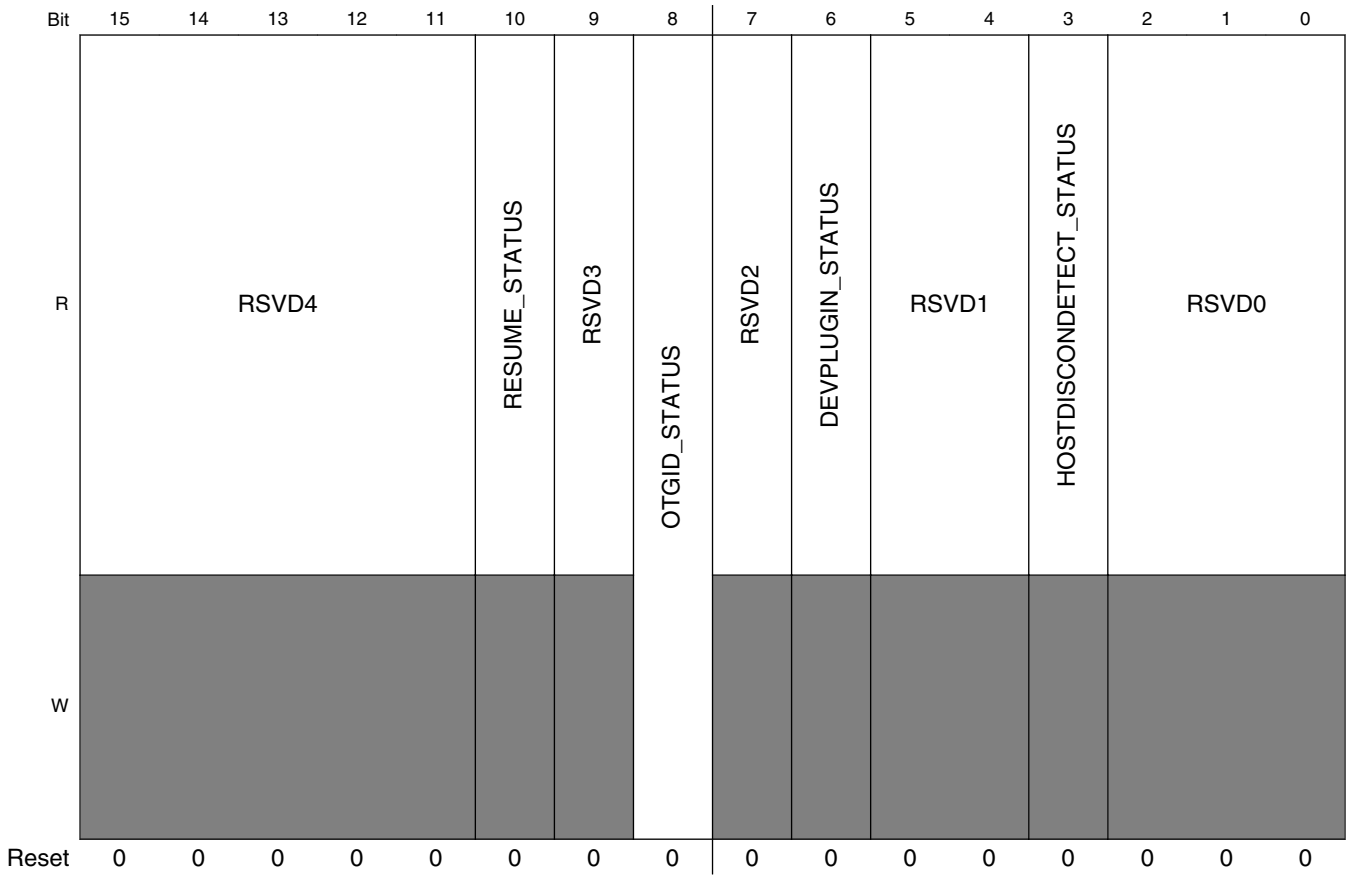
Field	Description
11 ENIRQDEVPLUGIN	Enables interrupt for the detection of connectivity to the USB line.
10 RESUME_IRQ	Indicates that the host is sending a wake-up after suspend. This bit is also set on a reset during suspend. Use this bit to wake up from suspend for either the resume or the reset case. Reset this bit by writing a 1 to the clear address space and not by a general write.
9 ENIRQRESUMEDETECT	Enables interrupt for detection of a non-J state on the USB line. This should only be enabled after the device has entered suspend mode.
8 RESUMEIRQSTICKY	Set to 1 will make RESUME_IRQ bit a sticky bit until software clear it. Set to 0, RESUME_IRQ only set during the wake-up period.
7 ENOTGIDDETECT	Enables circuit to detect resistance of MiniAB ID pin.
6 OTG_ID_CHG_IRQ	OTG ID change interrupt. Indicates the value of ID pin changed.
5 DEVPLUGIN_POLARITY	For device mode, if this bit is cleared to 0, then it trips the interrupt if the device is plugged in. If set to 1, then it trips the interrupt if the device is unplugged.
4 ENDEVPLUGINDETECT	For device mode, enables 200-KOhm pullups for detecting connectivity to the host.
3 HOSTDISCONDETECT_ IRQ	Indicates that the device has disconnected in high-speed mode. Reset this bit by writing a 1 to the clear address space and not by a general write.
2 ENIRQHOSTDISCON	Enables interrupt for detection of disconnection to Device when in high-speed host mode. This should be enabled after ENDEVPLUGINDETECT is enabled.
1 ENHOSTDISCONDETECT	For host mode, enables high-speed disconnect detector. This signal allows the override of enabling the detection that is normally done in the UTMI controller. The UTMI controller enables this circuit whenever the host sends a start-of-frame packet. SW shall set this bit when it found the high-speed device is connected, suggested during bus reset, after found high-speed device in USB_PORTSC1.PSPD). SW shall make sure this bit is not set at the end of resume, otherwise a wrong disconnect status may be detected. Suggest clear it after set USB_PORTSC1.SUSP, set it again after resume is ended(USB_PORTSC1.FPR==0).
0 ENOTG_ID_CHG_IRQ	Enable OTG_ID_CHG_IRQ.

43.4.5 USB PHY Status Register (USBPHYx_STATUS)

The USB PHY Status Register holds results of IRQ and other detects.

Address: Base address + 40h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RSVD4															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



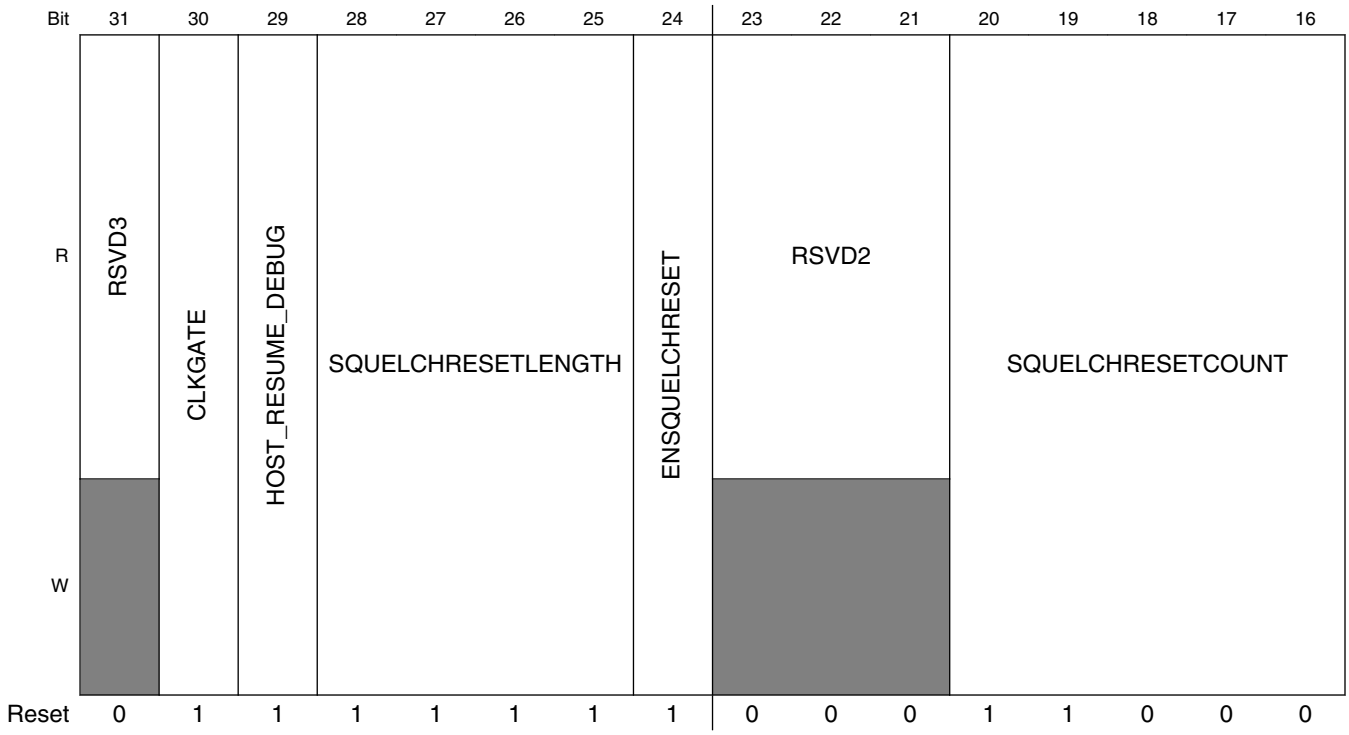
USBPHYx_STATUS field descriptions

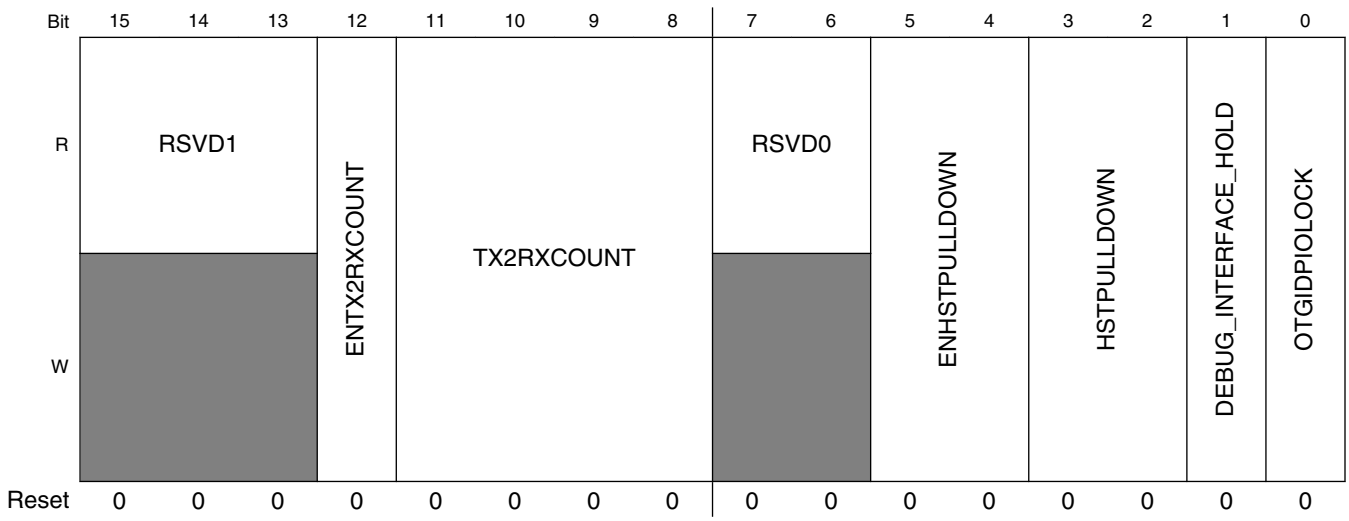
Field	Description
31–11 RSVD4	Reserved.
10 RESUME_STATUS	Indicates that the host is sending a wake-up after suspend and has triggered an interrupt.
9 RSVD3	Reserved.
8 OTGID_STATUS	Indicates the results of ID pin on MiniAB plug. False (0) is when ID resistance is less than Ra_Plug_ID, indicating host (A) side. True (1) is when ID resistance is greater than Rb_Plug_ID, indicating device (B) side.
7 RSVD2	Reserved.
6 DEVPLUGIN_STATUS	Indicates that the device has been connected on the USB_DP and USB_DM lines.
5–4 RSVD1	Reserved.
3 HOSTDISCONDETECT_ STATUS	Indicates that the device has disconnected while in high-speed host mode.
RSVD0	Reserved.

43.4.6 USB PHY Debug Register (USBPHYx_DEBUG*n*)

This register is used to debug the USB PHY.

Address: Base address + 50h offset + (4d × i), where i=0d to 3d





USBPHYx_DEBUG*n* field descriptions

Field	Description
31 RSVD3	Reserved.
30 CLKGATE	Gate Test Clocks. Clear to 0 for running clocks. Set to 1 to gate clocks. Set this to save power while the USB is not actively being used. Configuration state is kept while the clock is gated.
29 HOST_RESUME_DEBUG	Choose to trigger the host resume SE0 with HOST_FORCE_LS_SE0 = 0 or UTMI_SUSPEND = 1.
28–25 SQUELCHRESETLENGTH	Duration of RESET in terms of the number of 480-MHz cycles.
24 ENSQUELCHRESET	Set bit to allow squelch to reset high-speed receive.
23–21 RSVD2	Reserved.
20–16 SQUELCHRESETCOUNT	Delay in between the detection of squelch to the reset of high-speed RX.
15–13 RSVD1	Reserved.
12 ENTX2RXCOUNT	Set this bit to allow a countdown to transition in between TX and RX.
11–8 TX2RXCOUNT	Delay in between the end of transmit to the beginning of receive. This is a Johnson count value and thus will count to 8.
7–6 RSVD0	Reserved.
5–4 ENHSTPULLDOWN	Set bit 5 to 1 to override the control of the USB_DP 15-KOhm pulldown. Set bit 4 to 1 to override the control of the USB_DM 15-KOhm pulldown. Clear to 0 to disable.
3–2 HSTPULLDOWN	Set bit 3 to 1 to pull down 15-KOhm on USB_DP line. Set bit 2 to 1 to pull down 15-KOhm on USB_DM line. Clear to 0 to disable.
1 DEBUG_INTERFACE_HOLD	Use holding registers to assist in timing for external UTMI interface.
0 OTGIDPIOLOCK	Once OTG ID from USBPHYx_STATUS_OTGID_STATUS, use this to hold the value. This is to save power for the comparators that are used to determine the ID status.

43.4.7 UTMI Debug Status Register 0 (USBPHYx_DEBUG0_STATUS)

The UTMI Debug Status Register 0 holds multiple views for counters and status of state machines. This is used in conjunction with the USBPHYx_DEBUG1_DBG_ADDRESS field to choose which function to view. The default is described in the bit fields below and is used to count errors.

Address: Base address + 60h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SQUELCH_COUNT						UTMI_RXERROR_FAIL_COUNT										LOOP_BACK_FAIL_COUNT															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

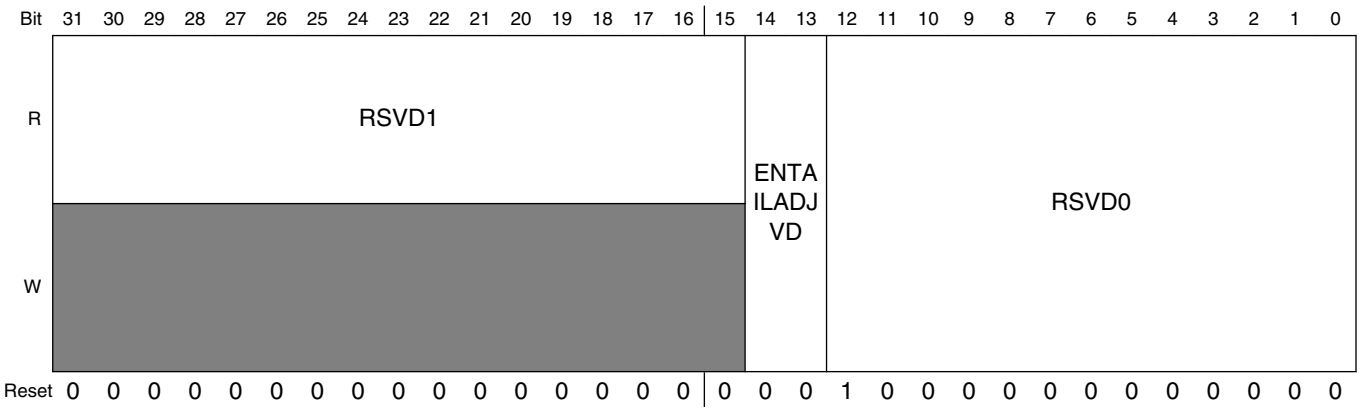
USBPHYx_DEBUG0_STATUS field descriptions

Field	Description
31–26 SQUELCH_ COUNT	Running count of the squelch reset instead of normal end for HS RX.
25–16 UTMI_ RXERROR_ FAIL_COUNT	Running count of the UTMI_RXERROR.
LOOP_BACK_ FAIL_COUNT	Running count of the failed pseudo-random generator loopback. Each time entering testmode, counter goes to 900D and will count up for every detected packet failure in digital/analog loopback tests.

43.4.8 UTMI Debug Status Register 1 (USBPHYx_DEBUG1*n*)

Chooses the muxing of the debug register to be shown in USBPHYx_DEBUG0_STATUS.

Address: Base address + 70h offset + (4d × i), where i=0d to 3d



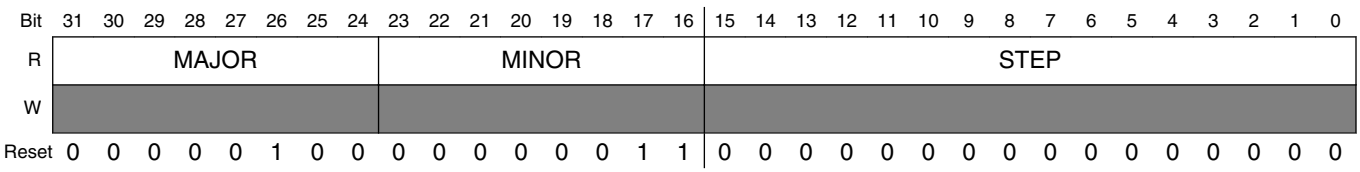
USBPHYx_DEBUG1*n* field descriptions

Field	Description
31–15 RSVD1	Reserved.
14–13 ENTAILADJVD	Delay increment of the rise of squelch: 00 = Delay is nominal 01 = Delay is +20% 10 = Delay is -20% 11 = Delay is -40%
RSVD0	Reserved. **Note:** This bit should remain clear.

43.4.9 UTMI RTL Version (USBPHYx_VERSION)

Fields for RTL Version.

Address: Base address + 80h offset



USBPHYx_VERSION field descriptions

Field	Description
31–24 MAJOR	Fixed read-only value reflecting the MAJOR field of the RTL version.
23–16 MINOR	Fixed read-only value reflecting the MINOR field of the RTL version.
STEP	Fixed read-only value reflecting the stepping of the RTL version.

43.5 USB Analog Memory Map/Register Definition

USB_ANALOG memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400D_81A0	USB VBUS Detect Register (USB_ANALOG_USB1_VBUS_DETECT)	32	R/W	0010_0004h	43.5.1/2541
400D_81A4	USB VBUS Detect Register (USB_ANALOG_USB1_VBUS_DETECT_SET)	32	R/W	0010_0004h	43.5.1/2541
400D_81A8	USB VBUS Detect Register (USB_ANALOG_USB1_VBUS_DETECT_CLR)	32	R/W	0010_0004h	43.5.1/2541
400D_81AC	USB VBUS Detect Register (USB_ANALOG_USB1_VBUS_DETECT_TOG)	32	R/W	0010_0004h	43.5.1/2541
400D_81B0	USB Charger Detect Register (USB_ANALOG_USB1_CHRG_DETECT)	32	R/W	0000_0000h	43.5.2/2543
400D_81B4	USB Charger Detect Register (USB_ANALOG_USB1_CHRG_DETECT_SET)	32	R/W	0000_0000h	43.5.2/2543
400D_81B8	USB Charger Detect Register (USB_ANALOG_USB1_CHRG_DETECT_CLR)	32	R/W	0000_0000h	43.5.2/2543
400D_81BC	USB Charger Detect Register (USB_ANALOG_USB1_CHRG_DETECT_TOG)	32	R/W	0000_0000h	43.5.2/2543

Table continues on the next page...

USB_ANALOG memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400D_81C0	USB VBUS Detect Status Register (USB_ANALOG_USB1_VBUS_DETECT_STAT)	32	R	0000_0000h	43.5.3/2545
400D_81D0	USB Charger Detect Status Register (USB_ANALOG_USB1_CHRG_DETECT_STAT)	32	R	0000_0000h	43.5.4/2547
400D_81E0	USB Loopback Test Register (USB_ANALOG_USB1_LOOPBACK)	32	R/W	0000_0000h	43.5.5/2548
400D_81E4	USB Loopback Test Register (USB_ANALOG_USB1_LOOPBACK_SET)	32	R/W	0000_0000h	43.5.5/2548
400D_81E8	USB Loopback Test Register (USB_ANALOG_USB1_LOOPBACK_CLR)	32	R/W	0000_0000h	43.5.5/2548
400D_81EC	USB Loopback Test Register (USB_ANALOG_USB1_LOOPBACK_TOG)	32	R/W	0000_0000h	43.5.5/2548
400D_81F0	USB Misc Register (USB_ANALOG_USB1_MISC)	32	R/W	0000_0002h	43.5.6/2549
400D_81F4	USB Misc Register (USB_ANALOG_USB1_MISC_SET)	32	R/W	0000_0002h	43.5.6/2549
400D_81F8	USB Misc Register (USB_ANALOG_USB1_MISC_CLR)	32	R/W	0000_0002h	43.5.6/2549
400D_81FC	USB Misc Register (USB_ANALOG_USB1_MISC_TOG)	32	R/W	0000_0002h	43.5.6/2549
400D_8200	USB VBUS Detect Register (USB_ANALOG_USB2_VBUS_DETECT)	32	R/W	0010_0004h	43.5.7/2550
400D_8204	USB VBUS Detect Register (USB_ANALOG_USB2_VBUS_DETECT_SET)	32	R/W	0010_0004h	43.5.7/2550
400D_8208	USB VBUS Detect Register (USB_ANALOG_USB2_VBUS_DETECT_CLR)	32	R/W	0010_0004h	43.5.7/2550
400D_820C	USB VBUS Detect Register (USB_ANALOG_USB2_VBUS_DETECT_TOG)	32	R/W	0010_0004h	43.5.7/2550
400D_8210	USB Charger Detect Register (USB_ANALOG_USB2_CHRG_DETECT)	32	R/W	0000_0000h	43.5.8/2552
400D_8214	USB Charger Detect Register (USB_ANALOG_USB2_CHRG_DETECT_SET)	32	R/W	0000_0000h	43.5.8/2552
400D_8218	USB Charger Detect Register (USB_ANALOG_USB2_CHRG_DETECT_CLR)	32	R/W	0000_0000h	43.5.8/2552
400D_821C	USB Charger Detect Register (USB_ANALOG_USB2_CHRG_DETECT_TOG)	32	R/W	0000_0000h	43.5.8/2552
400D_8220	USB VBUS Detect Status Register (USB_ANALOG_USB2_VBUS_DETECT_STAT)	32	R	0000_0000h	43.5.9/2554
400D_8230	USB Charger Detect Status Register (USB_ANALOG_USB2_CHRG_DETECT_STAT)	32	R	0000_0000h	43.5.10/ 2556
400D_8240	USB Loopback Test Register (USB_ANALOG_USB2_LOOPBACK)	32	R/W	0000_0000h	43.5.11/ 2557
400D_8244	USB Loopback Test Register (USB_ANALOG_USB2_LOOPBACK_SET)	32	R/W	0000_0000h	43.5.11/ 2557
400D_8248	USB Loopback Test Register (USB_ANALOG_USB2_LOOPBACK_CLR)	32	R/W	0000_0000h	43.5.11/ 2557
400D_824C	USB Loopback Test Register (USB_ANALOG_USB2_LOOPBACK_TOG)	32	R/W	0000_0000h	43.5.11/ 2557

Table continues on the next page...

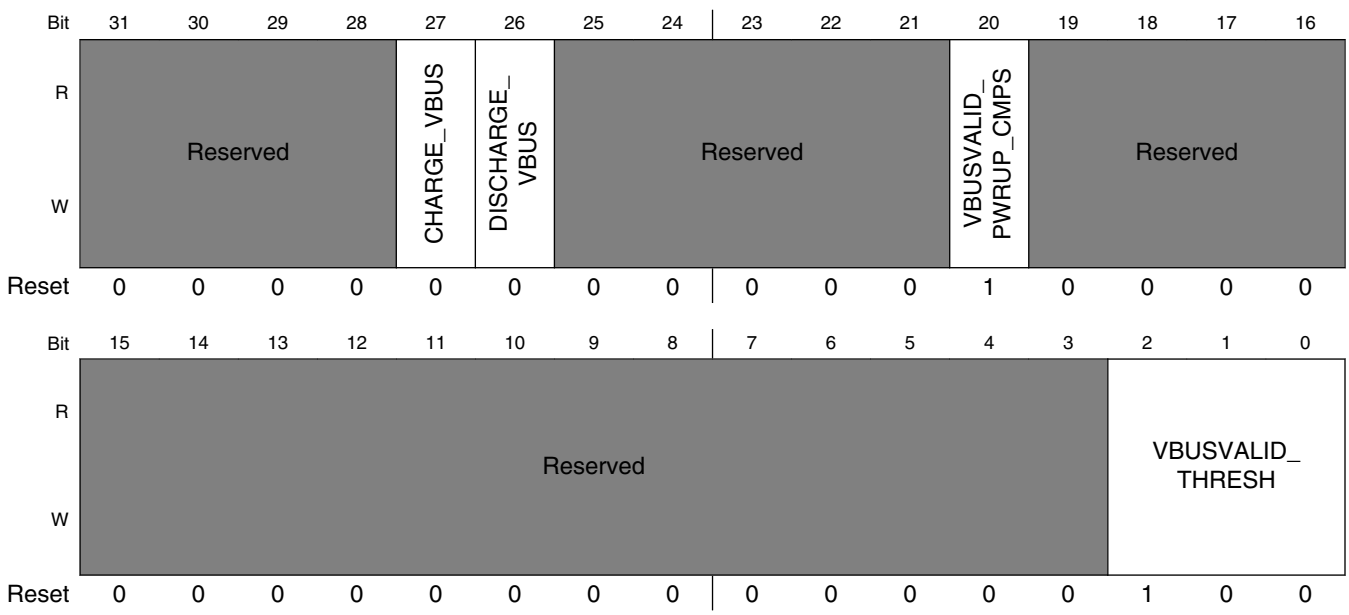
USB_ANALOG memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400D_8250	USB Misc Register (USB_ANALOG_USB2_MISC)	32	R/W	0000_0002h	43.5.12/ 2558
400D_8254	USB Misc Register (USB_ANALOG_USB2_MISC_SET)	32	R/W	0000_0002h	43.5.12/ 2558
400D_8258	USB Misc Register (USB_ANALOG_USB2_MISC_CLR)	32	R/W	0000_0002h	43.5.12/ 2558
400D_825C	USB Misc Register (USB_ANALOG_USB2_MISC_TOG)	32	R/W	0000_0002h	43.5.12/ 2558
400D_8260	Chip Silicon Version (USB_ANALOG_DIGPROG)	32	R	006C_0000h	43.5.13/ 2559

43.5.1 USB VBUS Detect Register (USB_ANALOG_USB1_VBUS_DETECT*n*)

This register defines controls for USB VBUS detect.

Address: 400D_8000h base + 1A0h offset + (4d × i), where i=0d to 3d



USB_ANALOG_USB1_VBUS_DETECT*n* field descriptions

Field	Description
31–28 -	This field is reserved. Reserved.

Table continues on the next page...

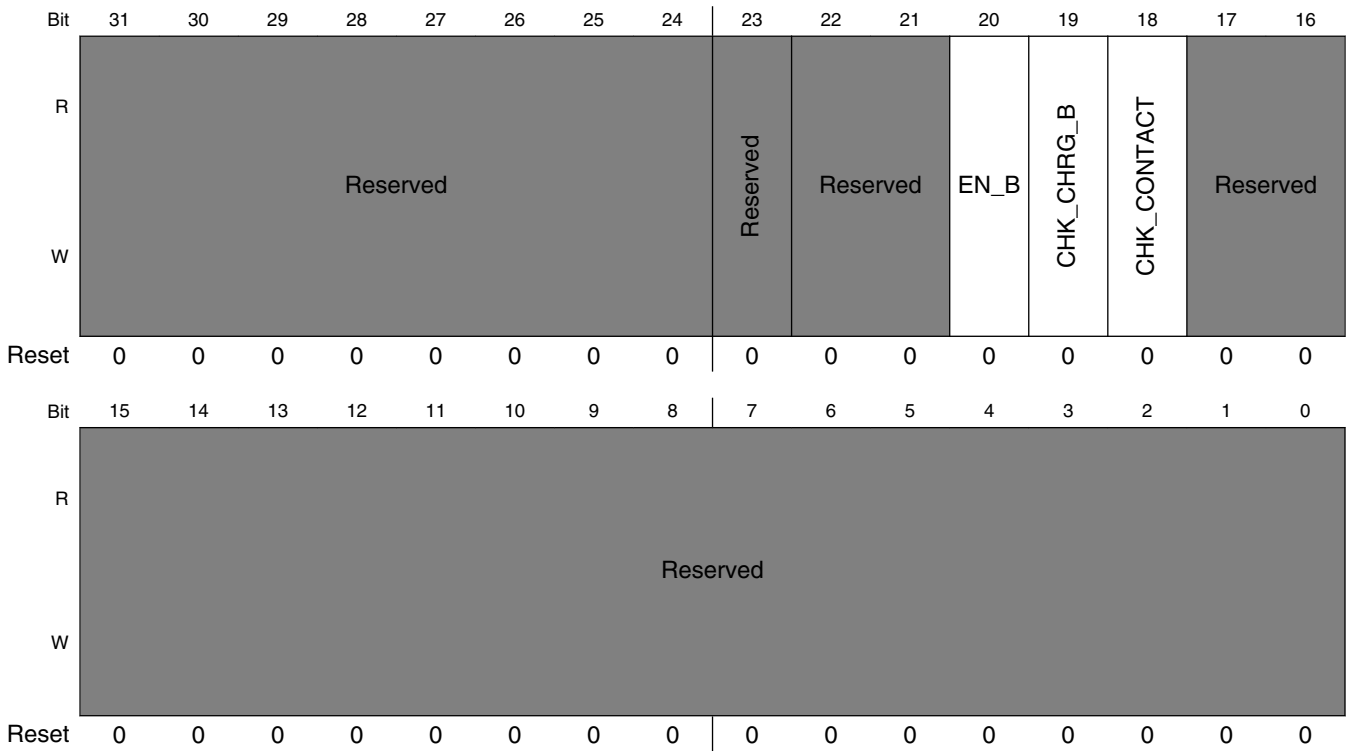
USB_ANALOG_USB1_VBUS_DETECT*n* field descriptions (continued)

Field	Description
27 CHARGE_VBUS	USB OTG charge VBUS.
26 DISCHARGE_ VBUS	USB OTG discharge VBUS.
25–21 -	This field is reserved. Reserved.
20 VBUSVALID_ PWRUP_CMPS	Powers up comparators for vbus_valid detector.
19–3 -	This field is reserved. Reserved.
VBUSVALID_ THRESH	Set the threshold for the VBUSVALID comparator. This comparator is the most accurate method to determine the presence of 5v, and includes hystersis to minimize the need for software debounce of the detection. This comparator has ~50mV of hystersis to prevent chattering at the comparator trip point. 000 **4V0** — 4.0V 001 **4V1** — 4.1V 010 **4V2** — 4.2V 011 **4V3** — 4.3V 100 **4V4** — 4.4V (default) 101 **4V5** — 4.5V 110 **4V6** — 4.6V 111 **4V7** — 4.7V

43.5.2 USB Charger Detect Register (USB_ANALOG_USB1_CHRG_DETECT*n*)

This register defines controls for USB charger detect.

Address: 400D_8000h base + 1B0h offset + (4d × i), where i=0d to 3d



USB_ANALOG_USB1_CHRG_DETECT*n* field descriptions

Field	Description
31–24 -	This field is reserved. Reserved.
23 -	This field is reserved. Reserved.
22–21 -	This field is reserved. Reserved.
20 EN_B	Control the charger detector. 0 **ENABLE** — Enable the charger detector. 1 **DISABLE** — Disable the charger detector.
19 CHK_CHRG_B	Check the charger connection

Table continues on the next page...

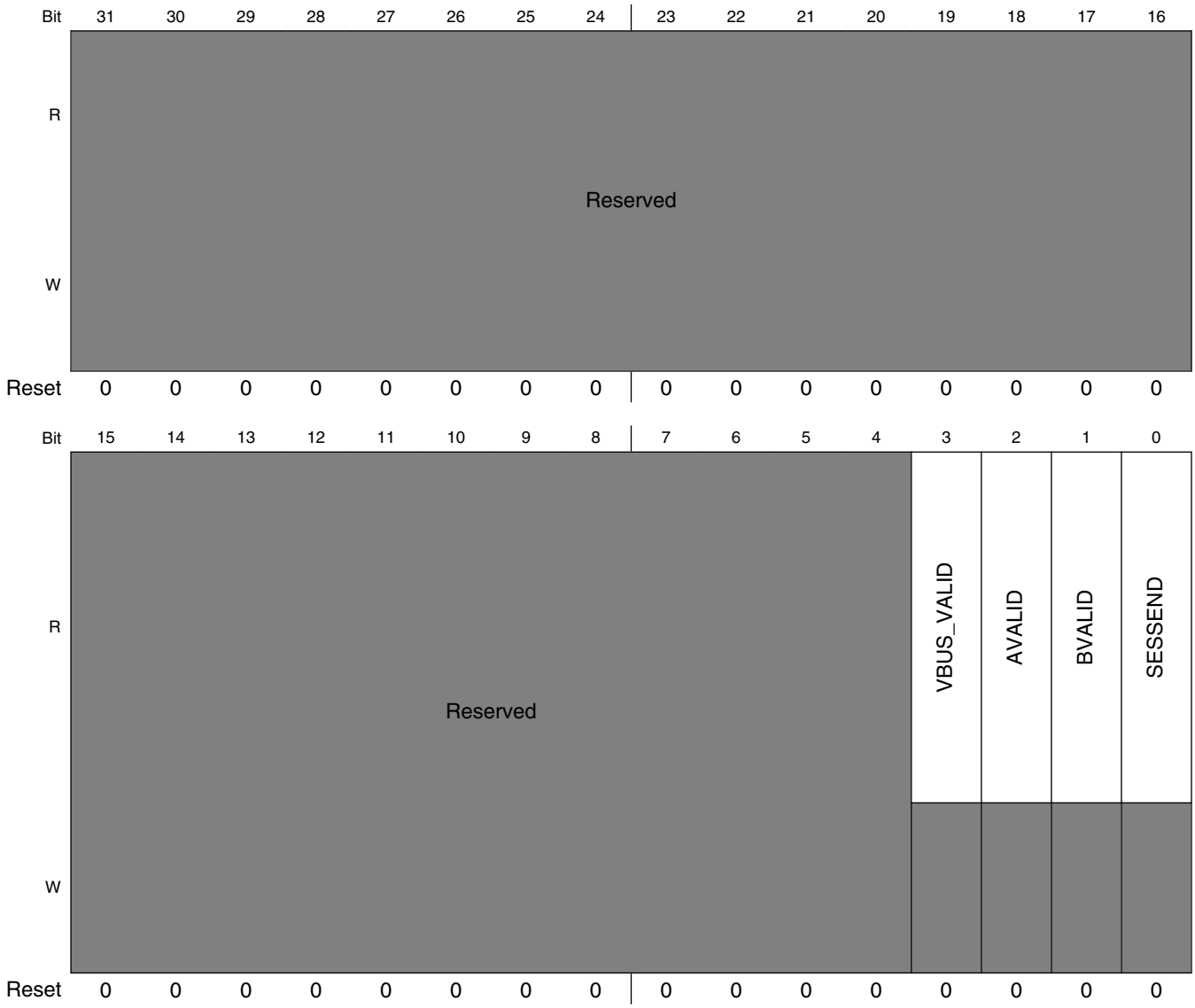
USB_ANALOG_USB1_CHRG_DETECT*n* field descriptions (continued)

Field	Description
	0 **CHECK** — Check whether a charger (either a dedicated charger or a host charger) is connected to USB port. 1 **NO_CHECK** — Do not check whether a charger is connected to the USB port.
18 CHK_CONTACT	Check the contact of USB plug 0 **NO_CHECK** — Do not check the contact of USB plug. 1 **CHECK** — Check whether the USB plug has been in contact with each other
-	This field is reserved. Reserved.

43.5.3 USB VBUS Detect Status Register (USB_ANALOG_USB1_VBUS_DETECT_STAT)

This register defines fields for USB VBUS Detect status.

Address: 400D_8000h base + 1C0h offset = 400D_81C0h



USB_ANALOG_USB1_VBUS_DETECT_STAT field descriptions

Field	Description
31–4 -	This field is reserved. Reserved.

Table continues on the next page...

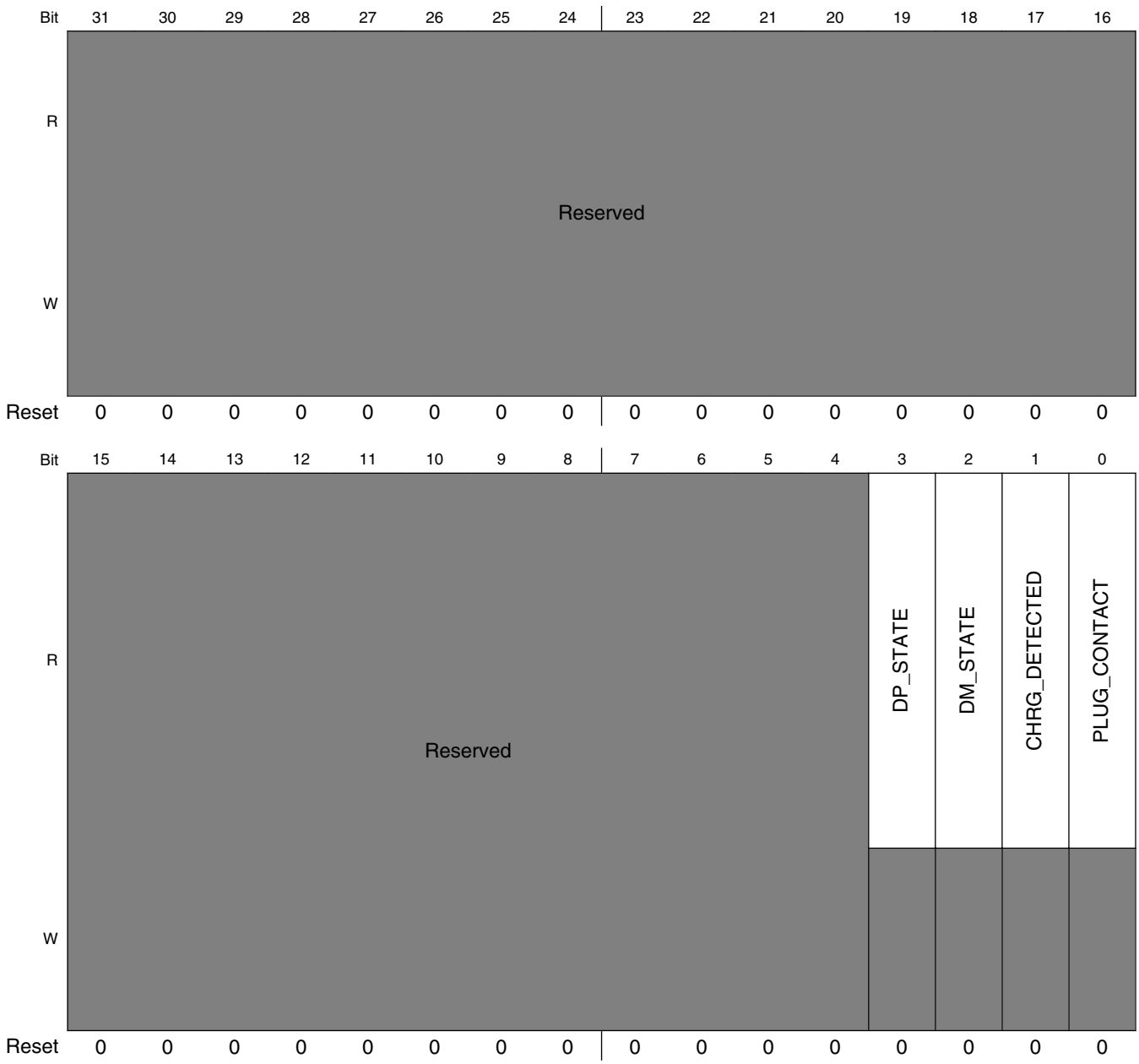
USB_ANALOG_USB1_VBUS_DETECT_STAT field descriptions (continued)

Field	Description
3 VBUS_VALID	VBus valid for USB OTG. This bit is a read only version of the state of the analog signal. It can not be overwritten by software.
2 AVALID	Indicates VBus is valid for a A-peripheral. This bit is a read only version of the state of the analog signal. It can not be overritten by software.
1 BVALID	Indicates VBus is valid for a B-peripheral. This bit is a read only version of the state of the analog signal. It can not be overritten by software.
0 SESSEND	Session End for USB OTG. This bit is a read only version of the state of the analog signal. It can not be overwritten by software like the SESSEND bit below. NOTE: This bit's default value depends on whether VDD5V is present, 0 if VDD5V is present, 1 if VDD5V is not present.

43.5.4 USB Charger Detect Status Register (USB_ANALOG_USB1_CHRG_DETECT_STAT)

This register defines fields for USB charger detect status.

Address: 400D_8000h base + 1D0h offset = 400D_81D0h



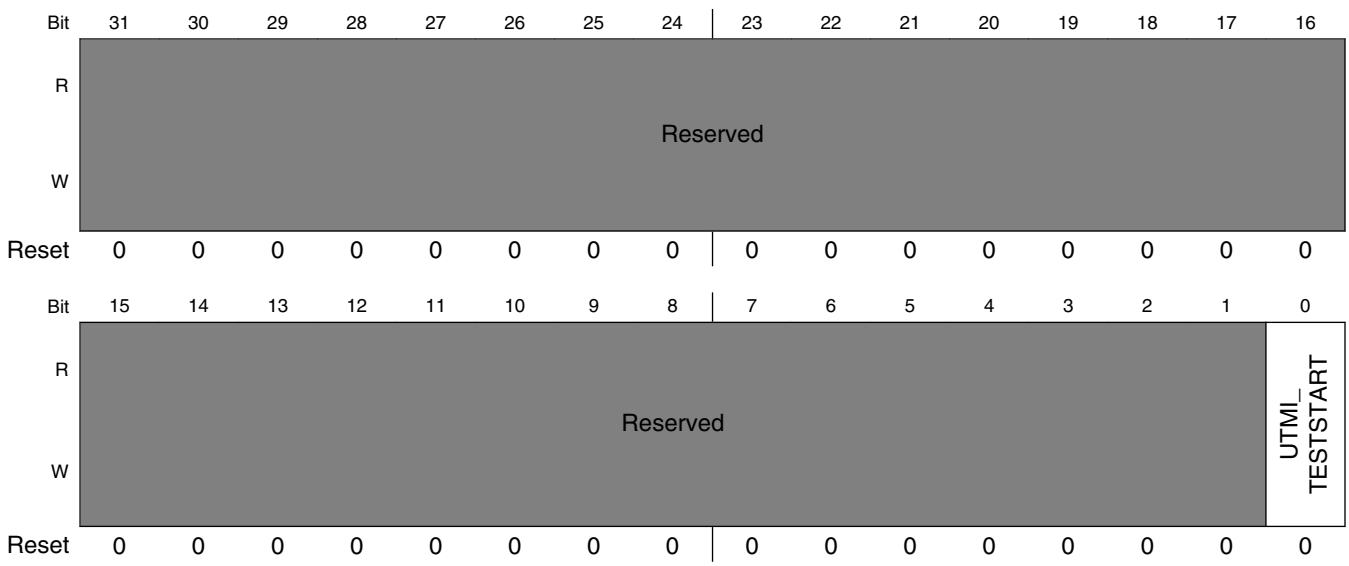
USB_ANALOG_USB1_CHRG_DETECT_STAT field descriptions

Field	Description
31–4 -	This field is reserved. Reserved.
3 DP_STATE	DP line state output of the charger detector.
2 DM_STATE	DM line state output of the charger detector.
1 CHRG_ DETECTED	State of charger detection. This bit is a read only version of the state of the analog signal. 0 **CHARGER_NOT_PRESENT** — The USB port is not connected to a charger. 1 **CHARGER_PRESENT** — A charger (either a dedicated charger or a host charger) is connected to the USB port.
0 PLUG_ CONTACT	State of the USB plug contact detector. 0 **NO_CONTACT** — The USB plug has not made contact. 1 **GOOD_CONTACT** — The USB plug has made good contact.

43.5.5 USB Loopback Test Register (USB_ANALOG_USB1_LOOPBACK*n*)

This register defines controls for the USB1 loopback test function.

Address: 400D_8000h base + 1E0h offset + (4d × i), where i=0d to 3d



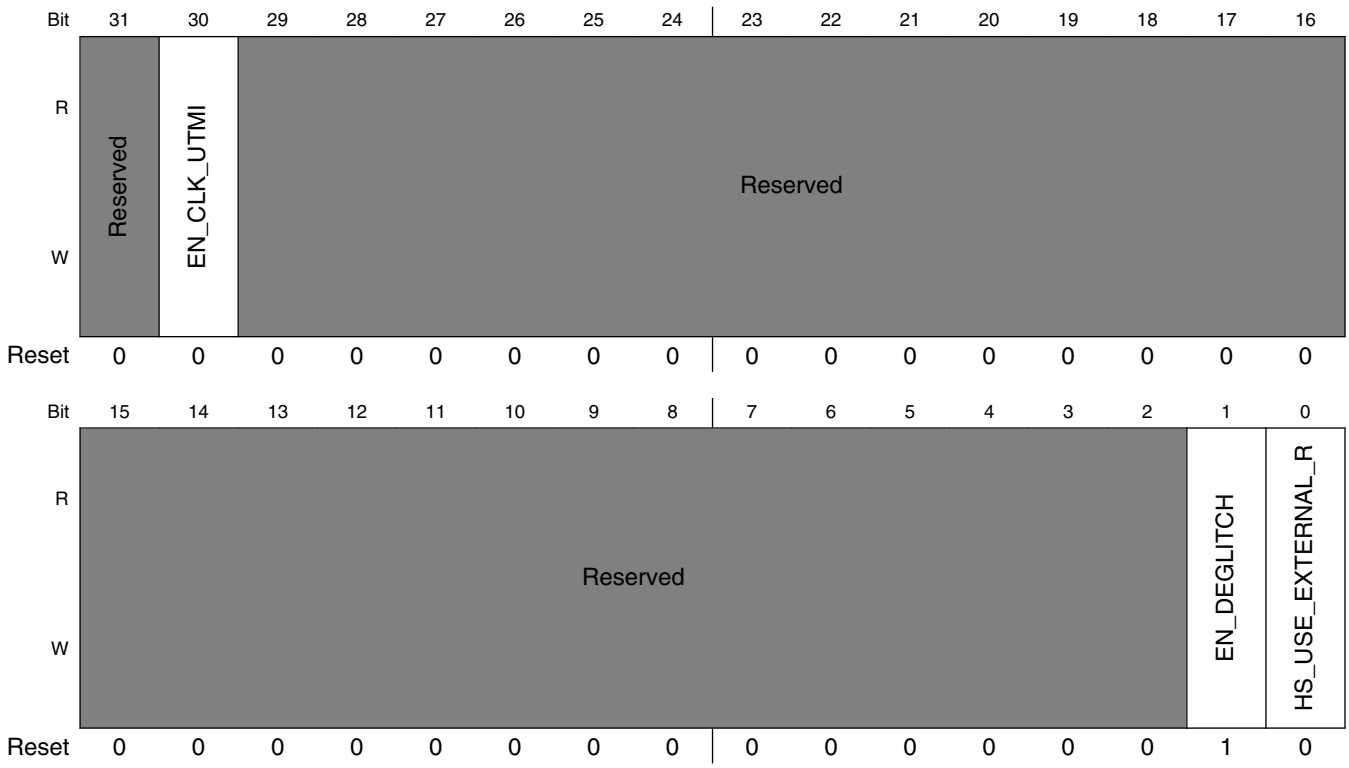
USB_ANALOG_USB1_LOOPBACK*n* field descriptions

Field	Description
31–1 -	This field is reserved. Reserved.
0 UTMI_ TESTSTART	Setting this bit can enable 1.5 kΩ pull-up resister on DP. **NOTE:** This bit can only be used as DCD detection, while it must be cleared in normal function.

43.5.6 USB Misc Register (USB_ANALOG_USB1_MISC*n*)

This register defines controls for USB.

Address: 400D_8000h base + 1F0h offset + (4d × i), where i=0d to 3d



USB_ANALOG_USB1_MISC*n* field descriptions

Field	Description
31 -	This field is reserved. Reserved.
30 EN_CLK_UTMI	Enables the clk to the UTMI block.
29–2 -	This field is reserved. Reserved.

Table continues on the next page...

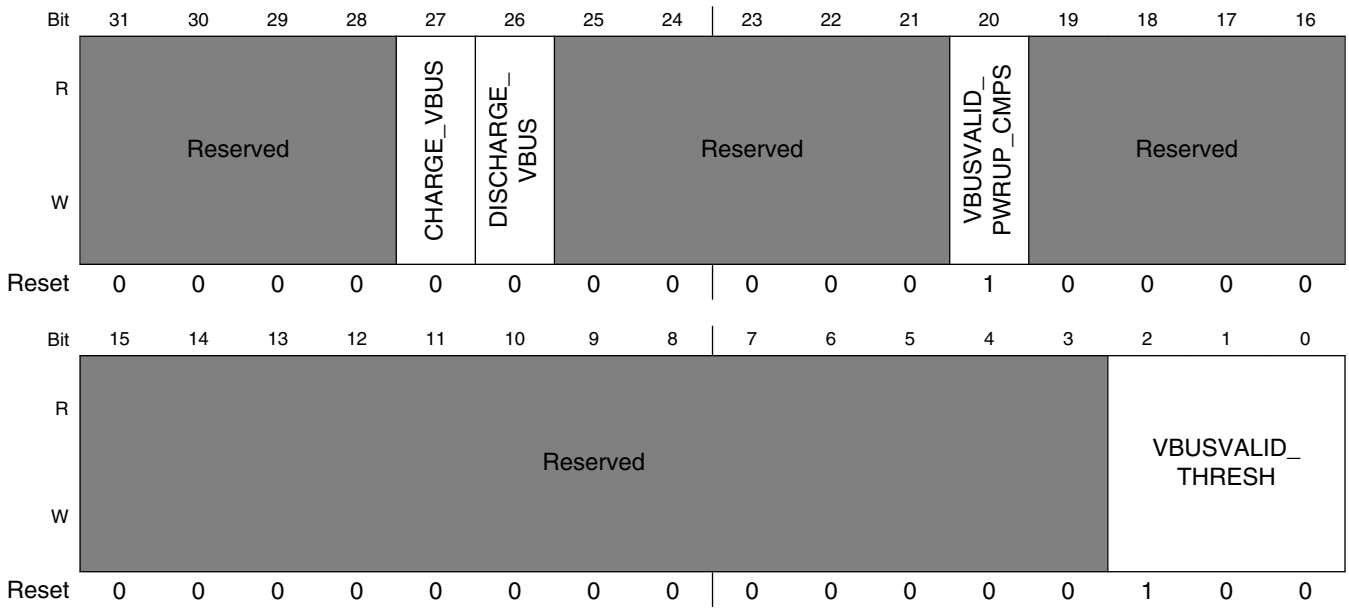
USB_ANALOG_USB1_MISC*n* field descriptions (continued)

Field	Description
1 EN_DEGLITCH	Enable the deglitching circuit of the USB PLL output.
0 HS_USE_ EXTERNAL_R	Use external resistor to generate the current bias for the high speed transmitter. This bit should not be changed unless recommended by NXP.

43.5.7 USB VBUS Detect Register (USB_ANALOG_USB2_VBUS_DETECT*n*)

This register defines controls for USB VBUS detect.

Address: 400D_8000h base + 200h offset + (4d × i), where i=0d to 3d



USB_ANALOG_USB2_VBUS_DETECT*n* field descriptions

Field	Description
31–28 -	This field is reserved. Reserved.
27 CHARGE_VBUS	USB OTG charge VBUS.
26 DISCHARGE_ VBUS	USB OTG discharge VBUS.
25–21 -	This field is reserved. Reserved.

Table continues on the next page...

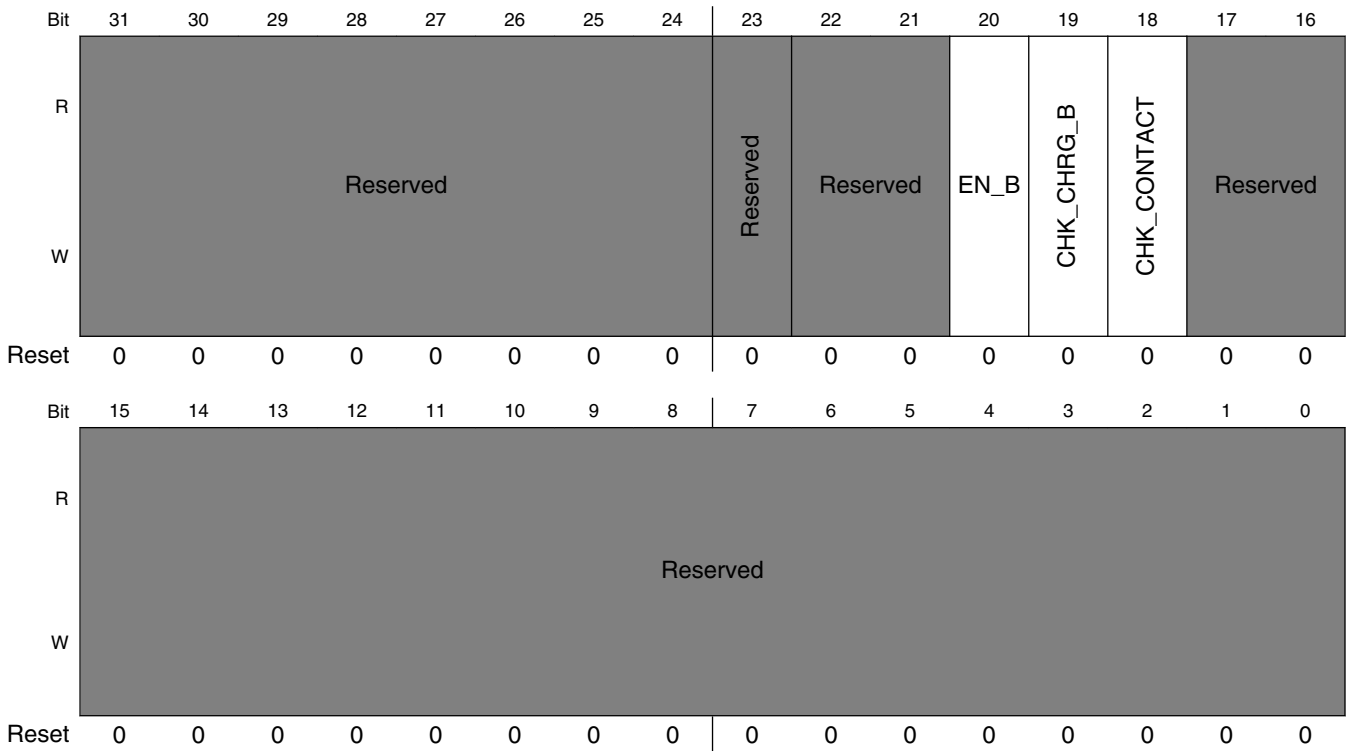
USB_ANALOG_USB2_VBUS_DETECT*n* field descriptions (continued)

Field	Description
20 VBUSVALID_ PWRUP_CMPS	Powers up comparators for vbus_valid detector.
19–3 -	This field is reserved. Reserved.
VBUSVALID_ THRESH	Set the threshold for the VBUSVALID comparator. This comparator is the most accurate method to determine the presence of 5v, and includes hystersis to minimize the need for software debounce of the detection. This comparator has ~50mV of hystersis to prevent chattering at the comparator trip point. 000 **4V0** — 4.0V 001 **4V1** — 4.1V 010 **4V2** — 4.2V 011 **4V3** — 4.3V 100 **4V4** — 4.4V (default) 101 **4V5** — 4.5V 110 **4V6** — 4.6V 111 **4V7** — 4.7V

43.5.8 USB Charger Detect Register (USB_ANALOG_USB2_CHRG_DETECT*n*)

This register defines controls for USB charger detect.

Address: 400D_8000h base + 210h offset + (4d × i), where i=0d to 3d



USB_ANALOG_USB2_CHRG_DETECT*n* field descriptions

Field	Description
31–24 -	This field is reserved. Reserved.
23 -	This field is reserved. Reserved.
22–21 -	This field is reserved. Reserved.
20 EN_B	Control the charger detector. 0 **ENABLE** — Enable the charger detector. 1 **DISABLE** — Disable the charger detector.
19 CHK_CHRG_B	Check the charger connection

Table continues on the next page...

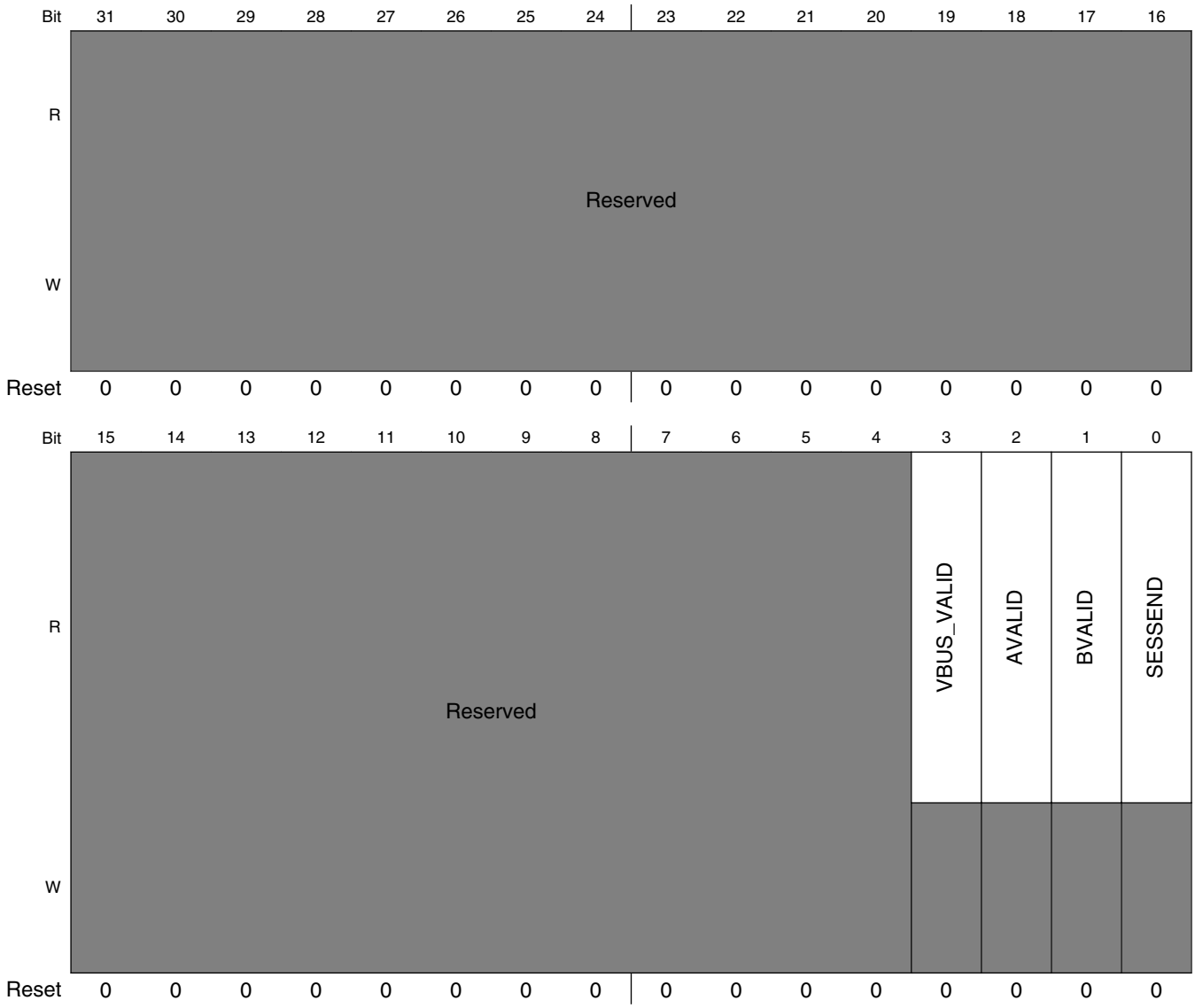
USB_ANALOG_USB2_CHRG_DETECT*n* field descriptions (continued)

Field	Description
	0 **CHECK** — Check whether a charger (either a dedicated charger or a host charger) is connected to USB port. 1 **NO_CHECK** — Do not check whether a charger is connected to the USB port.
18 CHK_CONTACT	Check the contact of USB plug 0 **NO_CHECK** — Do not check the contact of USB plug. 1 **CHECK** — Check whether the USB plug has been in contact with each other
-	This field is reserved. Reserved.

43.5.9 USB VBUS Detect Status Register (USB_ANALOG_USB2_VBUS_DETECT_STAT)

This register defines fields for USB VBUS Detect status.

Address: 400D_8000h base + 220h offset = 400D_8220h



USB_ANALOG_USB2_VBUS_DETECT_STAT field descriptions

Field	Description
31–4 -	This field is reserved. Reserved.

Table continues on the next page...

USB_ANALOG_USB2_VBUS_DETECT_STAT field descriptions (continued)

Field	Description
3 VBUS_VALID	VBus valid for USB OTG. This bit is a read only version of the state of the analog signal. It can not be overwritten by software.
2 AVALID	Indicates VBus is valid for a A-peripheral. This bit is a read only version of the state of the analog signal. It can not be overritten by software.
1 BVALID	Indicates VBus is valid for a B-peripheral. This bit is a read only version of the state of the analog signal. It can not be overritten by software.
0 SESSEND	Session End for USB OTG. This bit is a read only version of the state of the analog signal. It can not be overwritten by software like the SESSEND bit below. NOTE: This bit's default value depends on whether VDD5V is present, 0 if VDD5V is present, 1 if VDD5V is not present.

43.5.10 USB Charger Detect Status Register (USB_ANALOG_USB2_CHRG_DETECT_STAT)

This register defines fields for USB charger detect status.

Address: 400D_8000h base + 230h offset = 400D_8230h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved												DP_STATE	DM_STATE	CHRG_DETECTED	PLUG_CONTACT
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USB_ANALOG_USB2_CHRG_DETECT_STAT field descriptions

Field	Description
31–4 -	This field is reserved. Reserved.
3 DP_STATE	DP line state output of the charger detector.
2 DM_STATE	DM line state output of the charger detector.
1 CHRG_ DETECTED	State of charger detection. This bit is a read only version of the state of the analog signal. 0 **CHARGER_NOT_PRESENT** — The USB port is not connected to a charger. 1 **CHARGER_PRESENT** — A charger (either a dedicated charger or a host charger) is connected to the USB port.
0 PLUG_ CONTACT	State of the USB plug contact detector. 0 **NO_CONTACT** — The USB plug has not made contact. 1 **GOOD_CONTACT** — The USB plug has made good contact.

43.5.11 USB Loopback Test Register (USB_ANALOG_USB2_LOOPBACK*n*)

This register defines controls for the USB2 loopback test function.

Address: 400D_8000h base + 240h offset + (4d × i), where i=0d to 3d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved															UTMI_ TESTSTART
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

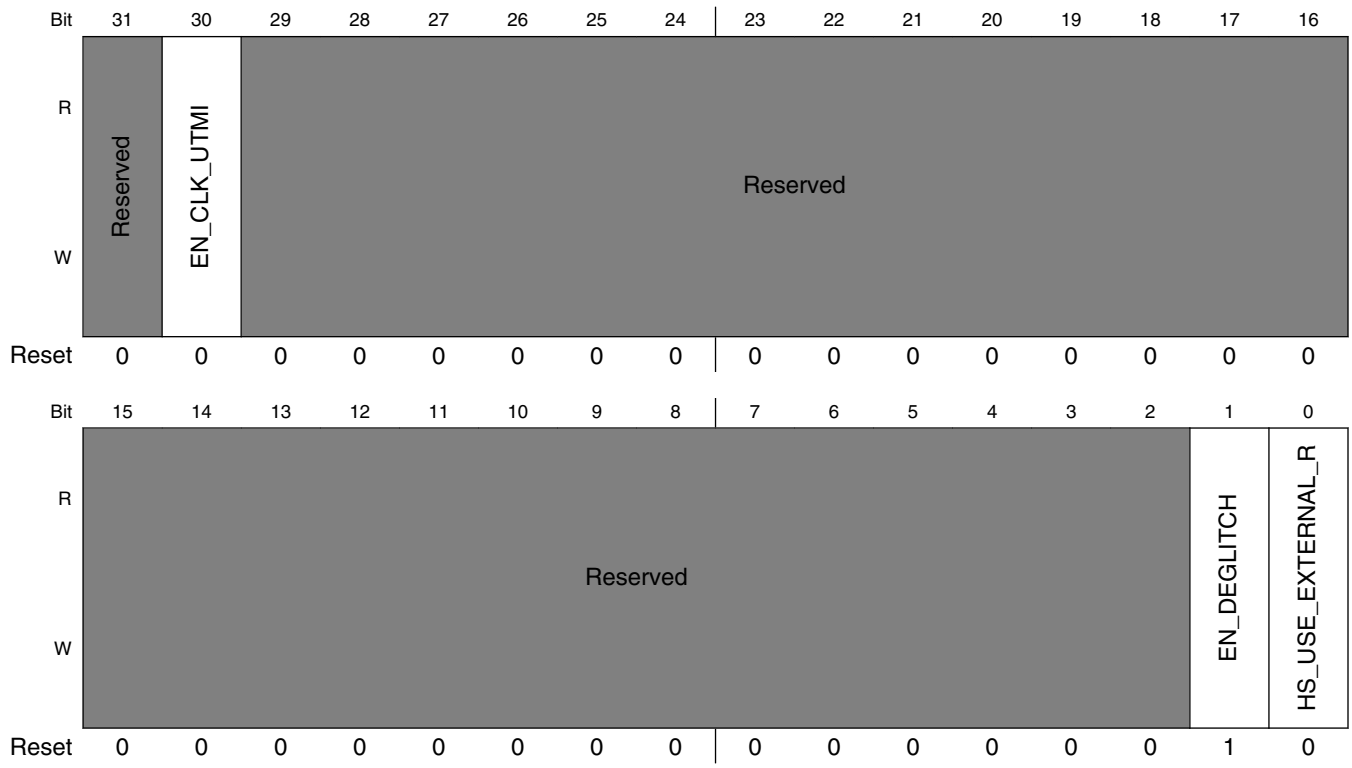
USB_ANALOG_USB2_LOOPBACK*n* field descriptions

Field	Description
31–1 -	This field is reserved. Reserved.
0 UTMI_ TESTSTART	Setting this bit can enable 1.5 kΩ pull-up resister on DP. **NOTE:** This bit can only be used as DCD detection, while it must be cleared in normal function.

43.5.12 USB Misc Register (USB_ANALOG_USB2_MISC*n*)

This register defines controls for USB.

Address: 400D_8000h base + 250h offset + (4d × i), where i=0d to 3d



USB_ANALOG_USB2_MISC*n* field descriptions

Field	Description
31 -	This field is reserved. Reserved.
30 EN_CLK_UTMI	Enables the clk to the UTMI block.
29–2 -	This field is reserved. Reserved.

Table continues on the next page...

USB_ANALOG_USB2_MISC*n* field descriptions (continued)

Field	Description
1 EN_DEGLITCH	Enable the deglitching circuit of the USB PLL output.
0 HS_USE_ EXTERNAL_R	Use external resistor to generate the current bias for the high speed transmitter. This bit should not be changed unless recommended by NXP.

43.5.13 Chip Silicon Version (USB_ANALOG_DIGPROG)

The DIGPROG register returns the digital program ID for the silicon.

Address: 400D_8000h base + 260h offset = 400D_8260h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SILICON_REVISION																															
W																																
Reset	0	0	0	0	0	0	0	0	0	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

USB_ANALOG_DIGPROG field descriptions

Field	Description
SILICON_ REVISION	Chip silicon revision 0x006C0000 Silicon revision 1.0

Chapter 44
Flexible Controller Area Network (FLEXCAN)

44.1 Chip-specific FLEXCAN information

Table 44-1. Reference links to related information

Topic	Related module(s)	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
Interrupts, DMA Events and XBAR Assignments	-	Interrupts, DMA Events and XBAR Assignments

44.2 Overview

The Flexible Controller Area Network (FLEXCAN) module is a communication controller implementing the CAN protocol according to the CAN 2.0B protocol specification.

The CAN protocol was primarily designed to be used as a vehicle serial data bus meeting the specific requirements of this field: real-time processing, reliable operation in the EMI environment of a vehicle, cost-effectiveness and required bandwidth. The FLEXCAN module is a full implementation of the CAN protocol specification, which supports both standard and extended message frames. 64 Message Buffers are supported by the Flexcan module.

44.2.1 Block Diagram

A general block diagram is shown in the figure below,which describes the main sub-blocks implemented in the FLEXCAN module, including the associated memory for storing Mailboxes, Rx Global Mask Registers, Rx Individual Mask Registers, Rx FIFO and Rx FIFO ID Filters.

Support for 64 Mailboxes and 6-deep Rx FIFO is provided. The functions of the sub-modules are described in subsequent sections.

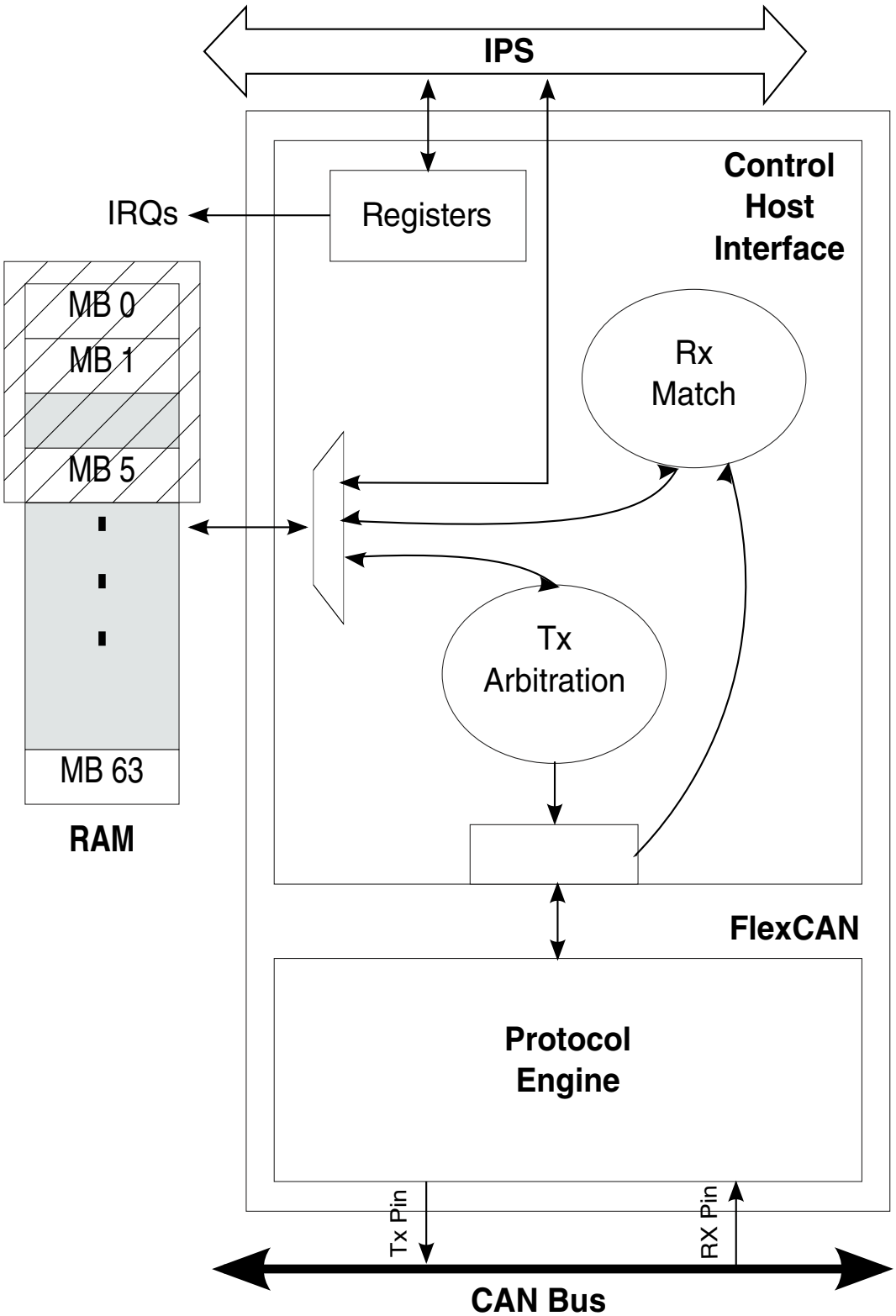


Figure 44-1. FLEXCAN Block Diagram

44.2.2 FLEXCAN Module Features

The FLEXCAN module includes these distinctive legacy features:

- Version 2.0B
  - Standard data and remote frames
  - Extended data and remote frames
  - Zero to eight bytes data length
  - Programmable bit rate up to 1 Mb/sec
  - Content-related addressing
- Flexible Mailboxes of eight bytes data length
- Each Mailbox is configurable as Rx or Tx, all supporting standard and extended messages
- Individual Rx Mask Registers per Mailbox
- Full featured Rx FIFO with storage capacity for 6 frames and internal pointer handling
- Transmission abort capability
- Powerful Rx FIFO ID filtering, capable of matching incoming IDs against either 128 extended, 256 standard or 512 partial (8 bits) IDs, with up to 32 individual masking capability
- 100% backwards compatibility with previous FLEXCAN version
- Unused structures space can be used as general purpose RAM space
- Listen only mode capability
- Programmable loop-back mode supporting self-test operation
- Programmable transmission priority scheme: lowest ID, lowest buffer number or highest priority
- Time Stamp based on 16-bit free-running timer
- Global network time, synchronized by a specific message
- Maskable interrupts independent of the transmission medium (an external transceiver is assumed)
- Short latency time due to an arbitration scheme for high-priority messages
- Low power modes, with programmable wake up on bus activity
- Configurable Glitch filter width to filter the noise on CAN bus when waking up
- Remote request frames may be handled automatically or by software.
- ID filter configuration in Normal Mode
- CAN bit time settings and configuration bits can only be written in Freeze Mode
- Tx mailbox status (Lowest priority buffer or empty buffer)
- SYNC bit status to inform that the module is synchronous with CAN bus
- CRC status for transmitted message
- Selectable priority between Mailboxes and Rx FIFO during matching process

44.2.3 Modes of Operation

The FLEXCAN module has four functional modes: Normal Mode (User and Supervisor), Freeze Mode, Listen-Only Mode and Loop-Back Mode. There are also two low power modes: Disable Mode and Stop Mode.

- Normal Mode (User or Supervisor):

In Normal Mode, the module operates receiving and/or transmitting message frames, errors are handled normally and all the CAN Protocol functions are enabled. User and Supervisor Modes differ in the access to some restricted control registers.

- Freeze Mode:

It is enabled when the FRZ bit in the MCR Register is asserted. If enabled, Freeze Mode is entered when the HALT bit in MCR is set or when Debug Mode is requested at MCU level and the FRZ_ACK bit in the MCR Register is asserted by the FlexCAN. In this mode, no transmission or reception of frames is done and synchronicity to the CAN bus is lost. See Freeze Mode for more information.

- Listen-Only Mode:

The module enters this mode when the LOM bit in the Control Register is asserted. In this mode, transmission is disabled, all error counters are frozen and the module operates in a CAN Error Passive mode. Only messages acknowledged by another CAN station will be received. If FLEXCAN detects a message that has not been acknowledged, it will flag a BIT0 error (without changing the REC), as if it was trying to acknowledge the message.

- Loop-Back Mode:

The module enters this mode when the LPB bit in the Control Register is asserted. In this mode, FLEXCAN performs an internal loop back that can be used for self test operation. The bit stream output of the transmitter is internally fed back to the receiver input. The FLEXCAN_RX input pin is ignored and the FLEXCAN_TX output goes to the recessive state (logic '1'). FLEXCAN behaves as it normally does when transmitting and treats its own transmitted message as a message received from a remote node. In this mode, FLEXCAN ignores the bit sent during the ACK slot in the CAN frame acknowledge field to ensure proper reception of its own message. Both transmit and receive interrupts are generated.

- Module Disable Mode:

This low power mode is entered when the MDIS bit in the MCR Register is asserted and the LPM_ACK is asserted by the FlexCAN. When disabled, the module requests to disable the clocks to the CAN Protocol Engine and Controller Host Interface sub-modules. Exit from this mode is done by negating the MDIS bit in the MCR Register. See Module Disable Mode for more information.

- Stop Mode:

This low power mode is entered when Stop Mode is requested at Arm level and the LPM_ACK bit in the MCR Register is asserted by the FlexCAN. When in Stop Mode, the module puts itself in an inactive state and then informs the Arm that the clocks can be shut down globally. Exit from this mode happens when the Stop Mode request is removed or when activity is detected on the CAN bus and the Self Wake Up mechanism is enabled. See Stop Mode for more information.

44.3 External Signals

The FLEXCAN module has two I/O signals.

Table 44-2. FLEXCAN External Signals

Signal	Description	Pad	Mode	Direction
FLEXCAN1_RX	FLEXCAN receive pin. This pin is the receive pin from the CAN bus transceiver. Dominant state is represented by logic level '0'. Recessive state is represented by logic level '1'.	GPIO_AD_B1_09 pin 23	ALT2	I
		GPIO_B0_03 pin 13	ALT2	
		GPIO_EMC_18	ALT3	
		GPIO_SD_B1_03	ALT4	
FLEXCAN1_TX	FLEXCAN transmit pin. This pin is the transmit pin to the CAN bus transceiver. Dominant state is represented by logic level '0'. Recessive state is represented by logic level '1'.	GPIO_AD_B1_08 pin 22	ALT2	O
		GPIO_B0_02 pin 11	ALT2	
		GPIO_EMC_17	ALT3	
		GPIO_SD_B1_02	ALT4	
FLEXCAN2_RX	FLEXCAN receive pin. This pin is the transmit pin to the CAN bus transceiver. Dominant state is represented by logic level '0'. Recessive state is represented by logic level '1'.	GPIO_AD_B0_03 pin 0	ALT0	I
		GPIO_EMC_10	ALT3	
		GPIO_AD_B0_15	ALT6	
		GPIO_B1_09	ALT6	

Table continues on the next page...

Table 44-2. FLEXCAN External Signals (continued)

FLEXCAN2_TX	FLEXCAN transmit pin. This pin is the receive pin from the CAN bus transceiver. Dominant state is represented by logic level '0'. Recessive state is represented by logic level '1'.	GPIO_AD_B0_02 pin 1	ALT0	O
		GPIO_EMC_09	ALT3	
		GPIO_B1_08	ALT6	
		GPIO_AD_B0_14	ALT6	
FLEXCAN3_RX	FLEXCAN receive pin. This pin is the transmit pin to the CAN bus transceiver. Dominant state is represented by logic level '0'. Recessive state is represented by logic level '1'.	GPIO_AD_B0_11	ALT8	I
		GPIO_AD_B0_15	ALT8	
		GPIO_EMC_37 pin 30	ALT9	
FLEXCAN3_TX	FLEXCAN transmit pin. This pin is the receive pin from the CAN bus transceiver. Dominant state is represented by logic level '0'. Recessive state is represented by logic level '1'.	GPIO_AD_B0_10	ALT8	O
		GPIO_AD_B0_14	ALT8	
		GPIO_EMC_36 pin 31	ALT9	

44.4 Clocks

The table found here describes the clock sources for FLEXCAN.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 44-3. FLEXCAN Clocks

Clock name	Clock Root	Description
ipg_clk	ipg_clk_root	Peripheral clock
ipg_clk_chi	ipg_clk_root	CHI clock
ipg_clk_pe	can_clk_root	Protocol Engine clock
ipg_clk_pe_nogate	can_clk_root	Protocol Engine clock (no gating)
ipg_clk_s	ipg_clk_root	Peripheral access clock
mem_ram_CLK	ipg_clk_root	RAM clock

44.5 Message Buffer Structure

Message Buffer Address: Base + 0x0080-0x047C.

The Message Buffer structure used by the FLEXCAN module is represented in the following table.

Both Extended and Standard Frames (29-bit Identifier and 11-bit Identifier, respectively) used in the CAN specification are represented.

Each individual Message buffer is formed by 16 bytes.

Table 44-4. Message Buffer Structure

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0					CODE					SRR	IDE	RTR	DLC				TIME STAMP															
0x4	PRIO				ID Standard												ID Extended															
0x8	DATA BYTE 0								DATA BYTE 1								DATA BYTE 2								DATA BYTE 3							
0xC	DATA BYTE 4								DATA BYTE 5								DATA BYTE 6								DATA BYTE 7							

CODE - Message Buffer Code

This 4-bit field can be accessed (read or write) by the CPU and by the FLEXCAN module itself, as part of the message buffer matching and arbitration process. The encoding is shown in the following tables. See Functional Description for additional information.

Table 44-5. Message Buffer Code for Rx buffers

CODE Description	Rx Code BEFORE receive New Frame	SRV[1]	Rx Code AFTER successful reception[2]	RRS[3]	Comment
0b0000: INACTIVE- MB is not active.	INACTIVE	-	-	-	MB does not participate in the matching process.
0b0100: EMPTY - MB is active and empty.	EMPTY	-	FULL	-	When a frame is received successfully (after move-in process. Refer to Move-in for details), the CODE field is automatically updated to FULL.
0b0010: FULL - MB is full.	FULL	Yes	FULL	-	The act of reading the C/S word followed by unlocking the MB (SRV) does not make the code return to EMPTY. It remains FULL. If a

Table continues on the next page...

Table 44-5. Message Buffer Code for Rx buffers (continued)

CODE Description	Rx Code BEFORE receive New Frame	SRV[1]	Rx Code AFTER successful reception[2]	RRS[3]	Comment
					new frame is moved to the MB after the MB was serviced, the code still remains FULL. Refer to Matching Process for matching details related to FULL code.
		No	OVERRUN	-	If the MB is FULL and a new frame is moved to this MB before the CPU services it, the CODE field is automatically updated to OVERRUN. Refer to Matching Process for details about overrun behavior.
0b0110: OVERRUN - MB is being overwritten into a full buffer.	OVERRUN	Yes	FULL	-	If the CODE field indicates OVERRUN and CPU has serviced the MB, when a new frame is moved to the MB, the code returns to FULL.
		No	OVERRUN	-	If the CODE field already indicates OVERRUN, and another new frame must be moved, the MB will be overwritten again, and the code will remain OVERRUN. Refer to Matching Process for details about overrun behavior.
0b1010: RANSWER[4] - A frame was configured to recognize a Remote Request Frame and transmit a Response Frame in return.	RANSWER	-	TANSWER(0b1110)	0	A Remote Answer was configured to recognize a remote request frame received, after that a MB is set to transmit a response frame. The code is automatically changed to TANSWER (0b1110). Refer to Matching Process for details. If CTRL2[RRS] is negated, transmit a response frame whenever a remote request frame with the same ID is received.
			-	1	This code is ignored during matching and arbitration process. Refer to Matching Process for details.
CODE[0]=1b1: BUSY - FlexCAN is updating the contents of the MB. The CPU must not access the MB.	BUSY[5]	-	FULL	-	Indicates that the MB is being updated, it will be negated automatically and does not interfere on the next CODE.
			OVERRUN	-	

1. SRV: Serviced MB. MB was read and unlocked by reading TIMER or other MB.
2. A frame is considered successful reception after the frame to be moved to MB (move-in process). Refer to Move-in for details)
3. Remote Request Stored bit from CTRL2 register. Refer to CTRL2 for details.
4. Code 4'b1010 is not considered as a Tx and a MB with this code should not to be aborted.
5. Note that for Tx MBs, the BUSY bit should be ignored upon read, except when AEN bit is set in the MCR register. If this bit is asserted, the corresponding MB does not participate in the matching process.

Table 44-6. Message Buffer Code for Tx buffers

CODE Description	Tx Code BEFORE tx frame	MB RTR	Tx Code AFTER successful transmission	Comment
0b1000: INACTIVE - MB is not active	INACTIVE	-	-	MB does not participate in the arbitration process.
0b1001: ABORT - MB is aborted	ABORT	-	-	MB does not participate in the arbitration process.
0b1100: DATA - MB is a Tx Data Frame (MB RTR must be 0)	DATA	0	INACTIVE	Transmit data frame unconditionally once. After transmission, the MB automatically returns to the INACTIVE state.
0b1100: REMOTE - MB is a Tx Remote Request Frame (MB RTR must be 1)	REMOTE	1	EMPTY	Transmit remote request frame unconditionally once. After transmission, the MB automatically becomes an Rx Empty MB with the same ID.
0b1110: TANSWER - MB is a Tx Response Frame from an incoming Remote Request Frame	TANSWER	-	RANSWER	This is an intermediate code that is automatically written to the MB by the CHI as a result of match to a remote request frame. The remote response frame will be transmitted unconditionally once and then the code will automatically return to RANSWER (0b1010). The CPU can also write this code with the same effect. The remote response frame can be either a data frame or another remote request frame depending on the RTR bit value. Refer to Matching Process and Arbitration process for details.

SRR - Substitute Remote Request

Fixed recessive bit, used only in extended format. It must be set to '1' by the user for transmission (Tx Buffers) and will be stored with the value received on the CAN bus for Rx receiving buffers. It can be received as either recessive or dominant. If FLEXCAN receives this bit as dominant, then it is interpreted as arbitration loss.

1= Recessive value is compulsory for transmission in Extended Format frames

0= Dominant is not a valid value for transmission in Extended Format frames

IDE - ID Extended Bit

This bit identifies whether the frame format is standard or extended. It is also used as part of the reception filter.

1= Frame format is extended

0= Frame format is standard

RTR - Remote Transmission Request

This bit affects the behavior of Remote Frames and is part of the reception filter. Refer to the tables above and RRS bit in Control 2 Register (FLEXCAN_CTRL2) for additional details.

If FLEXCAN transmits this bit as '1' (recessive) and receives it as '0' (dominant), it is interpreted as arbitration loss. If this bit is transmitted as '0' (dominant), then if it is received as '1' (recessive), the FLEXCAN module treats it as bit error. If the value received matches the value transmitted, it is considered as a successful bit transmission.

1= Indicates the current MB has a Remote Frame to be transmitted if MB is Tx. If the MB is Rx then incoming Remote Request Frames may be stored.

0= Indicates the current MB has a Data Frame to be transmitted. In Rx MB it may be considered in matching processes.

DLC - Length of Data in Bytes

This 4-bit field is the length (in bytes) of the Rx or Tx data, which is located in offset 0x08 through 0x0F of the MB space (see the first table above). In reception, this field is written by the FLEXCAN module, copied from the DLC (Data Length Code) field of the received frame. In transmission, this field is written by the Arm and corresponds to the DLC field value of the frame to be transmitted. When RTR=1, the Frame to be transmitted is a Remote Frame and does not include the data field, regardless of the Length field. The DLC field indicates which DATA BYTEs are valid as shown in the table below.

TIME STAMP - Free-Running Counter Time Stamp

This 16-bit field is a copy of the Free-Running Timer, captured for Tx and Rx frames at the time when the beginning of the Identifier field appears on the CAN bus.

PRIO - Local priority

This 3-bit field is only used when MCR[LPRIO_EN] bit is asserted and it only makes sense for Tx mailboxes. These bits are not transmitted. They are appended to the regular ID to define the transmission priority. See Arbitration process.

ID - Frame Identifier

In Standard Frame format, only the 11 most significant bits (28 to 18) are used for frame identification in both receive and transmit cases. The 18 least significant bits are ignored. In Extended Frame format, all bits are used for frame identification in both receive and transmit cases.

DATA BYTE 0-7 - Data Field

Up to eight bytes can be used for a data frame.

For Rx frames, the data is stored as it is received from the CAN bus. DATA BYTE (n) is valid only if *n* is less than DLC as shown in the table below.

For Tx frames, the CPU prepares the data field to be transmitted within the frame.

Table 44-7. DATA BYTEs validity

DLC	Valid DATA BYTEs
0	none
1	DATA BYTE 0
2	DATA BYTE 0-1
3	DATA BYTE 0-2
4	DATA BYTE 0-3
5	DATA BYTE 0-4
6	DATA BYTE 0-5
7	DATA BYTE 0-6
8	DATA BYTE 0-7

44.6 Rx FIFO Structure

When the MCR[RFEN] bit is set, the memory area from 0x80 to 0xDC (which is normally occupied by MBs 0 to 5) is used by the reception FIFO engine.

The region 0x80-0x8C contains the output of the FIFO which must be read by the CPU as a Message Buffer. This output contains the oldest message received and not read yet. The region 0x90-0xDC is reserved for internal use of the FIFO engine.

An additional memory area, that starts at 0xE0 and may extend up to 0x2DC (normally occupied by MBs 6 up to 37) depending on the CTRL2[RFFN] field setting, contains the ID Filter Table (configurable from 8 to 128 memory positions) that specifies filtering criteria for accepting frames into the FIFO.Table 44-8 shows the Rx FIFO data structure.

Each ID Filter Table Element occupies an entire 32-bit word and can be compounded by one, two or four Identifier Acceptance Filters (IDAF) depending on the MCR[IDAM] field setting. Table 44-9, Table 44-10 and Table 44-11 show the IDAF indexation. Table 44-12 show the three different formats that the IDAF can assume, depending on the MCR[IDAM] field setting. Note that all elements of the table must have the same format. See Rx FIFO for more information.

Out of reset, the ID Filter Table flexible memory area defaults to 0xE0 and only extends to 0xFC, which corresponds to MBs 6 to 7 for RFFN=0.

Table 44-8. Rx FIFO Structure

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x80										SRR	IDE	RTR	DLC				TIME STAMP															
0x84										ID Standard							ID Extended															
0x88	Data Byte 0									Data Byte 1							Data Byte 2								Data Byte 3							
0x8C	Data Byte 4									Data Byte 5							Data Byte 6								Data Byte 7							
0x90 to 0xDC	Reserved																															
0xE0	ID Filter Table Element 0																															
0xE4	ID Filter Table Element 1																															
0xE8 to 0x2D4	ID Filter Table Elements 2 through 125																															
0x2D8	ID Filter Table Element 126																															
0x2DC	ID Filter Table Element 127																															

Table 44-9. Position of ID Filter Table Elements in format A

Element	31																													0
0	IDAF0																													
1	IDAF1																													
2 through 125	Identifier Acceptance Filter 2 through 125																													
126	IDAF126																													
127	IDAF127																													

Table 44-10. Position of ID Filter Table Elements in format B

Element	31															16	15													0
0	IDAF0																IDAF1													
1	IDAF2																IDAF3													
2 through 125	Identifier Acceptance Filter 4 through 251																													
126	IDAF252																IDAF253													

Table continues on the next page...

Table 44-10. Position of ID Filter Table Elements in format B (continued)

127	IDAF254	IDAF255

Table 44-11. Position of ID Filter Table Elements in format C

Element	31 24	23 16	15 8	7 0
0	IDAF0	IDAF1	IDAF2	IDAF3
1	IDAF4	IDAF5	IDAF6	IDAF7
2 through 125	Identifier Acceptance Filter 8 through 503			
126	IDAF504	IDAF505	IDAF506	IDAF507
127	IDAF508	IDAF509	IDAF510	IDAF511

Table 44-12. Identifier Acceptance Filter Format A,B and C

Format	31	30	29 28 27 26 25 24	23 22 21 20 19 18 17 16	15	14	13 12 11 10 9 8	7 6 5 4 3 2 1 0
A	RTR	IDE	RXIDA (Standard = 29-19, Extended = 29-1)					
B	RTR	IDE	RXIDB_0 (Standard = 29-19, Extended = 29-16)		RTR	IDE	RXIDB_1 (Standard = 13-3, Extended = 13-0)	
C	RXIDC_0 (Std/Ext = 31-24)			RXIDC_1 (Std/Ext = 23-16)	RXIDC_2 (Std/Ext = 15-8)			RXIDC_3 (Std/Ext = 7-0)

RTR - Remote Frame

This bit specifies whether Remote Request Frames are accepted into the FIFO if they match the target ID in Formats A and B. If Format C is chosen the acceptance does not depend on whether the frame is a Remote Request Frame or not.

1= Remote Frames can be accepted and data frames are rejected

0= Remote Frames are rejected and data frames can be accepted

IDE - Extended Frame

Specifies if either Extended or Standard Format frames are accepted into the FIFO if they match the target ID in Formats A and B. If Format C is chosen the acceptance does not depend on whether the frame is of the Extended or Standard Format.

1= Extended frames can be accepted and standard frames are rejected

0= Extended frames are rejected and standard frames can be accepted

RXIDA - Rx Frame Identifier (Format A)

Specifies an ID to be used as acceptance criteria for the FIFO. In the Standard Format (IDAF's or incoming frame's IDE bit is negated), only the 11 most significant bits (29 to 19) are used for frame identification. In the Extended Format (both IDAF's and incoming frame's IDE are asserted), all bits are used.

RXIDB_0, RXIDB_1 - Rx Frame Identifier (Format B)

Specifies an ID to be used as acceptance criteria for the FIFO. In the Standard Format (IDAF's or incoming frame's IDE bit is negated), the 11 most significant bits (29 to 19 and 13 to 3) are used for frame identification. In the Extended Format (both IDAF's and incoming frame's IDE are asserted), all 14 bits of the field are compared with the 14 most significant bits of the Identifier of the incoming frame. The 15 least significant bits of the Identifier of an incoming Extended Format frame do not affect the acceptance.

RXIDC_0, RXIDC_1, RXIDC_2, RXIDC_3 - Rx Frame Identifier (Format C)

Specifies an ID to be used as acceptance criteria for the FIFO. In both Standard Format and Extended Format, all 8 bits of the field are compared to the 8 most significant bits of the Identifier of the incoming frame. The 3 least significant bits of the Identifier of an incoming Standard Format frame and the 21 least significant bits of the Identifier of an incoming Extended Format frame do not affect the acceptance.

44.7 Functional Description

This section provides a complete functional description of the block.

44.7.1 Functional Overview

The FLEXCAN module is a CAN protocol engine with a very flexible mailbox system for transmitting and receiving CAN frames. The mailbox system consists of a set of 64 Message Buffers (MB) that store configuration and control data, time stamp, message ID and data (see Message Buffer Structure). The memory corresponding to the first 38 MBs can be configured to support a FIFO reception scheme with a powerful ID filtering mechanism, capable of checking incoming frames against a table of IDs (up to 128 extended IDs or 256 standard IDs or 512 8-bit ID slices), with individual mask register for up to 32 ID Filter Table elements. Simultaneous reception through FIFO and mailbox is supported. For mailbox reception, a *matching* algorithm makes it possible to store received frames only into MBs that have the same ID. A masking scheme makes it possible to match the ID programmed on the MB with a range of Identifiers on received

CAN frames. For transmission, an *arbitration* algorithm decides the prioritization of MBs to be transmitted based on the message ID (optionally augmented by 3 local priority bits) or the MB ordering.

Before proceeding with the functional description, an important concept must be explained. A Message Buffer is said to be "active" at a given time if it can participate in both the Matching and Arbitration processes. An Rx MB with a 0b0000 code is inactive (refer to Table 44-5). Similarly, a Tx MB with a 0b1000 or 0b1001 code is also inactive (refer to Table 44-6).

44.7.2 Transmit Process

In order to transmit a CAN frame, the CPU must prepare a Message Buffer for transmission by executing the procedure found here.

1. Check if the respective interruption bit is set and clear it.
2. If the MB is active (transmission pending), write the ABORT code (0b1001) to the CODE field of the Control and Status word to request an abortion of the transmission. Wait for the corresponding IFLAG to be asserted by polling the IFLAG register or by the interrupt request if enabled by the respective IMASK. Then read back the CODE field to check if the transmission was aborted or transmitted (see Transmission Abort Mechanism). If backwards compatibility is desired (MCR[AEN] bit negated), just write the INACTIVE code (0b1000) to the CODE field to inactivate the MB but then the pending frame may be transmitted without notification (see Message Buffer Inactivation).
3. Write the ID word.
4. Write the data bytes.
5. Write the DLC, Control and Code fields of the Control and Status word to activate the MB.

Once the MB is activated, it will participate into the arbitration process and eventually be transmitted according to its priority.

At the end of the successful transmission, the value of the Free Running Timer at the time of the second bit of frame's Identifier field is written into the MB's Time Stamp field, the CODE field in the Control and Status word is updated, the CRC Register is updated, a status flag is set in the Interrupt Flag Register and an interrupt is generated if allowed by the corresponding Interrupt Mask Register bit. The new CODE field after transmission depends on the code that was used to activate the MB in step four (see Table 44-5 and Table 44-6 in Message Buffer Structure).

When the Abort feature is enabled (MCR[AEN] bit is asserted), after the Interrupt Flag is asserted for a Mailbox configured as transmit buffer, the Mailbox is blocked, therefore the CPU is not able to update it until it negates the Interrupt Flag. It means that the CPU must clear the corresponding IFLAG before starting to prepare this MB for a new transmission or reception.

44.7.3 Arbitration process

The arbitration process scans the Mailboxes searching the Tx one that holds the message to be sent in the next opportunity. This Mailbox is called the *arbitration winner*.

The scan starts from the lowest Mailbox number and runs toward the higher ones.

The arbitration process is triggered in the following events:

- From the CRC field of the CAN frame. The start point depends on the CTRL2[TASD] field value. See Control 2 Register (FLEXCAN_CTRL2) for details.
- During the error delimiter field of the CAN frame
- During the Overload Delimiter field of a CAN frame.
- When the winner is inactivated and the CAN bus has still not reached the first bit of the Intermission field.
- When Arm write to the C/S word of a winner MB and the CAN bus has still not reached the first bit of the Intermission field.
- When CHI is in Idle state and Arm writes to the C/S word of any MB.
- When FlexCAN exits Bus Off state
- Upon leaving Freeze Mode or Low Power Mode

If the arbitration process does not manage to evaluate all Mailboxes before the CAN bus has reached the first bit of the Intermission field the temporary arbitration winner is invalidated and the FlexCAN will not compete for the CAN bus in the next opportunity.

The arbitration process selects the winner among the active Tx Mailboxes at the end of the scan according to both CTRL1[LBUF] and MCR[LPRIO_EN] bits settings.

44.7.3.1 Lowest Mailbox number first

If CTRL1[LBUF] bit is asserted the first (lowest number) active Tx Mailbox found is the arbitration winner. MCR[LPRIO_EN] bit has no effect when CTRL1[LBUF] is asserted.

44.7.3.2 Highest Mailbox priority first

If CTRL1[LBUF] bit is negated then the arbitration process searches the active Tx Mailbox with the highest priority, and this Mailbox would have a higher probability to win the arbitration on CAN bus.

The sequence of bits considered for this arbitration is called the *arbitration value* of the Mailbox. The highest priority Tx Mailbox is the one that has the least arbitration value among all Tx Mailboxes.

If two or more Mailboxes have equivalent arbitration values the lowest Mailbox number is the arbitration winner.

The composition of the arbitration value depends on MCR[LPRIO_EN] bit setting.

44.7.3.2.1 Local Priority disabled

If MCR[LPRIO_EN] bit is negated the arbitration value is built in the exact sequence of bits as they would be transmitted in a CAN frame (see Table 44-13) in such a way that the Local Priority is disabled.

Table 44-13. Composition of the arbitration value when Local Priority is disabled

Format	Mailbox Arbitration Value (32 bits)				
Standard (IDE = 0)	Standard ID (11 bits)	RTR (1 bit)	IDE (1 bit)	- (18 bits)	- (1 bit)
Extended (IDE = 1)	Extended ID[28:18] (11 bits)	SRR (1 bit)	IDE (1 bit)	Extended ID[17:0] (18 bits)	RTR (1 bit)

44.7.3.2.2 Local Priority enabled

If Local Priority is desired MCR[LPRIO_EN] must be asserted.

In this case the Mailbox PRIO field is included at the very left of the arbitration value (see the table below).

Table 44-14. Composition of the arbitration value when Local Priority is enabled

Format	Mailbox Arbitration Value (35 bits)					
Standard (IDE = 0)	PRIO (3 bits)	Standard ID (11 bits)	RTR (1 bit)	IDE (1 bit)	- (18 bits)	- (1 bit)
Extended (IDE = 1)	PRIO (3 bits)	Extended ID[28:18](11 bits)	SRR (1 bit)	IDE (1 bit)	Extended ID[17:0] (18 bits)	RTR (1 bit)

As the PRIO field is the most significant part of the arbitration value Mailboxes with low PRIO values have higher priority than Mailboxes with high PRIO values regardless the rest of their arbitration values.

Note that the PRIO field is not part of the frame on the CAN bus. Its purpose is only to affect the internal arbitration process.

Once the arbitration winner is found, its content is copied to a hidden auxiliary MB called Tx Serial Message Buffer (Tx SMB), which has the same structure as a normal MB but is not user accessible. This operation is called "move-out" and after it is done, write access to the corresponding MB is blocked (if the AEN bit in MCR is asserted). The write access is released in the following events:

- After the MB is transmitted
- FlexCAN enters in Freeze Mode or Bus Off
- FlexCAN loses the bus arbitration or there is an error during the transmission

At the first opportunity window on the CAN bus, the message on the Tx SMB is transmitted according to the CAN protocol rules. FlexCAN transmits up to eight data bytes, even if the DLC (Data Length Code) field value is greater than that.

Arbitration process can be triggered in the following situations:

- During Rx and Tx frames from CAN CRC field to end of frame. Arbitration start point depends on instantiation parameters NUMBER_OF_MB and TASD. Additionally, TASD value may be changed (see Control 2 Register (FLEXCAN_CTRL2)) to optimize the arbitration start point.
- During CAN Bus Off state from TX_ERR_CNT=124 to 128. Arbitration start point depends on instantiation parameters NUMBER_OF_MB and TASD. Additionally, TASD value may be changed(see Control 2 Register (FLEXCAN_CTRL2)) to optimize the arbitration start point.
- During C/S write by CPU in Bus Idle. First C/S write starts arbitration process and a second C/S write during this same arbitration restarts the process. If other C/S writes are performed, Tx arbitration process is pending. If there is no arbitration winner after arbitration process has finished, then TX arbitration machine begins a new arbitration process.
- Arbitration winner deactivation during a valid arbitration window.
- Upon Leave Freeze Mode. If there is a re-synchronization during WaitForBusIdle arbitration process is restarted.

Arbitration process stops in the following situation:

- All Mailboxes were scanned.
- A Tx active Mailbox is found in case of Lowest Buffer feature enabled.
- Arbitration winner inactivation or abort during any arbitration process.

- There was not enough time to finish Tx arbitration process. For instance, a deactivation was performed near the end of frame). In this case arbitration process is pending.
- Error or Overload flag in the bus.
- Low Power or Freeze Mode request in Idle state

Arbitration is considered pending as described below:

- It was not possible to finish arbitration process in time.
- C/S write during arbitration if write is performed in a MB which number is lower than the Tx arbitration pointer.
- Any C/S write if there is no Tx Arbitration process in progress.
- Rx Match has just updated a Rx Code to Tx Code.
- Entering Bus off state.

C/S write during arbitration has the following effect:

- If C/S write is performed in the arbitration winner, a new process is restarted immediately.
- C/S write during arbitration if write is performed in a MB which number is higher than the Tx arbitration pointer.

44.7.4 Receive Process

To be able to receive CAN frames into a Mailbox, the CPU must prepare it for reception by executing the steps listed here.

1. If the Mailbox is active (either Tx or Rx) inactivate the Mailbox (see Message Buffer Inactivation), preferably with a *safe inactivation* (see Transmission Abort Mechanism);
2. Write the ID word;
3. Write the EMPTY code (0b0100) to the CODE field of the Control and Status word to activate the Mailbox.

Once the Mailbox is activated in the third step, it will be able to receive frames that match the programmed filter. At the end of a successful reception, the Mailbox is updated by the *move-in process* (see Move-in) as follows:

1. The received Data field (8 bytes at most) is stored;
2. The received Identifier field is stored;
3. The value of the Free Running Timer at the time of the second bit of frame's Identifier field is written into the Mailbox Time Stamp field;
4. The received SRR, IDE, RTR and DLC fields are stored;

5. The CODE field in the Control and Status word is updated. (see Table 44-5 and Table 44-6 in Section Message Buffer Structure)
6. A status flag is set in the Interrupt Flag Register and an interrupt is generated if allowed by the corresponding Interrupt Mask Register bit.

The recommended way for the CPU servicing (read) the frame received in a Mailbox is using the following procedure:

1. Read the Control and Status word of that Mailbox;
2. Check if the BUSY bit is deasserted, indicating that the Mailbox is locked. Repeat step 1) while it is asserted. See Message Buffer Lock Mechanism;
3. Read the contents of the Mailbox. Once Mailbox is locked now, its contents won't be modified by FlexCAN Move-in processes. See Move-in;
4. Acknowledge the proper flag at IFLAG registers;
5. Read the Free Running Timer. It is optional but recommended to unlock Mailbox as soon as possible and make it available for reception.

The CPU should synchronize to frame reception by the status flag bit for the specific Mailbox in one of the IFLAG Registers and not by the CODE field of that Mailbox. Polling the CODE field does not work because once a frame was received and the CPU services the Mailbox (by reading the C/S word followed by unlocking the Mailbox), the CODE field will not return to EMPTY. It will remain FULL. If the CPU tries to workaround this behavior by writing to the C/S word to force an EMPTY code after reading the Mailbox without a prior *safe inactivation*, a newly received message matching the filter of that Mailbox may be lost.

In summary: never do polling by reading directly the C/S word of the Mailboxes. Instead, read the IFLAG registers.

Note that the received frame's Identifier field is always stored in the matching Mailbox, thus the contents of the ID field in a Mailbox may change if the match was due to masking. Note also that FlexCAN does receive frames transmitted by itself if there exists a matching Rx Mailbox, provided the MCR[SRX_DIS] bit is not asserted. If MCR[SRX_DIS] bit is asserted, FlexCAN will not store messages transmitted by itself in any MB, even if it contains a matching MB, and no interrupt flag or interrupt signal will be generated due to the frame reception.

To be able to receive CAN messages through the Rx FIFO, the CPU must enable and configure the Rx FIFO during Freeze Mode(see Rx FIFO). Upon receiving the Frames Available in Rx FIFO interrupt(see Interrupt Masks 1 Register (FLEXCAN_IMASK1), bit IFLAG[BUF5I] - Frames available in Rx FIFO), the CPU should service the received frame using the following procedure:

1. Read the Control and Status word (optional - needed only if a mask was used for IDE and RTR bits);
2. Read the ID field (optional - needed only if a mask was used);
3. Read the Data field;
4. Read the RXFIR register (optional);
5. Clear the Frames Available in Rx FIFO interrupt by writing 1 to IFLAG[BUF5I] bit (mandatory - releases the MB and allows the CPU to read the next Rx FIFO entry)

44.7.5 Matching Process

The matching process scans the MB memory looking for Rx MBs programmed with the same ID as the one received from the CAN bus. If the FIFO is enabled, the priority of scanning can be selected between Mailboxes and FIFO filters. In any case, the matching starts from the lowest number Message Buffer toward the higher ones. If no match is found within the first structure then the other is scanned subsequently. In the event that the FIFO is full, the matching algorithm will always look for a matching MB outside the FIFO region.

As the frame is being received, it is stored in a hidden auxiliary MB called Rx Serial Message Buffer (Rx SMB).

The matching process start point depends on the following conditions:

- if the received frame is a remote frame, the start point is the CRC field of the frame;
- if the received frame is a data frame with DLC field equal to zero, the start point is the CRC field of the frame;
- if the received frame is a data frame with DLC field different than zero, the start point is the DATA field of the frame;

If a matching ID is found in the FIFO table or in one of the Mailboxes, the contents of the SMB will be transferred to the FIFO or to the matched Mailbox by the move-in process. If any CAN protocol error is detected then no match results will be transferred to the FIFO or to the matched Mailbox at the end of reception.

The matching process scans all matching elements of both Rx FIFO (if enabled) and active Rx Mailboxes (CODE is EMPTY, FULL, OVERRUN or RANSWER) in search of a successful comparison with the matching elements of the Rx SMB that is receiving the frame on the CAN bus. The SMB has the same structure of a Mailbox. The reception structures (Rx FIFO or Mailboxes) associated with the matching elements that had a successful comparison are the *matched structures*. The *matching winner* is selected at the end of the scan among those matched structures and depends on conditions described ahead. Refer to the following table for details.

Table 44-15. Matching architecture

Structure	SMB[RTR]	CTRL2[RRS]	CTRL2[EACEN]	MB[IDE]	MB[RTR]	MB[ID][1]	MB[CODE]
Mailbox	0	-	0	cmp[2]	no_cmp[3]	cmp_msk[4]	EMPTY or FULL or OVERRUN
Mailbox	0	-	1	cmp_msk	cmp_msk	cmp_msk	EMPTY or FULL or OVERRUN
Mailbox	1	0	-	cmp	no_cmp	cmp	RANSWER
Mailbox	1	1	0	cmp	no_cmp	cmp_msk	EMPTY or FULL or OVERRUN
Mailbox	1	1	1	cmp_msk	cmp_msk	cmp_msk	EMPTY or FULL or OVERRUN
FIFO[5]	-	-	-	cmp_msk	cmp_msk	cmp_msk	-

1. For Mailbox structure, If SMB[IDE] is asserted, the ID is 29 bits (ID Standard + ID Extended). In case of SMB[IDE] to be negated, the ID is only 11 bits (ID Standard). Please, refer to Message Buffer Structure for ID details. For FIFO structure, the ID depends on IDAM. Please, refer to Rx FIFO Structure for IDAM details.
2. cmp: Compares the SMB contents with the MB contents regardless the masks.
3. no_cmp: The SMB contents are not compared with the MB contents.
4. cmp_msk: Compares the SMB contents with MB contents taking into account the masks.
5. SMB[IDE] and SMB[RTR] are not taken into account when IDAM is type C.

A reception structure is *free-to-receive* when any of the following conditions is satisfied:

- the CODE field of the Mailbox is EMPTY;
- the CODE field of the Mailbox is either FULL or OVERRUN and it has already been serviced (the C/S word was read by the Arm and unlocked as described in Message Buffer Lock Mechanism);
- the CODE field of the Mailbox is either FULL or OVERRUN and an inactivation is performed. (see Message Buffer Inactivation)
- the Rx FIFO is not full.

The scan order for Mailboxes and Rx FIFO is from the matching element with lowest number to the higher ones.

The matching winner search for Mailboxes is affected by the MCR[IRMQ] bit. If it is negated the matching winner is the first matched Mailbox regardless if it is free-to-receive or not . If it is asserted, the matching winner is selected according to the priority below:

1. the first free-to-receive matched Mailbox;
2. the last non free-to-receive matched Mailbox.

It is possible to select the priority of scan between Mailboxes and Rx FIFO by the CTRL2[MRP] bit.

If the selected priority is Rx FIFO first:

- if the Rx FIFO is a matched structure and is free-to-receive then the Rx FIFO is the matching winner regardless of the scan for Mailboxes;
- otherwise (the Rx FIFO is not a matched structure or is not free-to-receive), then the matching winner is searched among Mailboxes as described above.

If the selected priority is Mailboxes first:

- if a free-to-receive matched Mailbox is found, it is the matching winner regardless the scan for Rx FIFO;
- if no matched Mailbox is found, then the matching winner is searched in the scan for the Rx FIFO;
- if both conditions above are not satisfied and a non free-to-receive matched Mailbox is found then the matching winner determination is conditioned by the MCR[IRMQ] bit:
  - if MCR[IRMQ] bit is negated the matching winner is the first matched Mailbox;
  - if MCR[IRMQ] bit is asserted the matching winner is the Rx FIFO if it is a free-to-receive matched structure, otherwise the matching winner is the last non free-to-receive matched Mailbox.

Please, refer to the table below for a summary of matching possibilities.

If a non-safe Mailbox inactivation (see Message Buffer Inactivation) occurs during matching process and the Mailbox inactivated is the temporary matching winner then the temporary matching winner is invalidated. The matching elements scan is not stopped nor restarted, it continues normally. The consequence is that the current matching process works as if the matching elements compared before the inactivation did not exist, therefore a message may be lost.

Suppose, for example, that the FIFO is disabled, IRMQ is enabled and there are two MBs with the same ID, and FlexCAN starts receiving messages with that ID. Let us say that these MBs are the second and the fifth in the array. When the first message arrives, the matching algorithm will find the first match in MB number 2. The code of this MB is EMPTY, so the message is stored there. When the second message arrives, the matching algorithm will find MB number 2 again, but it is not "free-to-receive", so it will keep looking and find MB number 5 and store the message there. If yet another message with the same ID arrives, the matching algorithm finds out that there are no matching MBs that are "free-to-receive", so it decides to overwrite the last matched MB, which is number 5. In doing so, it sets the CODE field of the MB to indicate OVERRUN.

Table 44-16. Matching Possibilities and Resulting Reception Structures

RFEN	IRMQ	MRP	Matched in MB	Matched in FIFO	Reception Structure	Description
No FIFO, only MB, match is always MB first						
0	0	X[1]	None[2]	-[3]	None	Frame lost by no match
0	0	X	Free[4]	-	FirstMB	
0	1	X	None	-	None	Frame lost by no match
0	1	X	Free	-	FirstMB	
0	1	X	NotFree	-	LastMB	Overrun
FIFO enabled, no match in FIFO is as if FIFO does not exist						
1	0	X	None	None[5]	None	Frame lost by no match
1	0	X	Free	None	FirstMB	
1	1	X	None	None	None	Frame lost by no match
1	1	X	Free	None	FirstMB	
1	1	X	NotFree	None	LastMB	Overrun
FIFO enabled, Queue disabled						
1	0	0	X	NotFull[6]	FIFO	
1	0	0	None	Full[7]	None	Frame lost by FIFO full (FIFO Overflow)
1	0	0	Free	Full	FirstMB	
1	0	0	NotFree	Full	FirstMB	
1	0	1	None	NotFull	FIFO	
1	0	1	None	Full	None	Frame lost by FIFO full (FIFO Overflow)
1	0	1	Free	X	FirstMB	
1	0	1	NotFree	X	FirtsMB	Overrun
FIFO enabled, Queue enabled						
1	1	0	X	NotFull	FIFO	
1	1	0	None	Full	None	Frame lost by FIFO full (FIFO Overflow)
1	1	0	Free	Full	FirstMB	
1	1	0	NotFree	Full	LastMB	Overrun
1	1	1	None	NotFull	FIFO	
1	1	1	Free	X	FirstMB	
1	1	1	NotFree	NotFull	FIFO	
1	1	1	NotFree	Full	LastMB	Overrun

1. It is a don't care condition.
2. Matched in MB "None" means that the frame has not matched any MB (free-to-receive or non-free-to-receive).
3. It is a forbidden condition.
4. Matched in MB "Free" means that the frame matched at least one MB free-to-receive regardless it has matched MBs non-free-to-receive.
5. Matched in FIFO "None" means that the frame has not matched any filter in FIFO. It is as the FIFO didn't exist (CTRL2[RFEN]=0).
6. Matched in FIFO "NotFull" means that the frame has matched a FIFO filter and has empty slots to receive it.
7. Matched in FIFO "Full" means that the frame has matched a FIFO filter but couldn't store it because it has no empty slots to receive it.

The ability to match the same ID in more than one MB can be exploited to implement a reception queue (in addition to the full featured FIFO) to allow more time for Arm to service the MBs. By programming more than one MB with the same ID, received messages will be queued into the MBs. Arm can examine the Time Stamp field of the MBs to determine the order in which the messages arrived.

Matching to a range of IDs is possible by using ID Acceptance Masks. FlexCAN supports individual masking per MB. Please refer to Rx Mailboxes Global Mask Register (FLEXCAN_RXMGMASK). During the matching algorithm, if a mask bit is asserted, then the corresponding ID bit is compared. If the mask bit is negated, the corresponding ID bit is "don't care". Please note that the Individual Mask Registers are implemented in RAM, so they are not initialized out of reset. Also, they can only be programmed while the module is in Freeze Mode,, otherwise they are blocked by hardware.

FlexCAN also supports an alternate masking scheme with only four mask registers (RGXMASK, RX14MASK, RX15MASK and RXFGMASK) for backward compatibility. This alternate masking scheme is enabled when the IRMQ bit in the MCR Register is negated.

44.7.6 Move Process

There are two types of move process, namely move-in and move-out.

44.7.6.1 Move-in

The move-in process is the copy of a message received by an Rx SMB to a Rx Mailbox or FIFO that has matched it. If the move destination is the Rx FIFO, attributes of the message are also copied to the RXFIR FIFO. Each Rx SMB has its own move-in process, but only one is performed at a given time as described ahead. The move-in starts only when the message held by the Rx SMB has a corresponding matching winner (see Matching Process) and all of the following conditions are true:

- the CAN bus has reached or let past either:
  - the second bit of Intermission field next to the frame that carried the message that is in the Rx SMB;
  - the first bit of an overload frame next to the frame that carried the message that is in the Rx SMB;
- there is no ongoing matching process;

- the destination Mailbox is not locked by Arm;
- there is no ongoing move-in process from another Rx SMB. If more than one move-in processes are to be started at the same time both are performed and the newest substitutes the oldest.

The term *pending move-in* is used throughout the document and stands for a move-to-be that still does not satisfy all of the aforementioned conditions.

The move-in is cancelled and the Rx SMB is able to receive another message if any of the following conditions is satisfied:

- the destination Mailbox is inactivated after the CAN bus has reached the first bit of Intermission field next to the frame that carried the message and its matching process has finished;
- there is a previous pending move-in to the same destination Mailbox.
- the Rx SMB is receiving a frame transmitted by the FlexCAN itself and the self-reception is disabled;
- any CAN protocol error is detected.

Note that the pending move-in is not cancelled if the module enters in Freeze or Low Power Mode. It only stays on hold waiting for exiting Low Power Mode and to be unlocked. If an MB is unlocked during Freeze Mode, the move-in happens immediately.

The move-in process consists of the following steps:

1. if the message is destined to the Rx FIFO, push IDHIT into the RXFIR FIFO;
2. reads the words DATA0-3 and DATA4-7 from the Rx SMB;
3. writes it in the words DATA0-3 and DATA4-7 of the Rx Mailbox;
4. reads the words Control/Status and ID from the Rx SMB;
5. writes it in the words Control/Status and ID of the Rx Mailbox, updating the CODE field according to Table 44-5.

The move-in process is not atomic, in such a way that it is immediately cancelled by the inactivation of the destination Mailbox (see Message Buffer Inactivation) and in this case the Mailbox may be left partially updated, thus incoherent. The exception is if the move-in destination is an Rx FIFO Message Buffer, then the process cannot be cancelled.

The BUSY Bit (least significant bit of the CODE field) of the destination Message Buffer is asserted while the move-in is being performed in such a way that Arm beware that the Message Buffer content is temporarily incoherent.

44.7.6.2 Move-out

The move-out process is the copy of the content from a Tx Mailbox to the Tx SMB when a message for transmission is available (see Arbitration process). The move-out occurs in the following conditions:

- the first bit of Intermission field;
- during Bus off field when TX Error Counter is in the 124 to 128 range;
- during BusIdle field;
- during Wait For Bus Idle field.

The move-out process is not atomic. Only Arm has priority to access the memory concurrently out of BusIdle state. In BusIdle, the move-out has the lowest priority to the concurrent memory accesses.

44.7.7 Data Coherence

In order to maintain data coherency and FlexCAN proper operation, the Arm must obey the rules described in the Transmit Process and Receive Process.

Any form of Arm accessing an MB structure within FlexCAN other than those specified may cause FlexCAN to behave in an unpredictable way.

44.7.7.1 Transmission Abort Mechanism

The abort mechanism provides a safe way to request the abortion of a pending transmission. A feedback mechanism is provided to inform Arm if the transmission was aborted or if the frame could not be aborted and was transmitted instead.

In order to abort a transmission, Arm must write a specific abort code (0b1001) to the CODE field of the Control and Status word. The active MBs configured as transmission must be aborted first and then they may be updated. If the abort code is written to a Mailbox that is currently being transmitted, or to a Mailbox that was already loaded into the SMB for transmission, the write operation is blocked and the MB is kept active, but the abort request is captured and kept pending until one of the following conditions are satisfied:

- The module loses the bus arbitration
- There is an error during the transmission
- The module is put into Freeze Mode
- The module enters in BusOff state
- There is an overload frame

If none of conditions above are reached, the MB is transmitted correctly, the interrupt flag is set in the IFLAG register and an interrupt to the Arm is generated (if enabled). The abort request is automatically cleared when the interrupt flag is set. In the other hand, if one of the above conditions is reached, the frame is not transmitted, therefore the abort code is written into the CODE field, the interrupt flag is set in the IFLAG and an interrupt is (optionally) generated to Arm.

If Arm writes the ABORT code before the transmission begins internally, then the write operation is not blocked, therefore the MB is updated and the interrupt flag is set. In this way Arm just needs to read the abort code to make sure the active MB was *safely inactivated*. Although the AEN bit is asserted and Arm wrote the abort code, in this case the MB is inactivated and not aborted, because the transmission did not start yet. One Mailbox is only aborted when the abort request is captured and kept pending until one of the previous conditions are satisfied.

The abort procedure can be summarized as follows:

1. Arm checks the corresponding IFLAG and clears it, if asserted.
2. Arm writes 0b1001 into the CODE field of the C/S word.
3. Arm waits for the corresponding IFLAG indicating that the frame was either transmitted or aborted.
4. Arm reads the CODE field to check if the frame was either transmitted (CODE=0b1000) or aborted (CODE=0b1001).
5. It is necessary to clear the corresponding IFLAG in order to allow the MB to be reconfigured.

44.7.7.2 Message Buffer Inactivation

Inactivation is a mechanism provided to protect the Mailbox against updates by the FlexCAN internal processes, thus allowing Arm to rely on Mailbox data coherence after having updated it, even in Normal Mode.

If a Mailbox is inactivated it does not participate neither in the arbitration nor in the matching process until it is reactivated. See Transmit Process and Receive Process for more detailed instruction on how to inactivate and reactivate a Mailbox.

In order to inactivate a Mailbox Arm must update its CODE field to INACTIVE (either 0b0000 or 0b1000).

As the user is not able to synchronize the CODE field update with the FlexCAN internal processes an inactivation can lead to undesirable results:

- a frame in the bus that matches the filtering of the inactivated Rx Mailbox may be lost without notice, even if there are other Mailboxes with the same filter;
- a frame containing the message within the inactivated Tx Mailbox may be transmitted without notice.

In order to eliminate such risk and perform a *safe inactivation* Arm must use the following mechanism along with the inactivation itself:

- for Tx Mailboxes, the Transmission Abort (see Transmission Abort Mechanism);

The inactivation automatically unlocks the Mailbox (see Message Buffer Lock Mechanism).

Message Buffers that are part of the Rx FIFO cannot be inactivated. There is no write protection on FIFO region by FlexCAN. Arm must keep the data coherence into FIFO region when RFEN is asserted.

44.7.7.3 Message Buffer Lock Mechanism

Besides MB inactivation, FlexCAN has another data coherence mechanism for the receive process. When Arm reads the Control and Status word of an Rx MB with codes FULL or OVERRUN, FlexCAN assumes that Arm wants to read the whole MB in an atomic operation, and thus it sets an internal lock flag for that MB. The lock is released when Arm reads the Free Running Timer (global unlock operation), or when it reads the Control and Status word of another MB regardless of its code or when Arm writes into C/S word from locked MB. The MB locking is done to prevent a new frame to be written into the MB while Arm is reading it.

The locking mechanism only applies to Rx MBs that are not part of FIFO and have a code different than INACTIVE (0b0000) or EMPTY[1] (0b0100). Also, Tx MBs can not be locked.

Suppose, for example, that the FIFO is disabled and the second and the fifth MBs of the array are programmed with the same ID, and FlexCAN has already received and stored messages into these two MBs. Suppose now that the Arm decides to read MB number 5 and at the same time another message with the same ID is arriving. When Arm reads the Control and Status word of MB number 5, this MB is locked. The new message arrives and the matching algorithm finds out that there are no "free-to-receive" MBs, so it decides to override MB number 5. However, this MB is locked, so the new message can not be written there. It will remain in the SMB waiting for the MB to be unlocked, and only then will be written to the MB. If the MB is not unlocked in time and yet another

1. In previous FlexCAN versions, reading the C/S word locks the MB even if it is EMPTY. This behavior is maintained when the IRMQ bit is negated.

new message with the same ID arrives, then the new message overwrites the one on the SMB and there will be no indication of lost messages either in the CODE field of the MB or in the Error and Status Register.

While the message is being moved-in from the SMB to the MB, the BUSY bit on the CODE field is asserted. If Arm reads the Control and Status word and finds out that the BUSY bit is set, it should defer accessing the MB until the BUSY bit is negated.

If the BUSY bit is asserted or if the MB is empty, then reading the Control and Status word does not lock the MB.

Inactivation takes precedence over locking. If Arm inactivates a locked Rx Mailbox, then its lock status is negated and the Mailbox is marked as invalid for the current matching round. Any pending message on the SMB will not be transferred anymore to the Mailbox.An MB is unlocked when Arm reads the Free Running Timer Register (see Free Running Timer Register (FLEXCAN_TIMER)), or the C/S word of another MB.

Lock and unlock mechanisms have the same functionality in both Normal and Freeze modes.

An unlock during Normal or Freeze mode results in the move-in of the pending message. However, the move-in is postponed if an unlock occurs during any of the low power modes (see in Modes of Operation specific information on Module Disable or Stop modes) and it will take place only when the module resumes to Normal or Freeze modes.

44.7.8 Rx FIFO

The receive-only FIFO is enabled by asserting the RFEN bit in the MCR.

The reset value of this bit is zero to maintain software backward compatibility with previous versions of the module that did not have the FIFO feature. The FIFO is 6-message deep, therefore when the FIFO is enabled, the memory region occupied by the first 6 Message Buffers is reserved for use of the FIFO engine (see Rx FIFO Structure). Arm can read the received messages sequentially, in the order they were received, by repeatedly reading a Message Buffer structure at the output of the FIFO.

The IFLAG[BUF5I] (Frames available in Rx FIFO) is asserted when there is at least one frame available to be read from the FIFO. An interrupt is generated if it is enabled by the corresponding mask bit. Upon receiving the interrupt, Arm can read the message (accessing the output of the FIFO as a Message Buffer) and the RXFIR register and then clear the interrupt. If there are more messages in the FIFO the act of clearing the interrupt updates the output of the FIFO with the next message and update the RXFIR with the attributes of that message, reissuing the interrupt to Arm. Otherwise, the flag remains negated. The output of the FIFO is only valid when the IFLAG[BUF5I] is asserted.

The IFLAG[BUF6I] (Rx FIFO Warning) is asserted when the number of unread messages within the Rx FIFO is increased to 5 from 4 due to the reception of a new one, meaning that the Rx FIFO is almost full. The flag remains asserted until Arm clears it.

The IFLAG[BUF7I] (Rx FIFO Overflow) is asserted when an incoming message was lost because the Rx FIFO is full. Note that the flag will not be asserted when the Rx FIFO is full and the message was captured by a Mailbox. The flag remains asserted until the Arm clears it.

Clearing one of those three flags does not affect the state of the other two.

An interrupt is generated if an IFLAG bit is asserted and the corresponding mask bit is asserted too.

A powerful filtering scheme is provided to accept only frames intended for the target application, thus reducing the interrupt servicing work load. The filtering criteria is specified by programming a table of up to 128 32-bit registers, according to CTRL2[RFFN] setting, that can be configured to one of the following formats (see also Rx FIFO Structure):

- Format A: 128 IDAFs (extended or standard IDs including IDE and RTR)
- Format B: 256 IDAFs (standard IDs or extended 14-bit ID slices including IDE and RTR)
- Format C: 512 IDAFs (standard or extended 8-bit ID slices)

Every frame available in the FIFO has a corresponding IDHIT (Identifier Acceptance Filter Hit Indicator) that can be read by accessing the RXFIR register. The RXFIR[IDHIT] field refers to the message at the output of the FIFO and is valid whilst the IFLAG[BUF5I] flag is asserted. The RXFIR register must be read only before clearing the flag, which guarantees that the information refers to the correct frame within the FIFO.

Up to thirty two elements of the ID Filter Table are individually affected by the Individual Mask Registers (RXIMR0 - RXIMR31), according to CTRL2[RFFN] setting (refer to Control 2 Register (FLEXCAN_CTRL2)), allowing very powerful filtering criteria to be defined. If the MCR[IRMQ] bit is negated (or if the RXIMR are not available for the particular MCU), then the FIFO ID Filter Table is affected by RXFGMASK.

44.7.9 CAN Protocol Related Features

44.7.9.1 Remote Frames

Remote frame is a special kind of frame. The user can program a mailbox to be a Remote Request Frame by writing the mailbox as Transmit with the RTR bit set to '1'. After the remote request frame is transmitted successfully, the mailbox becomes a Receive Message Buffer, with the same ID as before.

When a remote request frame is received by FlexCAN, it can be treated in three ways, depending on Remote Request Storing (CTRL2[RRS]) and Rx FIFO Enable (MCR[RFEN]) bits:

- If RRS is negated the frame's ID is compared to the IDs of the Transmit Message Buffers with the CODE field 0b1010. If there is a matching ID, then this mailbox frame will be transmitted. Note that if the matching mailbox has the RTR bit set, then FlexCAN will transmit a remote frame as a response. The received remote request frame is not stored in a receive buffer. It is only used to trigger a transmission of a frame in response. The mask registers are not used in remote frame matching, and all ID bits (except RTR) of the incoming received frame should match. In the case that a remote request frame was received and matched a mailbox, this message buffer immediately enters the internal arbitration process, but is considered as normal Tx mailbox, with no higher priority. The data length of this frame is independent of the DLC field in the remote frame that initiated its transmission.
- If RRS is asserted the frame's ID is compared to the IDs of the receive mailboxes with the CODE field 0b0100, 0b0010 or 0b0110. If there is a matching ID, then this mailbox will store the remote frame in the same fashion of a data frame. No automatic remote response frame will be generated. The mask registers are used in the matching process.
- If RFEN is asserted FlexCAN will not generate an automatic response for remote request frames that match the FIFO filtering criteria. If the remote frame matches one of the target IDs, it will be stored in the FIFO and presented to the Arm. Note that for filtering formats A and B, it is possible to select whether remote frames are accepted or not. For format C, remote frames are always accepted (if they match the ID).Remote Request Frames are considered as normal frames, and generate a FIFO overflow when a successful reception occurs and the FIFO is already full.

44.7.9.2 Overload Frames

FLEXCAN does transmit overload frames due to detection of following conditions on CAN bus:

- Detection of a dominant bit in the first/second bit of Intermission

- Detection of a dominant bit at the 7th bit (last) of End of Frame field (Rx frames)
- Detection of a dominant bit at the 8th bit (last) of Error Frame Delimiter or Overload Frame Delimiter

44.7.9.3 Time Stamp

The value of the Free Running Timer is sampled at the beginning of the Identifier field on the CAN bus, and is stored at the end of "move-in" in the TIME STAMP field, providing network behavior with respect to time.

Note that the Free Running Timer can be reset upon a specific frame reception, enabling network time synchronization. Refer to TSYN description in Control 1 Register (FLEXCAN_CTRL1).

44.7.9.4 Protocol Timing

The FLEXCAN module supports a variety of means to setup bit timing parameters that are required by the CAN protocol. The Control Register has various fields used to control bit timing parameters: PRESDIV, PROPSEG, PSEG1, PSEG2 and RJW. See Control 1 Register (FLEXCAN_CTRL1).

The PRESDIV field controls a prescaler that generates the Serial Clock (Sclock), whose period defines the 'time quantum' used to compose the CAN waveform. A time quantum is the atomic unit of time handled by the CAN engine.

$$f_{Tq} = \frac{f_{CANCLK}}{(\text{Prescaler value})}$$

A bit time is subdivided into three segments[2] (reference Table 44-17):

2. For further explanation of the underlying concepts please refer to ISO/DIS 11519-1, Section 10.3. Reference also the Bosch CAN 2.0A/B protocol specification dated September 1991 for bit timing.

- SYNC_SEG: This segment has a fixed length of one time quantum. Signal edges are expected to happen within this section.
- Time Segment 1: This segment includes the Propagation Segment and the Phase Segment 1 of the CAN standard. It can be programmed by setting the PROPSEG and the PSEG1 fields of the CTRL Register so that their sum (plus 2) is in the range of 2 to 16 time quanta.
- Time Segment 2: This segment represents the Phase Segment 2 of the CAN standard. It can be programmed by setting the PSEG2 field of the CTRL Register (plus 1) to be 2 to 8 time quanta long.

$$\text{Bit Rate} = \frac{f_{Tq}}{(\text{number of Time Quanta})}$$

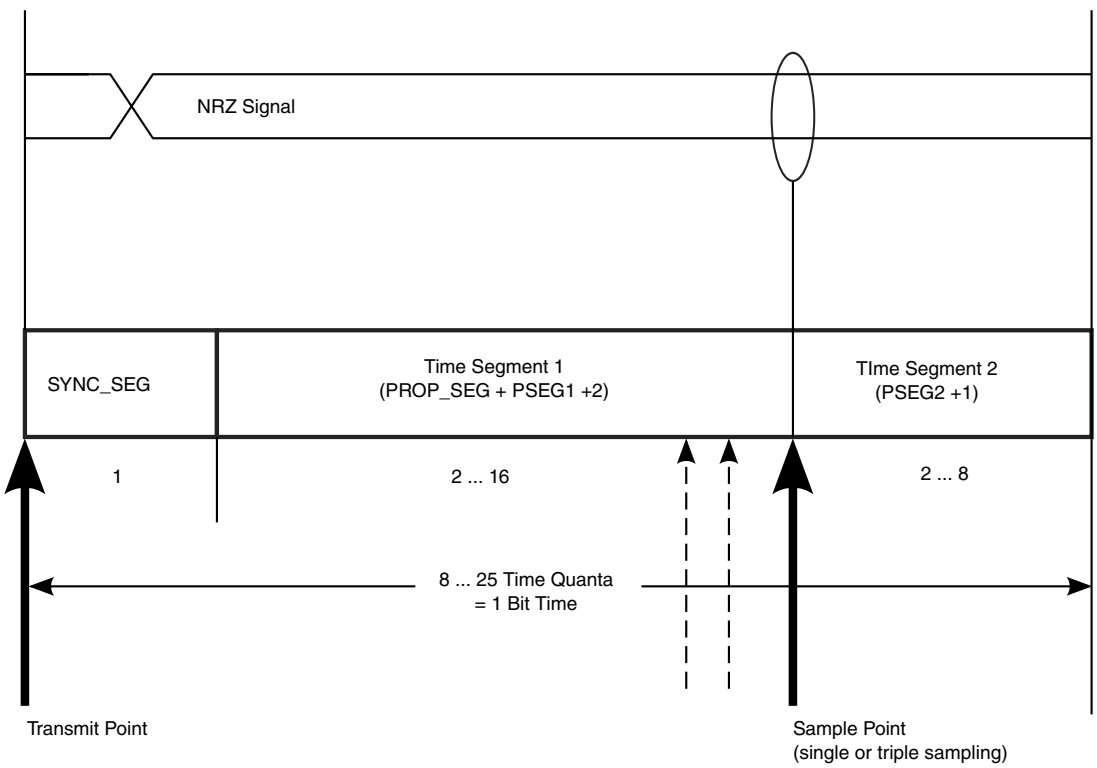


Figure 44-2. Segments within the Bit Time

Whenever CAN bit is used as a measure of duration (e.g. MCR[FRZ_ACK] and MCR[LPM_ACK] in Module Configuration Register (FLEXCAN_MCR)), the number of peripheral clocks in one CAN bit can be calculated as:

$$\text{NCCP} = \frac{f_{sys} \times [1 + (\text{PSEG1} + 1) + (\text{PSEG2} + 1) + (\text{PROPSEG} + 1)] \times (\text{PRESDIV} + 1)}{f_{CANCLK}}$$

where:

NCCP is the number of peripheral clocks in one CAN bit;

f_{CANCLK} is the Protocol Engine (PE) Clock in Hz;

f_{SYS} is the frequency of operation of the system (CHI) clock, in Hz;

PSEG1 is the value in CTRL1[PSEG1] field;

PSEG2 is the value in CTRL1[PSEG2] field;

PROPSEG is the value in CTRL1[PROPSEG] field;

PRESDIV is the value in CTRL1[PRESDIV] field.

For example, 180 CAN bits = 180 x NCCP peripheral clock periods.

Figure 44-2 gives an overview of the CAN compliant segment settings and the related parameter values.

Table 44-17. Time Segment Syntax

Syntax	Description
SYNC_SEG	System expects transitions to occur on the bus during this period.
Transmit Point	A node in transmit mode transfers a new value to the CAN bus at this point.
Sample Point	A node samples the bus at this point. If the three samples per bit option is selected, then this point marks the position of the third sample.

Table 44-18. CAN Standard Compliant Bit Time Segment Settings

Time Segment 1	Time Segment 2	Re-synchronization Jump Width
5 .. 10	2	1 .. 2
4 .. 11	3	1 .. 3
5 .. 12	4	1 .. 4
6 .. 13	5	1 .. 4
7 .. 14	6	1 .. 4
8 .. 15	7	1 .. 4
9 .. 16	8	1 .. 4

NOTE

The user must ensure the bit time settings are in compliance with the CAN Protocol standard (ISO 11898-1).

44.7.9.5 Arbitration and Matching Timing

During normal reception and transmission of frames, the matching, arbitration, move-in and move-out processes are executed during certain time windows inside the CAN frame, as shown in the following figures.

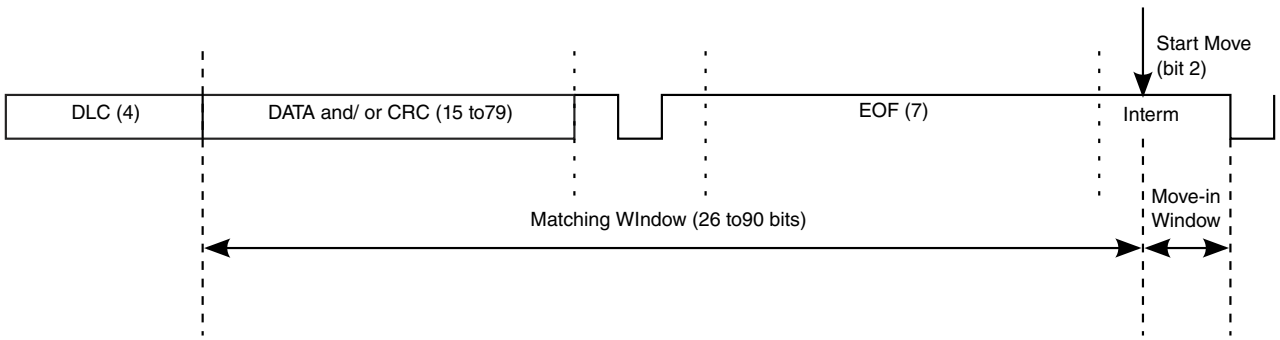


Figure 44-3. Matching and Move-In Time Windows

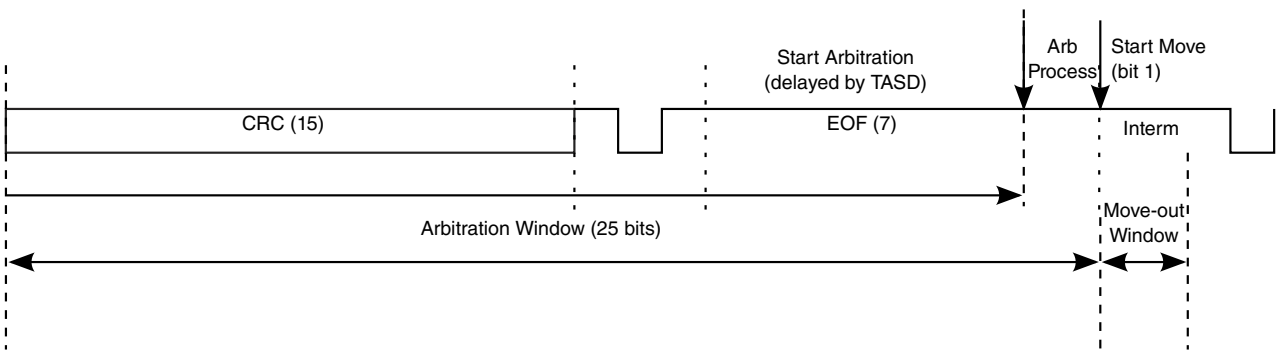


Figure 44-4. Arbitration and Move-Out Time Windows

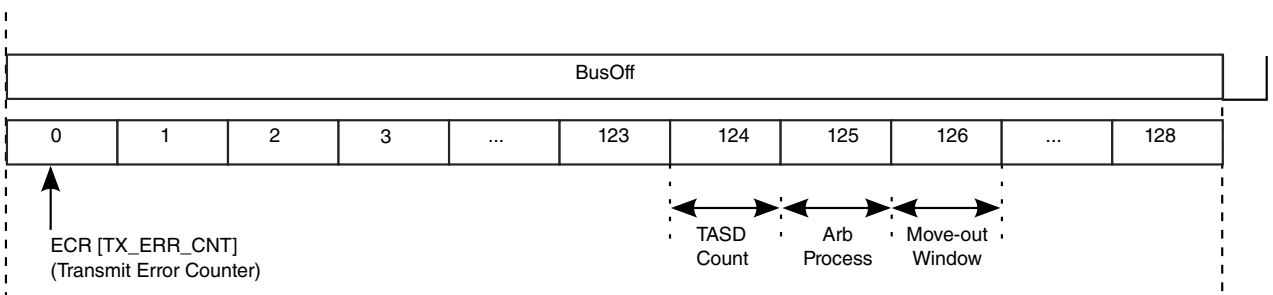


Figure 44-5. Arbitration at the end of Bus Off and Move-Out Time Windows

When doing matching and arbitration, FlexCAN needs to scan the whole Message Buffer memory during the available time window. In order to have sufficient time to do that, the following requirements must be observed:

- A valid CAN bit timing must be programmed, as indicated in Table 44-18
- The peripheral clock frequency can not be smaller than the oscillator clock frequency, i.e. the PLL can not be programmed to divide down the oscillator clock
- There must be a minimum ratio between the peripheral clock frequency and the CAN bit rate, as specified in the following table.

Table 44-19. Minimum Ratio Between Peripheral Clock Frequency and CAN Bit Rate

Number of Message Buffers	RFEN	Minimum Number of Peripheral Clocks per CAN bit
16 and 32	0	16
64	0	25
16	1	16
32	1	17
64	1	30

A direct consequence of the first requirement is that the minimum number of time quanta per CAN bit must be 8, so the oscillator clock frequency should be at least 8 times the CAN bit rate. The minimum frequency ratio specified in Table 44-19 can be achieved by choosing a high enough peripheral clock frequency when compared to the oscillator clock frequency, or by adjusting one or more of the bit timing parameters (PRESDIV, PROPSEG, PSEG1, PSEG2). As an example, taking the case of 64 MBs, if the oscillator and peripheral clock frequencies are equal and the CAN bit timing is programmed to have 8 time quanta per bit, then the prescaler factor (PRESDIV + 1) should be at least 2. For prescaler factor equal to one and CAN bit timing with 8 time quanta per bit, the ratio between peripheral and oscillator clock frequencies should be at least 2.

44.7.10 Modes of Operation Details

The FlexCAN module has four functional modes (Normal Mode, Freeze Mode, Listen-Only Mode and Loop-Back Mode) and two low power modes (Disable Mode and Stop Mode).See in Modes of Operation an introductory description of all these modes of operation. The following sub-sections bring functional details on Freeze mode and the low power modes.

44.7.10.1 Freeze Mode

This mode is requested by Arm through the assertion of the HALT bit in the MCR Register or when the MCU is put into Debug Mode . In both cases it is also necessary that the FRZ bit is asserted in the MCR Register and the module is not in any of the low power modes (Disable, Stop). The acknowledgement is obtained through the assertion by

the FlexCAN of FRZ_ACK bit in the same register. The Arm must only consider the FlexCAN in Freeze Mode when both request and acknowledgement conditions are satisfied.

When Freeze Mode is requested during transmission or reception, FlexCAN does the following:

- Waits to be in either Intermission, Passive Error, Bus Off or Idle state
- Waits for all internal activities like arbitration, matching, move-in and move-out to finish. Pending move-in is not taken in account
- Ignores the FLEXCAN_RX input pin and drives the FLEXCAN_TX pin as recessive
- Stops the prescaler, thus halting all CAN protocol activities
- Grants write access to the Error Counters Register, which is read-only in other modes
- Sets the NOT_RDY and FRZ_ACK bits in MCR

After requesting Freeze Mode, the user must wait for the FRZ_ACK bit to be asserted in MCR before executing any other action, otherwise FlexCAN may operate in an unpredictable way. In Freeze mode, all memory mapped registers are accessible, except for CTRL1[CLK_SRC] bit that can be read but cannot be written.

Exiting Freeze Mode is done in one of the following ways:

- Arm negates the FRZ bit in the MCR Register
- The Arm is removed from Debug Mode and the HALT bit is negated

The FRZ_ACK bit is negated after protocol engine recognizes the negation of freeze request. Once out of Freeze Mode, FlexCAN tries to re-synchronize to the CAN bus by waiting for 11 consecutive recessive bits.

44.7.10.2 Module Disable Mode

This low power mode is normally used to temporarily disable a complete FlexCAN block, with no power consumption. It is requested by the Arm through the assertion of the MDIS bit in the MCR Register and the acknowledgement is obtained through the assertion by the FlexCAN of the LPM_ACK bit in the same register. The Arm must only consider the FlexCAN in Disable Mode when both request and acknowledgement conditions are satisfied.

If the module is disabled during Freeze Mode, it requests to disable the clocks to the PE and CHI sub-modules, sets the LPM_ACK bit and negates the FRZ_ACK bit. The ability to shut down the clocks depends on how FlexCAN is integrated into the MCU. If the module is disabled during transmission or reception, FlexCAN does the following:

- Waits to be in either Idle or Bus Off state, or else waits for the third bit of Intermission and then checks it to be recessive
- Waits for all internal activities like arbitration, matching, move-in and move-out to finish. Pending move-in is not taken in account
- Ignores its FLEXCAN_RX input pin and drives its FLEXCAN_TX pin as recessive
- May shut down the clocks to the PE and CHI sub-modules, depending on how FlexCAN is integrated into the MCU
- Sets the NOT_RDY and LPM_ACK bits in MCR

The Bus Interface Unit continues to operate, enabling the Arm to access memory mapped registers, except the Rx Mailboxes Global Mask Registers, the Rx Buffer 14 Mask Register, the Rx Buffer 15 Mask Register, the Rx FIFO Global Mask Register. The Rx FIFO Information Register, the Message Buffers, the Rx Individual Mask Registers, and the reserved words within RAM may not be accessed when the module is in Disable Mode depending on how FlexCAN RAM is integrated into the Arm. Exiting from this mode is done by negating the MDIS bit by Arm, which make FlexCAN requests to resume the clocks and negates the LPM_ACK bit after CAN protocol engine recognizes the negation of disable mode requested by Arm.

44.7.10.3 Stop Mode

This is a system low power mode in which system clocks can be stopped for maximum power savings. To enter stop mode, the CPU should manually assert a global Stop Mode request (see the CAN1_STOP_REQ and CAN2_STOP_REQ bit in the register IOMUXC_GPR4) and check the acknowledgement asserted by the FlexCAN (see the CAN1_STOP_ACK and CAN2_STOP_ACK in the register IOMUXC_GPR4). The CPU must only consider the FlexCAN in Stop Mode when both request and acknowledgement conditions are satisfied.

If FlexCAN receives the global Stop Mode request during Freeze Mode, it sets the LPM_ACK bit, negates the FRZ_ACK bit and then sends the Stop Acknowledge signal to the CPU, in order to shut down the clocks globally. If Stop Mode is requested during transmission or reception, FlexCAN does the following:

- Waits to be in either Idle or Bus Off state, or else waits for the third bit of Intermission and checks it to be recessive
- Waits for all internal activities like arbitration, matching, move-in and move-out to finish. Pending move-in is not taken in account
- Ignores its FLEXCAN_RX input pin and drives its FLEXCAN_TX pin as recessive
- Sets the NOT_RDY and LPM_ACK bits in MCR
- Sends a Stop Acknowledge signal to the CPU, so that it can shut down the clocks globally

Exiting Stop Mode is done in one of the following ways:

- Arm resuming the clocks and removing the Stop Mode request
- Arm resuming the clocks and Stop Mode request as a result of the Self Wake mechanism

In the Self Wake mechanism, if the SLF_WAK bit in MCR Register was set at the time FlexCAN entered Stop Mode, then upon detection of a recessive to dominant transition on the CAN bus, FlexCAN sets the WAK_INT bit in the ESR Register and, if enabled by the WAK_MSK bit in MCR, generates a Wake Up interrupt to the Arm. Upon receiving the interrupt, the Arm should resume the clocks and remove the Stop Mode request manually. FlexCAN will then wait for 11 consecutive recessive bits to synchronize to the CAN bus. As a consequence, it will not receive the frame that woke it up .

The sensitivity to CAN bus activity can be modified by applying a low-pass filter function to the FLEXCAN_RX input line while in Stop Mode. See the WAK_SRC bit in Module Configuration Register (FLEXCAN_MCR) . This feature can be used to protect FlexCAN from waking up due to short glitches on the CAN bus lines. Such glitches can result from electromagnetic interference within noisy environments, the glitch filter width can be set in Glitch Filter Width Register (FLEXCAN_GFWR).

44.7.11 Interrupts

The module can generate up to 70 interrupt sources (64 interrupts due to message buffers and 6 interrupts due to Ored interrupts from MBs, Bus Off, Error, Tx Warning, Rx Warning and Wake Up)).

The number of actual sources depends on the configured number of message buffers.

Each one of the message buffers can be an interrupt source, if its corresponding IMASK bit is set. There is no distinction between Tx and Rx interrupts for a particular buffer, under the assumption that the buffer is initialized for either transmission or reception. Each of the buffers has assigned a flag bit in the IFLAG Registers. The bit is set when the corresponding buffer completes a successful transmission/reception and is cleared when the Arm writes it to '1' (unless another interrupt is generated at the same time).

If the Rx FIFO is enabled (bit RFEN on MCR set), the interrupts corresponding to MBs 0 to 7 have a different behavior. Bit 7 of the IFLAG1 becomes the "FIFO Overflow" flag; bit 6 becomes the FIFO Warning flag, bit 5 becomes the "Frames Available in FIFO flag" and bits 4-0 are unused. See Interrupt Flags 1 Register (FLEXCAN_IFLAG1) for more information.

A combined interrupt for all MBs is also generated by an Or of all the interrupt sources from MBs. This interrupt gets generated when any of the Mailboxes or FIFO generates an interrupt. The Arm must read the IFLAG Registers to determine which MB or FIFO caused the interrupt.

The other 5 interrupt sources (Bus Off, Error, Tx Warning, Rx Warning and Wake Up generate interrupts like the MB ones, and can be read from both the Error and Status Register 1 and 2. The Bus Off, Error, Tx Warning and Rx Warning interrupt mask bits are located in the Control 1 Register and the Wake-Up interrupt mask bit is located in the MCR.

44.8 Initialization/Application Information

This section provide instructions for initializing the FLEXCAN module.

44.8.1 FLEXCAN Initialization Sequence

The FLEXCAN module may be reset in two ways:

- SOC level hard reset which resets all memory mapped registers asynchronously
- SOFT_RST bit in MCR, which resets some of the memory mapped registers synchronously

Soft reset is synchronous and has to follow an internal request/acknowledge procedure across clock domains. Therefore, it may take some time to fully propagate its effects. The SOFT_RST bit remains asserted while soft reset is pending, so software can poll this bit to know when the reset has completed. Also, soft reset can not be applied while clocks are shut down in any of the low power modes. The low power mode should be exited and the clocks resumed before applying soft reset.

After the module is enabled (MDIS bit negated), FLEXCAN automatically goes to Freeze Mode. In Freeze Mode, FLEXCAN is un-synchronized to the CAN bus, the HALT and FRZ bits in MCR Register are set, the internal state machines are disabled and the FRZ_ACK and NOT_RDY bits in the MCR Register are set. The FLEXCAN_TX pin is in recessive state and FLEXCAN does not initiate any transmission or reception of CAN frames. Note that the Message Buffers and the Rx Individual Mask Registers are not affected by reset, so they are not automatically initialized.

For any configuration change/initialization it is required that FLEXCAN is put into Freeze Mode. The following is a generic initialization sequence applicable to the FLEXCAN module:

- Initialize the Module Configuration Register
  - Enable the individual filtering per MB and reception queue features by setting the IRMQ bit
  - Enable the warning interrupts by setting the WRN_EN bit
  - If required, disable frame self reception by setting the SRX_DIS bit
  - Enable the FIFO by setting the RFEN bit
  - Enable the abort mechanism by setting the AEN bit
  - Enable the local priority feature by setting the LPRIO_EN bit
- Initialize the Control Register
  - Determine the bit timing parameters: PROPSEG, PSEG1, PSEG2, RJW
  - Determine the bit rate by programming the PRESDIV field
  - Determine the internal arbitration mode (LBUF bit)
- Initialize the Message Buffers
  - The Control and Status word of all Message Buffers must be initialized
  - If FIFO was enabled, the 8-entry ID table must be initialized
  - Other entries in each Message Buffer should be initialized as required
- Initialize the Rx Individual Mask Registers
- Set required interrupt mask bits in the IMASK Registers (for all MB interrupts), in CTRL Register (for Bus Off and Error interrupts) and in MCR Register for Wake-Up interrupt
- Negate the HALT bit in MCR

Starting with the last event, FLEXCAN attempts to synchronize to the CAN bus.

44.9 FLEXCAN Memory Map/Register Definition

The complete memory map for a FLEXCAN module with 64 MBs capability is shown in the following table. Each individual register is identified by its complete name and the corresponding mnemonic. The access type can be Supervisor (S) or Unrestricted (U). Most of the registers can be configured to have either Supervisor or Unrestricted access by programming the SUPV bit in the MCR Register. The MCR register allows only Supervisor access regardless the SUPV bit state.

The FLEXCAN module stores CAN messages for transmission and reception using a Mailboxes and Rx FIFO structure.

FLEXCAN memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
401D_0000	Module Configuration Register (FLEXCAN1_MCR)	32	R/W	5980_000Fh	44.9.1/2611
401D_0004	Control 1 Register (FLEXCAN1_CTRL1)	32	R/W	0000_0000h	44.9.2/2616
401D_0008	Free Running Timer Register (FLEXCAN1_TIMER)	32	R/W	0000_0000h	44.9.3/2619
401D_0010	Rx Mailboxes Global Mask Register (FLEXCAN1_RXMGMASK)	32	R/W	FFFF_FFFFh	44.9.4/2619
401D_0014	Rx Buffer 14 Mask Register (FLEXCAN1_RX14MASK)	32	R/W	FFFF_FFFFh	44.9.5/2620
401D_0018	Rx Buffer 15 Mask Register (FLEXCAN1_RX15MASK)	32	R/W	FFFF_FFFFh	44.9.6/2621
401D_001C	Error Counter Register (FLEXCAN1_ECR)	32	R/W	0000_0000h	44.9.7/2622
401D_0020	Error and Status 1 Register (FLEXCAN1_ESR1)	32	R/W	0000_0000h	44.9.8/2623
401D_0024	Interrupt Masks 2 Register (FLEXCAN1_IMASK2)	32	R/W	0000_0000h	44.9.9/2627
401D_0028	Interrupt Masks 1 Register (FLEXCAN1_IMASK1)	32	R/W	0000_0000h	44.9.10/ 2627
401D_002C	Interrupt Flags 2 Register (FLEXCAN1_IFLAG2)	32	R/W	0000_0000h	44.9.11/ 2628
401D_0030	Interrupt Flags 1 Register (FLEXCAN1_IFLAG1)	32	R/W	0000_0000h	44.9.12/ 2628
401D_0034	Control 2 Register (FLEXCAN1_CTRL2)	32	R/W	0000_0000h	44.9.13/ 2630
401D_0038	Error and Status 2 Register (FLEXCAN1_ESR2)	32	R	0000_0000h	44.9.14/ 2636
401D_0044	CRC Register (FLEXCAN1_CRCR)	32	R	0000_0000h	44.9.15/ 2638
401D_0048	Rx FIFO Global Mask Register (FLEXCAN1_RXFGMASK)	32	R/W	FFFF_FFFFh	44.9.16/ 2639
401D_004C	Rx FIFO Information Register (FLEXCAN1_RXFIR)	32	R	0000_0000h	44.9.17/ 2640
401D_0058	Debug 1 register (FLEXCAN1_DBG1)	32	R	0001_0000h	44.9.18/ 2640
401D_005C	Debug 2 register (FLEXCAN1_DBG2)	32	R	0000_0000h	44.9.19/ 2643
401D_0880	Rx Individual Mask Registers (FLEXCAN1_RXIMR0)	32	R/W	0000_0000h	44.9.20/ 2644
401D_0884	Rx Individual Mask Registers (FLEXCAN1_RXIMR1)	32	R/W	0000_0000h	44.9.20/ 2644
401D_0888	Rx Individual Mask Registers (FLEXCAN1_RXIMR2)	32	R/W	0000_0000h	44.9.20/ 2644
401D_088C	Rx Individual Mask Registers (FLEXCAN1_RXIMR3)	32	R/W	0000_0000h	44.9.20/ 2644
401D_0890	Rx Individual Mask Registers (FLEXCAN1_RXIMR4)	32	R/W	0000_0000h	44.9.20/ 2644
401D_0894	Rx Individual Mask Registers (FLEXCAN1_RXIMR5)	32	R/W	0000_0000h	44.9.20/ 2644

Table continues on the next page...

FLEXCAN memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
401D_0898	Rx Individual Mask Registers (FLEXCAN1_RXIMR6)	32	R/W	0000_0000h	44.9.20/ 2644
401D_089C	Rx Individual Mask Registers (FLEXCAN1_RXIMR7)	32	R/W	0000_0000h	44.9.20/ 2644
401D_08A0	Rx Individual Mask Registers (FLEXCAN1_RXIMR8)	32	R/W	0000_0000h	44.9.20/ 2644
401D_08A4	Rx Individual Mask Registers (FLEXCAN1_RXIMR9)	32	R/W	0000_0000h	44.9.20/ 2644
401D_08A8	Rx Individual Mask Registers (FLEXCAN1_RXIMR10)	32	R/W	0000_0000h	44.9.20/ 2644
401D_08AC	Rx Individual Mask Registers (FLEXCAN1_RXIMR11)	32	R/W	0000_0000h	44.9.20/ 2644
401D_08B0	Rx Individual Mask Registers (FLEXCAN1_RXIMR12)	32	R/W	0000_0000h	44.9.20/ 2644
401D_08B4	Rx Individual Mask Registers (FLEXCAN1_RXIMR13)	32	R/W	0000_0000h	44.9.20/ 2644
401D_08B8	Rx Individual Mask Registers (FLEXCAN1_RXIMR14)	32	R/W	0000_0000h	44.9.20/ 2644
401D_08BC	Rx Individual Mask Registers (FLEXCAN1_RXIMR15)	32	R/W	0000_0000h	44.9.20/ 2644
401D_08C0	Rx Individual Mask Registers (FLEXCAN1_RXIMR16)	32	R/W	0000_0000h	44.9.20/ 2644
401D_08C4	Rx Individual Mask Registers (FLEXCAN1_RXIMR17)	32	R/W	0000_0000h	44.9.20/ 2644
401D_08C8	Rx Individual Mask Registers (FLEXCAN1_RXIMR18)	32	R/W	0000_0000h	44.9.20/ 2644
401D_08CC	Rx Individual Mask Registers (FLEXCAN1_RXIMR19)	32	R/W	0000_0000h	44.9.20/ 2644
401D_08D0	Rx Individual Mask Registers (FLEXCAN1_RXIMR20)	32	R/W	0000_0000h	44.9.20/ 2644
401D_08D4	Rx Individual Mask Registers (FLEXCAN1_RXIMR21)	32	R/W	0000_0000h	44.9.20/ 2644
401D_08D8	Rx Individual Mask Registers (FLEXCAN1_RXIMR22)	32	R/W	0000_0000h	44.9.20/ 2644
401D_08DC	Rx Individual Mask Registers (FLEXCAN1_RXIMR23)	32	R/W	0000_0000h	44.9.20/ 2644
401D_08E0	Rx Individual Mask Registers (FLEXCAN1_RXIMR24)	32	R/W	0000_0000h	44.9.20/ 2644
401D_08E4	Rx Individual Mask Registers (FLEXCAN1_RXIMR25)	32	R/W	0000_0000h	44.9.20/ 2644
401D_08E8	Rx Individual Mask Registers (FLEXCAN1_RXIMR26)	32	R/W	0000_0000h	44.9.20/ 2644
401D_08EC	Rx Individual Mask Registers (FLEXCAN1_RXIMR27)	32	R/W	0000_0000h	44.9.20/ 2644

Table continues on the next page...

FLEXCAN memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
401D_08F0	Rx Individual Mask Registers (FLEXCAN1_RXIMR28)	32	R/W	0000_0000h	44.9.20/ 2644
401D_08F4	Rx Individual Mask Registers (FLEXCAN1_RXIMR29)	32	R/W	0000_0000h	44.9.20/ 2644
401D_08F8	Rx Individual Mask Registers (FLEXCAN1_RXIMR30)	32	R/W	0000_0000h	44.9.20/ 2644
401D_08FC	Rx Individual Mask Registers (FLEXCAN1_RXIMR31)	32	R/W	0000_0000h	44.9.20/ 2644
401D_0900	Rx Individual Mask Registers (FLEXCAN1_RXIMR32)	32	R/W	0000_0000h	44.9.20/ 2644
401D_0904	Rx Individual Mask Registers (FLEXCAN1_RXIMR33)	32	R/W	0000_0000h	44.9.20/ 2644
401D_0908	Rx Individual Mask Registers (FLEXCAN1_RXIMR34)	32	R/W	0000_0000h	44.9.20/ 2644
401D_090C	Rx Individual Mask Registers (FLEXCAN1_RXIMR35)	32	R/W	0000_0000h	44.9.20/ 2644
401D_0910	Rx Individual Mask Registers (FLEXCAN1_RXIMR36)	32	R/W	0000_0000h	44.9.20/ 2644
401D_0914	Rx Individual Mask Registers (FLEXCAN1_RXIMR37)	32	R/W	0000_0000h	44.9.20/ 2644
401D_0918	Rx Individual Mask Registers (FLEXCAN1_RXIMR38)	32	R/W	0000_0000h	44.9.20/ 2644
401D_091C	Rx Individual Mask Registers (FLEXCAN1_RXIMR39)	32	R/W	0000_0000h	44.9.20/ 2644
401D_0920	Rx Individual Mask Registers (FLEXCAN1_RXIMR40)	32	R/W	0000_0000h	44.9.20/ 2644
401D_0924	Rx Individual Mask Registers (FLEXCAN1_RXIMR41)	32	R/W	0000_0000h	44.9.20/ 2644
401D_0928	Rx Individual Mask Registers (FLEXCAN1_RXIMR42)	32	R/W	0000_0000h	44.9.20/ 2644
401D_092C	Rx Individual Mask Registers (FLEXCAN1_RXIMR43)	32	R/W	0000_0000h	44.9.20/ 2644
401D_0930	Rx Individual Mask Registers (FLEXCAN1_RXIMR44)	32	R/W	0000_0000h	44.9.20/ 2644
401D_0934	Rx Individual Mask Registers (FLEXCAN1_RXIMR45)	32	R/W	0000_0000h	44.9.20/ 2644
401D_0938	Rx Individual Mask Registers (FLEXCAN1_RXIMR46)	32	R/W	0000_0000h	44.9.20/ 2644
401D_093C	Rx Individual Mask Registers (FLEXCAN1_RXIMR47)	32	R/W	0000_0000h	44.9.20/ 2644
401D_0940	Rx Individual Mask Registers (FLEXCAN1_RXIMR48)	32	R/W	0000_0000h	44.9.20/ 2644
401D_0944	Rx Individual Mask Registers (FLEXCAN1_RXIMR49)	32	R/W	0000_0000h	44.9.20/ 2644

Table continues on the next page...

FLEXCAN memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
401D_0948	Rx Individual Mask Registers (FLEXCAN1_RXIMR50)	32	R/W	0000_0000h	44.9.20/ 2644
401D_094C	Rx Individual Mask Registers (FLEXCAN1_RXIMR51)	32	R/W	0000_0000h	44.9.20/ 2644
401D_0950	Rx Individual Mask Registers (FLEXCAN1_RXIMR52)	32	R/W	0000_0000h	44.9.20/ 2644
401D_0954	Rx Individual Mask Registers (FLEXCAN1_RXIMR53)	32	R/W	0000_0000h	44.9.20/ 2644
401D_0958	Rx Individual Mask Registers (FLEXCAN1_RXIMR54)	32	R/W	0000_0000h	44.9.20/ 2644
401D_095C	Rx Individual Mask Registers (FLEXCAN1_RXIMR55)	32	R/W	0000_0000h	44.9.20/ 2644
401D_0960	Rx Individual Mask Registers (FLEXCAN1_RXIMR56)	32	R/W	0000_0000h	44.9.20/ 2644
401D_0964	Rx Individual Mask Registers (FLEXCAN1_RXIMR57)	32	R/W	0000_0000h	44.9.20/ 2644
401D_0968	Rx Individual Mask Registers (FLEXCAN1_RXIMR58)	32	R/W	0000_0000h	44.9.20/ 2644
401D_096C	Rx Individual Mask Registers (FLEXCAN1_RXIMR59)	32	R/W	0000_0000h	44.9.20/ 2644
401D_0970	Rx Individual Mask Registers (FLEXCAN1_RXIMR60)	32	R/W	0000_0000h	44.9.20/ 2644
401D_0974	Rx Individual Mask Registers (FLEXCAN1_RXIMR61)	32	R/W	0000_0000h	44.9.20/ 2644
401D_0978	Rx Individual Mask Registers (FLEXCAN1_RXIMR62)	32	R/W	0000_0000h	44.9.20/ 2644
401D_097C	Rx Individual Mask Registers (FLEXCAN1_RXIMR63)	32	R/W	0000_0000h	44.9.20/ 2644
401D_09E0	Glitch Filter Width Registers (FLEXCAN1_GFWR)	32	R/W	0000_007Fh	44.9.21/ 2644
401D_4000	Module Configuration Register (FLEXCAN2_MCR)	32	R/W	5980_000Fh	44.9.1/2611
401D_4004	Control 1 Register (FLEXCAN2_CTRL1)	32	R/W	0000_0000h	44.9.2/2616
401D_4008	Free Running Timer Register (FLEXCAN2_TIMER)	32	R/W	0000_0000h	44.9.3/2619
401D_4010	Rx Mailboxes Global Mask Register (FLEXCAN2_RXMGMASK)	32	R/W	FFFF_FFFFh	44.9.4/2619
401D_4014	Rx Buffer 14 Mask Register (FLEXCAN2_RX14MASK)	32	R/W	FFFF_FFFFh	44.9.5/2620
401D_4018	Rx Buffer 15 Mask Register (FLEXCAN2_RX15MASK)	32	R/W	FFFF_FFFFh	44.9.6/2621
401D_401C	Error Counter Register (FLEXCAN2_ECR)	32	R/W	0000_0000h	44.9.7/2622
401D_4020	Error and Status 1 Register (FLEXCAN2_ESR1)	32	R/W	0000_0000h	44.9.8/2623
401D_4024	Interrupt Masks 2 Register (FLEXCAN2_IMASK2)	32	R/W	0000_0000h	44.9.9/2627
401D_4028	Interrupt Masks 1 Register (FLEXCAN2_IMASK1)	32	R/W	0000_0000h	44.9.10/ 2627

Table continues on the next page...

FLEXCAN memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
401D_402C	Interrupt Flags 2 Register (FLEXCAN2_IFLAG2)	32	R/W	0000_0000h	44.9.11/ 2628
401D_4030	Interrupt Flags 1 Register (FLEXCAN2_IFLAG1)	32	R/W	0000_0000h	44.9.12/ 2628
401D_4034	Control 2 Register (FLEXCAN2_CTRL2)	32	R/W	0000_0000h	44.9.13/ 2630
401D_4038	Error and Status 2 Register (FLEXCAN2_ESR2)	32	R	0000_0000h	44.9.14/ 2636
401D_4044	CRC Register (FLEXCAN2_CRCR)	32	R	0000_0000h	44.9.15/ 2638
401D_4048	Rx FIFO Global Mask Register (FLEXCAN2_RXFGMASK)	32	R/W	FFFF_FFFFh	44.9.16/ 2639
401D_404C	Rx FIFO Information Register (FLEXCAN2_RXFIR)	32	R	0000_0000h	44.9.17/ 2640
401D_4058	Debug 1 register (FLEXCAN2_DBG1)	32	R	0001_0000h	44.9.18/ 2640
401D_405C	Debug 2 register (FLEXCAN2_DBG2)	32	R	0000_0000h	44.9.19/ 2643
401D_4880	Rx Individual Mask Registers (FLEXCAN2_RXIMR0)	32	R/W	0000_0000h	44.9.20/ 2644
401D_4884	Rx Individual Mask Registers (FLEXCAN2_RXIMR1)	32	R/W	0000_0000h	44.9.20/ 2644
401D_4888	Rx Individual Mask Registers (FLEXCAN2_RXIMR2)	32	R/W	0000_0000h	44.9.20/ 2644
401D_488C	Rx Individual Mask Registers (FLEXCAN2_RXIMR3)	32	R/W	0000_0000h	44.9.20/ 2644
401D_4890	Rx Individual Mask Registers (FLEXCAN2_RXIMR4)	32	R/W	0000_0000h	44.9.20/ 2644
401D_4894	Rx Individual Mask Registers (FLEXCAN2_RXIMR5)	32	R/W	0000_0000h	44.9.20/ 2644
401D_4898	Rx Individual Mask Registers (FLEXCAN2_RXIMR6)	32	R/W	0000_0000h	44.9.20/ 2644
401D_489C	Rx Individual Mask Registers (FLEXCAN2_RXIMR7)	32	R/W	0000_0000h	44.9.20/ 2644
401D_48A0	Rx Individual Mask Registers (FLEXCAN2_RXIMR8)	32	R/W	0000_0000h	44.9.20/ 2644
401D_48A4	Rx Individual Mask Registers (FLEXCAN2_RXIMR9)	32	R/W	0000_0000h	44.9.20/ 2644
401D_48A8	Rx Individual Mask Registers (FLEXCAN2_RXIMR10)	32	R/W	0000_0000h	44.9.20/ 2644
401D_48AC	Rx Individual Mask Registers (FLEXCAN2_RXIMR11)	32	R/W	0000_0000h	44.9.20/ 2644
401D_48B0	Rx Individual Mask Registers (FLEXCAN2_RXIMR12)	32	R/W	0000_0000h	44.9.20/ 2644

Table continues on the next page...

FLEXCAN memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
401D_48B4	Rx Individual Mask Registers (FLEXCAN2_RXIMR13)	32	R/W	0000_0000h	44.9.20/ 2644
401D_48B8	Rx Individual Mask Registers (FLEXCAN2_RXIMR14)	32	R/W	0000_0000h	44.9.20/ 2644
401D_48BC	Rx Individual Mask Registers (FLEXCAN2_RXIMR15)	32	R/W	0000_0000h	44.9.20/ 2644
401D_48C0	Rx Individual Mask Registers (FLEXCAN2_RXIMR16)	32	R/W	0000_0000h	44.9.20/ 2644
401D_48C4	Rx Individual Mask Registers (FLEXCAN2_RXIMR17)	32	R/W	0000_0000h	44.9.20/ 2644
401D_48C8	Rx Individual Mask Registers (FLEXCAN2_RXIMR18)	32	R/W	0000_0000h	44.9.20/ 2644
401D_48CC	Rx Individual Mask Registers (FLEXCAN2_RXIMR19)	32	R/W	0000_0000h	44.9.20/ 2644
401D_48D0	Rx Individual Mask Registers (FLEXCAN2_RXIMR20)	32	R/W	0000_0000h	44.9.20/ 2644
401D_48D4	Rx Individual Mask Registers (FLEXCAN2_RXIMR21)	32	R/W	0000_0000h	44.9.20/ 2644
401D_48D8	Rx Individual Mask Registers (FLEXCAN2_RXIMR22)	32	R/W	0000_0000h	44.9.20/ 2644
401D_48DC	Rx Individual Mask Registers (FLEXCAN2_RXIMR23)	32	R/W	0000_0000h	44.9.20/ 2644
401D_48E0	Rx Individual Mask Registers (FLEXCAN2_RXIMR24)	32	R/W	0000_0000h	44.9.20/ 2644
401D_48E4	Rx Individual Mask Registers (FLEXCAN2_RXIMR25)	32	R/W	0000_0000h	44.9.20/ 2644
401D_48E8	Rx Individual Mask Registers (FLEXCAN2_RXIMR26)	32	R/W	0000_0000h	44.9.20/ 2644
401D_48EC	Rx Individual Mask Registers (FLEXCAN2_RXIMR27)	32	R/W	0000_0000h	44.9.20/ 2644
401D_48F0	Rx Individual Mask Registers (FLEXCAN2_RXIMR28)	32	R/W	0000_0000h	44.9.20/ 2644
401D_48F4	Rx Individual Mask Registers (FLEXCAN2_RXIMR29)	32	R/W	0000_0000h	44.9.20/ 2644
401D_48F8	Rx Individual Mask Registers (FLEXCAN2_RXIMR30)	32	R/W	0000_0000h	44.9.20/ 2644
401D_48FC	Rx Individual Mask Registers (FLEXCAN2_RXIMR31)	32	R/W	0000_0000h	44.9.20/ 2644
401D_4900	Rx Individual Mask Registers (FLEXCAN2_RXIMR32)	32	R/W	0000_0000h	44.9.20/ 2644
401D_4904	Rx Individual Mask Registers (FLEXCAN2_RXIMR33)	32	R/W	0000_0000h	44.9.20/ 2644
401D_4908	Rx Individual Mask Registers (FLEXCAN2_RXIMR34)	32	R/W	0000_0000h	44.9.20/ 2644

Table continues on the next page...

FLEXCAN memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
401D_490C	Rx Individual Mask Registers (FLEXCAN2_RXIMR35)	32	R/W	0000_0000h	44.9.20/ 2644
401D_4910	Rx Individual Mask Registers (FLEXCAN2_RXIMR36)	32	R/W	0000_0000h	44.9.20/ 2644
401D_4914	Rx Individual Mask Registers (FLEXCAN2_RXIMR37)	32	R/W	0000_0000h	44.9.20/ 2644
401D_4918	Rx Individual Mask Registers (FLEXCAN2_RXIMR38)	32	R/W	0000_0000h	44.9.20/ 2644
401D_491C	Rx Individual Mask Registers (FLEXCAN2_RXIMR39)	32	R/W	0000_0000h	44.9.20/ 2644
401D_4920	Rx Individual Mask Registers (FLEXCAN2_RXIMR40)	32	R/W	0000_0000h	44.9.20/ 2644
401D_4924	Rx Individual Mask Registers (FLEXCAN2_RXIMR41)	32	R/W	0000_0000h	44.9.20/ 2644
401D_4928	Rx Individual Mask Registers (FLEXCAN2_RXIMR42)	32	R/W	0000_0000h	44.9.20/ 2644
401D_492C	Rx Individual Mask Registers (FLEXCAN2_RXIMR43)	32	R/W	0000_0000h	44.9.20/ 2644
401D_4930	Rx Individual Mask Registers (FLEXCAN2_RXIMR44)	32	R/W	0000_0000h	44.9.20/ 2644
401D_4934	Rx Individual Mask Registers (FLEXCAN2_RXIMR45)	32	R/W	0000_0000h	44.9.20/ 2644
401D_4938	Rx Individual Mask Registers (FLEXCAN2_RXIMR46)	32	R/W	0000_0000h	44.9.20/ 2644
401D_493C	Rx Individual Mask Registers (FLEXCAN2_RXIMR47)	32	R/W	0000_0000h	44.9.20/ 2644
401D_4940	Rx Individual Mask Registers (FLEXCAN2_RXIMR48)	32	R/W	0000_0000h	44.9.20/ 2644
401D_4944	Rx Individual Mask Registers (FLEXCAN2_RXIMR49)	32	R/W	0000_0000h	44.9.20/ 2644
401D_4948	Rx Individual Mask Registers (FLEXCAN2_RXIMR50)	32	R/W	0000_0000h	44.9.20/ 2644
401D_494C	Rx Individual Mask Registers (FLEXCAN2_RXIMR51)	32	R/W	0000_0000h	44.9.20/ 2644
401D_4950	Rx Individual Mask Registers (FLEXCAN2_RXIMR52)	32	R/W	0000_0000h	44.9.20/ 2644
401D_4954	Rx Individual Mask Registers (FLEXCAN2_RXIMR53)	32	R/W	0000_0000h	44.9.20/ 2644
401D_4958	Rx Individual Mask Registers (FLEXCAN2_RXIMR54)	32	R/W	0000_0000h	44.9.20/ 2644
401D_495C	Rx Individual Mask Registers (FLEXCAN2_RXIMR55)	32	R/W	0000_0000h	44.9.20/ 2644
401D_4960	Rx Individual Mask Registers (FLEXCAN2_RXIMR56)	32	R/W	0000_0000h	44.9.20/ 2644

Table continues on the next page...

FLEXCAN memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
401D_4964	Rx Individual Mask Registers (FLEXCAN2_RXIMR57)	32	R/W	0000_0000h	44.9.20/ 2644
401D_4968	Rx Individual Mask Registers (FLEXCAN2_RXIMR58)	32	R/W	0000_0000h	44.9.20/ 2644
401D_496C	Rx Individual Mask Registers (FLEXCAN2_RXIMR59)	32	R/W	0000_0000h	44.9.20/ 2644
401D_4970	Rx Individual Mask Registers (FLEXCAN2_RXIMR60)	32	R/W	0000_0000h	44.9.20/ 2644
401D_4974	Rx Individual Mask Registers (FLEXCAN2_RXIMR61)	32	R/W	0000_0000h	44.9.20/ 2644
401D_4978	Rx Individual Mask Registers (FLEXCAN2_RXIMR62)	32	R/W	0000_0000h	44.9.20/ 2644
401D_497C	Rx Individual Mask Registers (FLEXCAN2_RXIMR63)	32	R/W	0000_0000h	44.9.20/ 2644
401D_49E0	Glitch Filter Width Registers (FLEXCAN2_GFWR)	32	R/W	0000_007Fh	44.9.21/ 2644

44.9.1 Module Configuration Register (FLEXCANx_MCR)

This register defines global system configurations, such as the module operation mode (e.g., low power) and maximum message buffer configuration.

Address: Base address + 0h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	MDIS	FRZ	RFEN	HALT	NOT_RDY	WAK_MSK	SOFT_RST	FRZ_ACK	SUPV	SLF_WAK	WRN_EN	LPM_ACK	WAK_SRC	Reserved	SRX_DIS	IRMQ
W																
Reset	0	1	0	1	1	0	0	1	1	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved		LPRIO_EN	AEN	Reserved		IDAM		Reserved	MAXMB						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

FLEXCANx_MCR field descriptions

Field	Description
31 MDIS	This bit controls whether FLEXCAN is enabled or not. When disabled, FLEXCAN shuts down the clocks to the CAN Protocol Interface and Message Buffer Management sub-modules. This is the only bit in MCR not affected by soft reset. See Module Disable Mode for more information. 1 Disable the FLEXCAN module 0 Enable the FLEXCAN module
30 FRZ	The FRZ bit specifies the FLEXCAN behavior when the HALT bit in the MCR Register is set or when Debug Mode is requested at Arm level. When FRZ is asserted, FLEXCAN is enabled to enter Freeze Mode. Negation of this bit field causes FLEXCAN to exit from Freeze Mode. 1 Enabled to enter Freeze Mode 0 Not enabled to enter Freeze Mode
29 RFEN	This bit controls whether the Rx FIFO feature is enabled or not. When RFEN is set, MBs 0 to 5 cannot be used for normal reception and transmission because the corresponding memory region (0x80-0xDC) is used by the FIFO engine as well as additional MBs (up to 32, depending on CTRL2[RFFN] setting) which are used as Rx FIFO ID Filter Table elements.RFEN also impacts the definition of the minimum number of peripheral clocks per CAN bit as described in Table 44-19 (see Arbitration and Matching Timing).This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 1 FIFO enabled 0 FIFO not enabled
28 HALT	Assertion of this bit puts the FLEXCAN module into Freeze Mode. The Arm should clear it after initializing the Message Buffers and Control Register. No reception or transmission is performed by FLEXCAN before this bit is cleared. Freeze Mode can not be entered while FLEXCAN is in any of the low power modes.See Freeze Mode for more information 1 Enters Freeze Mode if the FRZ bit is asserted. 0 No Freeze Mode request.
27 NOT_RDY	This read-only bit indicates that FLEXCAN is either in Disable Mode, Stop Mode or Freeze Mode. It is negated once FLEXCAN has exited these modes. 1 FLEXCAN module is either in Disable Mode, Stop Mode or Freeze Mode 0 FLEXCAN module is either in Normal Mode, Listen-Only Mode or Loop-Back Mode
26 WAK_MSK	This bit enables the Wake Up Interrupt generation. 1 Wake Up Interrupt is enabled 0 Wake Up Interrupt is disabled
25 SOFT_RST	When this bit is asserted, FlexCAN resets its internal state machines and some of the memory mapped registers. The following registers are reset: MCR (except the MDIS bit), TIMER, ECR, ESR1, ESR2, IMASK1, IMASK2, IFLAG1, IFLAG2 and CRCR. Configuration registers that control the interface to the CAN bus are not affected by soft reset. The following registers are unaffected: CTRL1, CTRL2, RXIMR0_RXIMR63, RXGMASK, RX14MASK, RX15MASK, RXFGMASK, RXFIR and all Message Buffers The SOFT_RST bit can be asserted directly by the Arm when it writes to the MCR Register. It may take some time to fully propagate its effect. The SOFT_RST bit remains asserted while reset is pending, and is automatically negated when reset completes. Therefore, software can poll this bit to know when the soft reset has completed. Soft reset cannot be applied while clocks are shut down in any of the low power modes. The module should be first removed from low power mode, and then soft reset can be applied.

Table continues on the next page...

FLEXCANx_MCR field descriptions (continued)

Field	Description
	1 Reset the registers 0 No reset request
24 FRZ_ACK	This read-only bit indicates that FLEXCAN is in Freeze Mode and its prescaler is stopped. The Freeze Mode request cannot be granted until current transmission or reception processes have finished. Therefore the software can poll the FRZ_ACK bit to know when FLEXCAN has actually entered Freeze Mode. If Freeze Mode request is negated, then this bit is negated once the FLEXCAN prescaler is running again. If Freeze Mode is requested while FLEXCAN is in any of the low power modes, then the FRZ_ACK bit will only be set when the low power mode is exited. See Freeze Mode for more information 1 FLEXCAN in Freeze Mode, prescaler stopped 0 FLEXCAN not in Freeze Mode, prescaler running
23 SUPV	This bit configures some of the FLEXCAN registers to be either in Supervisor or User Mode. Reset value of this bit is '1', so the affected registers start with Supervisor access allowance only. This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 1 FlexCAN is in Supervisor Mode. Affected registers allow only Supervisor access. Unrestricted access behaves as though the access was done to an unimplemented register location 0 FlexCAN is in User Mode. Affected registers allow both Supervisor and Unrestricted accesses
22 SLF_WAK	This bit enables the Self Wake Up feature when FLEXCAN is in Stop Mode. If this bit had been asserted by the time FLEXCAN entered Stop Mode, then FLEXCAN will look for a recessive to dominant transition on the bus during these modes. If a transition from recessive to dominant is detected during Stop Mode, then FLEXCAN generates, if enabled to do so, a Wake Up interrupt to the Arm so that it can resume the clocks globally and FlexCAN can request to resume the clocks. This bit can not be written while the module is in Stop Mode. 1 FLEXCAN Self Wake Up feature is enabled 0 FLEXCAN Self Wake Up feature is disabled
21 WRN_EN	When asserted, this bit enables the generation of the TWRN_INT and RWRN_INT flags in the Error and Status Register. If WRN_EN is negated, the TWRN_INT and RWRN_INT flags will always be zero, independent of the values of the error counters, and no warning interrupt will ever be generated.This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 1 TWRN_INT and RWRN_INT bits are set when the respective error counter transition from <96 to ≥ 96. 0 TWRN_INT and RWRN_INT bits are zero, independent of the values in the error counters.
20 LPM_ACK	This read-only bit indicates that FLEXCAN is either in Disable Mode or Stop Mode. Either of these low power modes can not be entered until all current transmission or reception processes have finished, so the Arm can poll the LPM_ACK bit to know when FLEXCAN has actually entered low power mode. See Module Disable Mode, and Stop Mode for more information 1 FLEXCAN is either in Disable Mode, or Stop mode 0 FLEXCAN not in any of the low power modes
19 WAK_SRC	This bit defines whether the integrated low-pass filter is applied to protect the FLEXCAN_RX input from spurious wake up. See Stop Mode for more information. This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 1 FLEXCAN uses the filtered FLEXCAN_RX input to detect recessive to dominant edges on the CAN bus 0 FLEXCAN uses the unfiltered FLEXCAN_RX input to detect recessive to dominant edges on the CAN bus.
18 -	This field is reserved. Reserved

Table continues on the next page...

FLEXCANx_MCR field descriptions (continued)

Field	Description
17 SRX_DIS	This bit defines whether FlexCAN is allowed to receive frames transmitted by itself. If this bit is asserted, frames transmitted by the module will not be stored in any MB, regardless if the MB is programmed with an ID that matches the transmitted frame, and no interrupt flag or interrupt signal will be generated due to the frame reception. This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 1 Self reception disabled 0 Self reception enabled
16 IRMQ	This bit indicates whether Rx matching process will be based either on individual masking and queue or on masking scheme with RXMGMASK, RX14MASK and RX15MASK, RXFGMASK. This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 1 Individual Rx masking and queue feature are enabled. 0 Individual Rx masking and queue feature are disabled.For backward compatibility, the reading of C/S word locks the MB even if it is EMPTY.
15–14 -	This field is reserved. Reserved
13 LPRIO_EN	This bit is provided for backwards compatibility reasons. It controls whether the local priority feature is enabled or not. It is used to extend the ID used during the arbitration process. With this extended ID concept, the arbitration process is done based on the full 32-bit word, but the actual transmitted ID still has 11-bit for standard frames and 29-bit for extended frames.This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 1 Local Priority enabled 0 Local Priority disabled
12 AEN	This bit is supplied for backwards compatibility reasons. When asserted, it enables the Tx abort feature. This feature guarantees a safe procedure for aborting a pending transmission, so that no frame is sent in the CAN bus without notification. This bit can only be written in Freeze mode as it is blocked by hardware in other modes.Write Abort code into Rx Mailboxes can cause unpredictable results when the MCR[AEN] is asserted. 1 Abort enabled 0 Abort disabled
11–10 -	This field is reserved. Reserved
9–8 IDAM	This 2-bit field identifies the format of the elements of the Rx FIFO filter table, as shown below. Note that all elements of the table are configured at the same time by this field (they are all the same format). See Rx FIFO Structure. This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 00 Format A One full ID (standard or extended) per ID filter Table element. 01 Format B Two full standard IDs or two partial 14-bit extended IDs per ID filter Table element. 10 Format C Four partial 8-bit IDs (standard or extended) per ID filter Table element. 11 Format D All frames rejected.
7 -	This field is reserved. Reserved
MAXMB	This 7-bit field defines the number of the last Message Buffers that will take part in the matching and arbitration processes. The reset value (0x0F) is equivalent to 16 MB configuration. This field can only be written in Freeze Mode as it is blocked by hardware in other modes Number of the last MB = MAXMB.

Table continues on the next page...

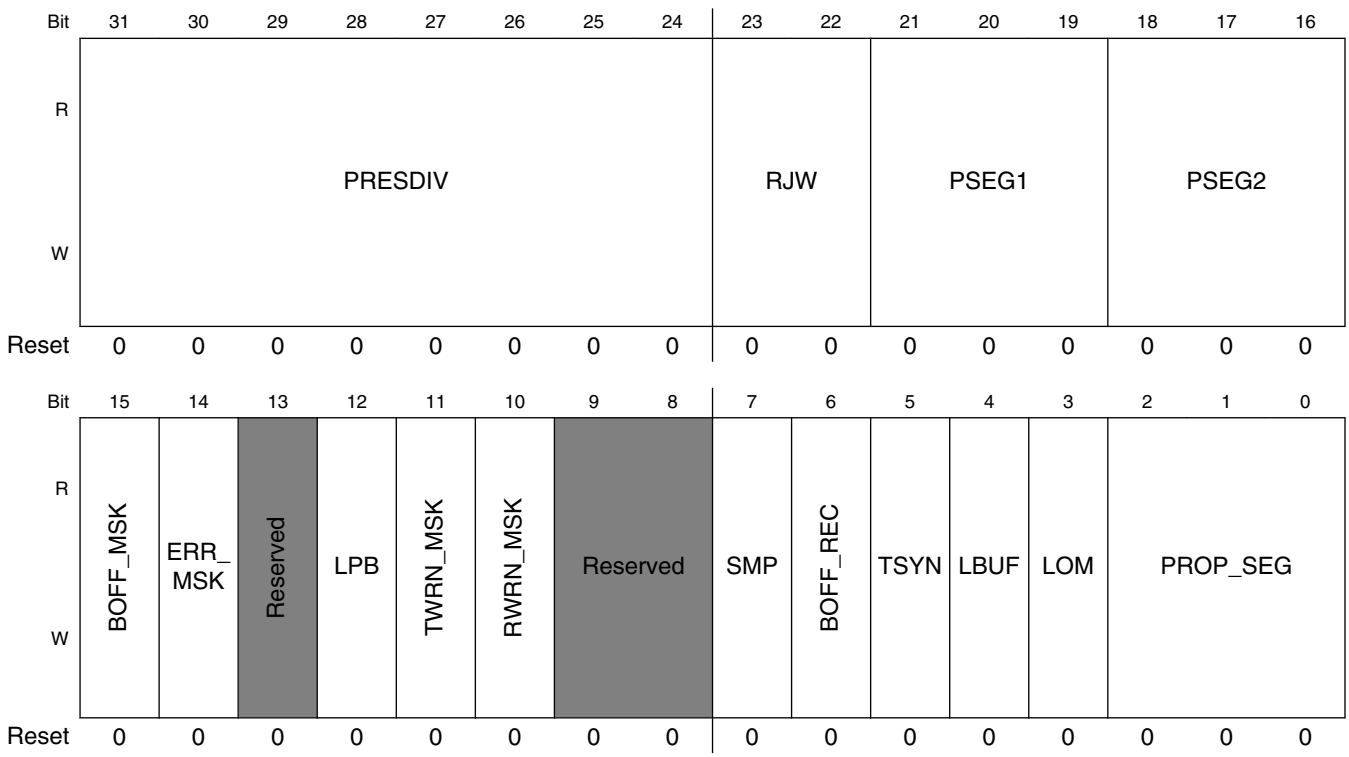
FLEXCANx_MCR field descriptions (continued)

Field	Description
	NOTE: Additionally, the value of MAXMB must encompass the FIFO size defined by CTRL2[RFFN] MAXMB also impacts the definition of the minimum number of peripheral clocks per CAN bit as described in Table 44-19 (see Arbitration and Matching Timing).

44.9.2 Control 1 Register (FLEXCANx_CTRL1)

This register is defined for specific FLEXCAN control features related to the CAN bus, such as bit-rate, programmable sampling point within an Rx bit, Loop Back Mode, Listen Only Mode, Bus Off recovery behavior and interrupt enabling (Bus-Off, Error, Warning). It also determines the Division Factor for the clock prescaler.

Address: Base address + 4h offset



FLEXCANx_CTRL1 field descriptions

Field	Description
31–24 PRESDIV	This 8-bit field defines the ratio between the PE clock frequency and the Serial Clock (Sclock) frequency. The Sclock period defines the time quantum of the CAN protocol. For the reset value, the Sclock frequency is equal to the PE clock frequency. The Maximum value of this register is 0xFF, that gives a minimum Sclock frequency equal to the PE clock frequency divided by 256.For more information refer to Protocol Timing. This field can only be written in Freeze mode as it is blocked by hardware in other modes.

Table continues on the next page...

FLEXCANx_CTRL1 field descriptions (continued)

Field	Description
	Sclock frequency = CPI clock frequency / (PRESDIV+1)
23–22 RJW	This 2-bit field defines the maximum number of time quanta[1] that a bit time can be changed by one re-synchronization. The valid programmable values are 0-3. This field can only be written in Freeze mode as it is blocked by hardware in other modes Resync Jump Width = RJW + 1.
21–19 PSEG1	This 3-bit field defines the length of Phase Buffer Segment 1 in the bit time . The valid programmable values are 0-7. This field can only be written in Freeze mode as it is blocked by hardware in other modes Phase Buffer Segment 1 = (PSEG1 + 1) x Time-Quanta.
18–16 PSEG2	This 3-bit field defines the length of Phase Buffer Segment 2 in the bit time . The valid programmable values are 1-7. This field can only be written in Freeze mode as it is blocked by hardware in other modes Phase Buffer Segment 2 = (PSEG2 + 1) x Time-Quanta.
15 BOFF_MSK	This bit provides a mask for the Bus Off Interrupt. 1 Bus Off interrupt enabled 0 Bus Off interrupt disabled
14 ERR_MSK	This bit provides a mask for the Error Interrupt. 1 Error interrupt enabled 0 Error interrupt disabled
13 -	This field is reserved. Reserved
12 LPB	This bit configures FlexCAN to operate in Loop-Back Mode. In this mode, FlexCAN performs an internal loop back that can be used for self test operation. The bit stream output of the transmitter is fed back internally to the receiver input. The FLEXCAN_RX input pin is ignored and the FLEXCAN_TX output goes to the recessive state (logic '1'). FlexCAN behaves as it normally does when transmitting, and treats its own transmitted message as a message received from a remote node. In this mode, FlexCAN ignores the bit sent during the ACK slot in the CAN frame acknowledge field, generating an internal acknowledge bit to ensure proper reception of its own message. Both transmit and receive interrupts are generated. This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 1 Loop Back enabled 0 Loop Back disabled
11 TWRN_MSK	This bit provides a mask for the Tx Warning Interrupt associated with the TWRN_INT flag in the Error and Status Register. This bit is read as zero when MCR[WRN_EN] bit is negated. This bit can only be written if MCR[WRN_EN] bit is asserted. 1 Tx Warning Interrupt enabled 0 Tx Warning Interrupt disabled
10 RWRN_MSK	This bit provides a mask for the Rx Warning Interrupt associated with the RWRN_INT flag in the Error and Status Register. This bit is read as zero when MCR[WRN_EN] bit is negated. This bit can only be written if MCR[WRN_EN] bit is asserted. 1 Rx Warning Interrupt enabled 0 Rx Warning Interrupt disabled
9–8 -	This field is reserved. Reserved
7 SMP	This bit defines the sampling mode of CAN bits at the FLEXCAN_RX. This bit can only be written in Freeze mode as it is blocked by hardware in other modes.

Table continues on the next page...

FLEXCANx_CTRL1 field descriptions (continued)

Field	Description
	1 Three samples are used to determine the value of the received bit: the regular one (sample point) and 2 preceding samples, a majority rule is used 0 Just one sample is used to determine the bit value
6 BOFF_REC	This bit defines how FLEXCAN recovers from Bus Off state. If this bit is negated, automatic recovering from Bus Off state occurs according to the CAN Specification 2.0B. If the bit is asserted, automatic recovering from Bus Off is disabled and the module remains in Bus Off state until the bit is negated by the user. If the negation occurs before 128 sequences of 11 recessive bits are detected on the CAN bus, then Bus Off recovery happens as if the BOFF_REC bit had never been asserted. If the negation occurs after 128 sequences of 11 recessive bits occurred, then FLEXCAN will re-synchronize to the bus by waiting for 11 recessive bits before joining the bus. After negation, the BOFF_REC bit can be re-asserted again during Bus Off, but it will only be effective the next time the module enters Bus Off. If BOFF_REC was negated when the module entered Bus Off, asserting it during Bus Off will not be effective for the current Bus Off recovery. 1 Automatic recovering from Bus Off state disabled 0 Automatic recovering from Bus Off state enabled, according to CAN Spec 2.0 part B
5 TSYN	This bit enables a mechanism that resets the free-running timer each time a message is received in Message Buffer 0. This feature provides means to synchronize multiple FLEXCAN stations with a special "SYNC" message (i.e., global network time). If the RFEN bit in MCR is set (FIFO enabled), the first available Mailbox, according to CTRL2[RFFN] setting, is used for timer synchronization instead of MB0.This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 1 Timer Sync feature enabled 0 Timer Sync feature disabled
4 LBUF	This bit defines the ordering mechanism for Message Buffer transmission. When asserted, the LPRIO_EN bit does not affect the priority arbitration.This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 1 Lowest number buffer is transmitted first 0 Buffer with highest priority is transmitted first
3 LOM	This bit configures FLEXCAN to operate in Listen Only Mode. In this mode, transmission is disabled, all error counters are frozen and the module operates in a CAN Error Passive mode. Only messages acknowledged by another CAN station will be received. If FLEXCAN detects a message that has not been acknowledged, it will flag a BIT0 error (without changing the REC), as if it was trying to acknowledge the message. Listen-Only Mode acknowledgement can be obtained by the state of ESR1[FLT_CONF] field which is Passive Error when Listen-Only Mode is entered. There can be some delay between the Listen-Only Mode request and acknowledge. This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 1 FLEXCAN module operates in Listen Only Mode 0 Listen Only Mode is deactivated
PROP_SEG	This 3-bit field defines the length of the Propagation Segment in the bit time. The valid programmable values are 0-7. This field can only be written in Freeze mode as it is blocked by hardware in other modes Propagation Segment Time = (PROPSEG + 1) * Time-Quanta. Time-Quantum = one Sclock period.

1. One time quantum is equal to the Sclock period.

44.9.3 Free Running Timer Register (FLEXCANx_TIMER)

This register represents a 16-bit free running counter that can be read and written by the Arm. The timer starts from $0000 after Reset, counts linearly to $FFFF, and wraps around.

The timer is clocked by the FLEXCAN bit-clock (which defines the baud rate on the CAN bus). During a message transmission/reception, it increments by one for each bit that is received or transmitted. When there is no message on the bus, it counts using the previously programmed baud rate. During Freeze Mode, disable, and stop mode, the timer is not incremented.

The timer value is captured at the beginning of the identifier field of any frame on the CAN bus. This captured value is written into the Time Stamp entry in a message buffer after a successful reception or transmission of a message.

If bit CTRL1[TSYN] is asserted the Timer is reset whenever a message is received in the first available Mailbox, according to CTRL2[RFFN] setting.

Arm can write to this register anytime. However, if the write occurs at the same time that the Timer is being reset by a reception in the first Mailbox, then the write value is discarded.

Reading this register affects the Mailbox Unlocking procedure. For additional details, refer to Message Buffer Lock Mechanism.

Address: Base address + 8h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																TIMER															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FLEXCANx_TIMER field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
TIMER	TIMER

44.9.4 Rx Mailboxes Global Mask Register (FLEXCANx_RXMGMASK)

RXMGMASK is provided for legacy support. Asserting the MCR[IRMQ] bit causes the RXMGMASK Register to have no effect on the module operation.

RXMGMASK is used to mask the filter fields of all Rx MBs, excluding MBs 14-15, which have individual mask registers.

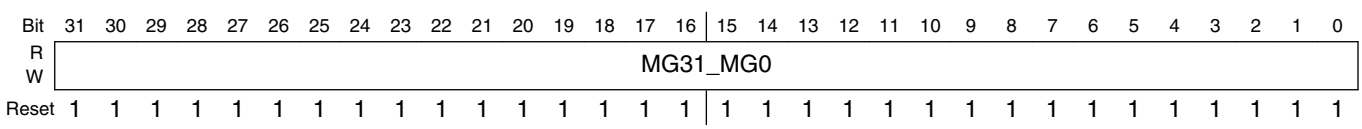
This register can only be written in Freeze mode as it is blocked by hardware in other modes.

Table 44-20. Rx Mailboxes Global Mask usage

SMB[RTR][1]	CTRL2[RRS]	CTRL2[EACEN]	Mailbox filter fields			
			MB[RTR]	MB[IDE]	MB[ID]	reserved
0	-	0	- Note[2]	- Note[3]	MG[28:0]	MG[31:29]
0	-	1	MG[31]	MG[30]	MG[28:0]	MG[29]
1	0	-	-	-	-	MG[31:0]
1	1	0	-	-	MG[28:0]	MG[31:29]
1	1	1	MG[31]	MG[30]	MG[28:0]	MG[29]

1. RTR bit of the Incoming Frame. It is saved into an auxiliary MB called Rx Serial Message Buffer (Rx SMB).
2. If CTRL2[EACEN] bit is negated the RTR bit of Mailbox is never compared with the RTR bit of the Incoming Frame (Rx SMB[RTR]).
3. If CTRL2[EACEN] bit is negated the IDE bit of Mailbox is always compared with the IDE bit of the Incoming Frame (Rx SMB[IDE]).

Address: Base address + 10h offset



FLEXCANx_RXMGMASK field descriptions

Field	Description
MG31_MG0	These bits mask the Mailbox filter bits as shown in the figure above. Note that the alignment with the ID word of the Mailbox is not perfect as the two most significant MG bits affect the fields RTR and IDE which are located in the Control and Status word of the Mailbox. Rx Mailboxes Global Mask Register (FLEXCAN_RXMGMASK) shows in detail which MG bits mask each Mailbox filter field. 1 The corresponding bit in the filter is checked against the one received 0 the corresponding bit in the filter is "don't care"

44.9.5 Rx Buffer 14 Mask Register (FLEXCANx_RX14MASK)

RX14MASK is provided for legacy support, asserting the MCR[IRMQ] bit causes the RX14MASK to have no effect on the module operation.

RX14MASK is used to mask the filter fields of Message Buffer 14.

This register can only be programmed while the module is in Freeze Mode as it is blocked by hardware in other modes.

Address: Base address + 14h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RX14M31_RX14M0																															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

FLEXCANx_RX14MASK field descriptions

Field	Description
RX14M31_ RX14M0	These bits mask Mailbox 14 filter bits in the same fashion as RXMGMASK masks other Mailboxes filters (see Rx Mailboxes Global Mask Register (FLEXCAN_RXMGMASK)) 1 The corresponding bit in the filter is checked 0 the corresponding bit in the filter is "don't care"

44.9.6 Rx Buffer 15 Mask Register (FLEXCANx_RX15MASK)

RX15MASK is provided for legacy support, asserting the MCR[IRMQ] bit causes the RX15MASK Register to have no effect on the module operation.

RX15MASK is used to mask the filter fields of Message Buffer 15.

This register can only be programmed while the module is in Freeze Mode as it is blocked by hardware in other modes.

Address: Base address + 18h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	RX15M31_RX15M0																															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

FLEXCANx_RX15MASK field descriptions

Field	Description
RX15M31_ RX15M0	These bits mask Mailbox 15 filter bits in the same fashion as RXMGMASK masks other Mailboxes filters (see Rx Mailboxes Global Mask Register (FLEXCAN_RXMGMASK)). 1 The corresponding bit in the filter is checked 0 the corresponding bit in the filter is "don't care"

44.9.7 Error Counter Register (FLEXCANx_ECR)

This register has 2 8-bit fields reflecting the value of two FLEXCAN error counters: Transmit Error Counter (Tx_Err_Counter field) and Receive Error Counter (Rx_Err_Counter field). The rules for increasing and decreasing these counters are described in the CAN protocol and are completely implemented in the FLEXCAN module. Both counters are read only except in Freeze Mode, where they can be written by the Arm.

FLEXCAN responds to any bus state as described in the protocol, e.g. transmit 'Error Active' or 'Error Passive' flag, delay its transmission start time ('Error Passive') and avoid any influence on the bus when in 'Bus Off' state. The following are the basic rules for FLEXCAN bus state transitions.

- If the value of Tx_Err_Counter or Rx_Err_Counter increases to be greater than or equal to 128, the FLT_CONF field in the Error and Status Register is updated to reflect 'Error Passive' state.
- If the FLEXCAN state is 'Error Passive', and either Tx_Err_Counter or Rx_Err_Counter decrements to a value less than or equal to 127 while the other already satisfies this condition, the FLT_CONF field in the Error and Status Register is updated to reflect 'Error Active' state.
- If the value of Tx_Err_Counter increases to be greater than 255, the FLT_CONF field in the Error and Status Register is updated to reflect 'Bus Off' state, and an interrupt may be issued. The value of Tx_Err_Counter is then reset to zero.
- If FLEXCAN is in 'Bus Off' state, then Tx_Err_Counter is cascaded together with another internal counter to count the 128th occurrences of 11 consecutive recessive bits on the bus. Hence, Tx_Err_Counter is reset to zero and counts in a manner where the internal counter counts 11 such bits and then wraps around while incrementing the Tx_Err_Counter. When Tx_Err_Counter reaches the value of 128, the FLT_CONF field in the Error and Status Register is updated to be 'Error Active' and both error counters are reset to zero. At any instance of dominant bit following a stream of less than 11 consecutive recessive bits, the internal counter resets itself to zero without affecting the Tx_Err_Counter value.
- If during system start-up, only one node is operating, then its Tx_Err_Counter increases in each message it is trying to transmit, as a result of acknowledge errors (indicated by the ACK_ERR bit in the Error and Status Register). After the transition to 'Error Passive' state, the Tx_Err_Counter does not increment anymore by acknowledge errors. Therefore the device never goes to the 'Bus Off' state.
- If the Rx_Err_Counter increases to a value greater than 127, it is not incremented further, even if more errors are detected while being a receiver. At the next

successful message reception, the counter is set to a value between 119 and 127 to resume to 'Error Active' state.

Address: Base address + 1Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved																RX_ERR_COUNTER								TX_ERR_COUNTER							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FLEXCANx_ECR field descriptions

Field	Description
31–16 -	This field is reserved. Reserved
15–8 RX_ERR_ COUNTER	Rx_Err_Counter
TX_ERR_ COUNTER	Tx_Err_Counter

44.9.8 Error and Status 1 Register (FLEXCANx_ESR1)

This register reflects various error conditions, some general status of the device and it is the source of four interrupts to the Arm.

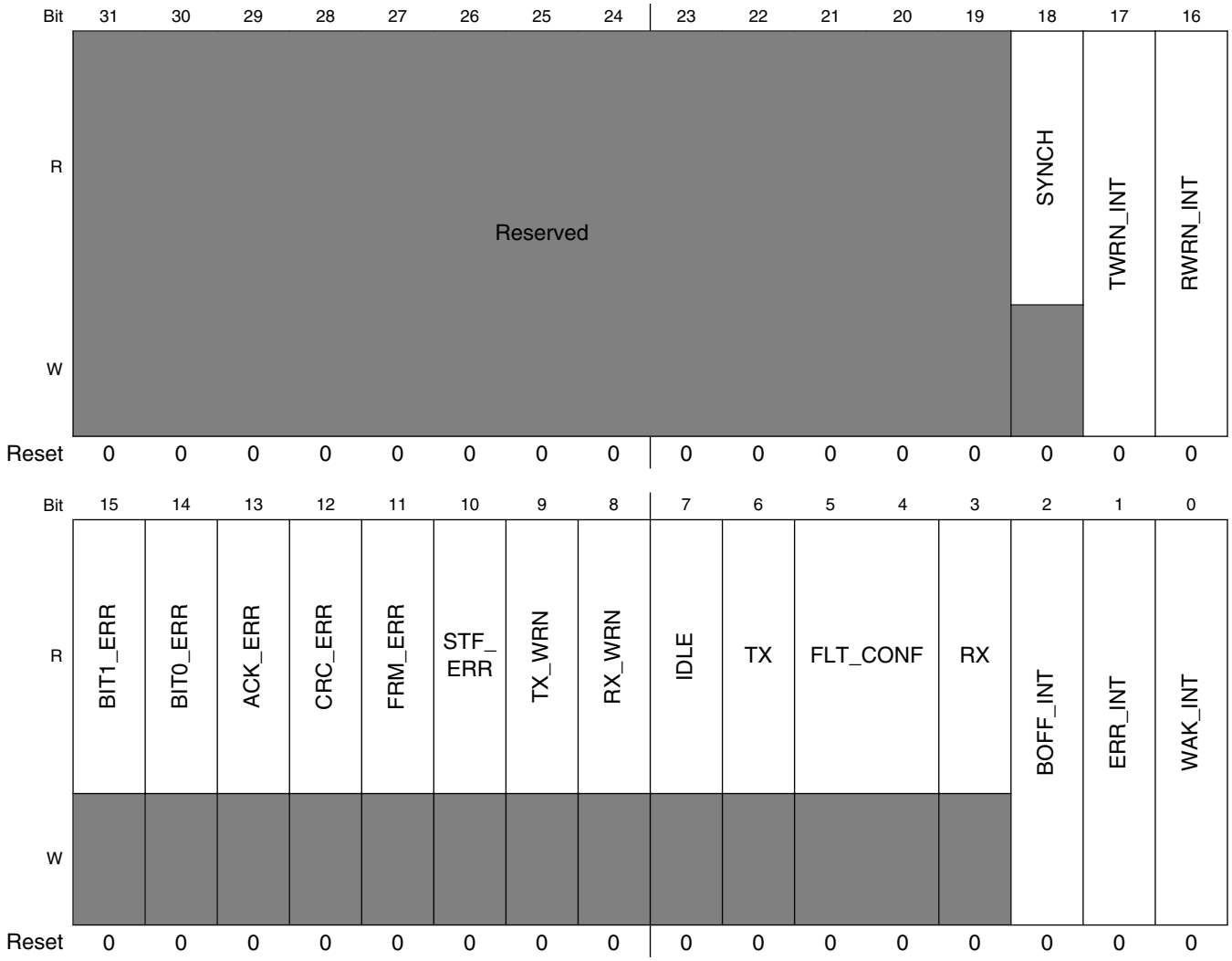
The Arm read action clears bits 15-10, therefore the reported *error conditions*(bits 15-10) are those that occurred since the last time the Arm read this register. Bits 9-3 are status bits .

Some bits in this register are read-only and some are not .

Table 44-21. FlexCAN State

SYNCH	IDLE	TX	RX	FlexCAN state
0	0	0	0	Not synchronized to CAN bus
1	1	x	x	Idle
1	0	1	0	Transmitting
1	0	0	1	Receiving
other combinations				Reserved

Address: Base address + 20h offset



FLEXCANx_ESR1 field descriptions

Field	Description
31–19 -	This field is reserved. Reserved
18 SYNCH	This read-only flag indicates whether the FlexCAN is synchronized to the CAN bus and able to participate in the communication process. It is set and cleared by the FlexCAN. Refer to Table 44-21 1 FlexCAN is synchronized to the CAN bus 0 FlexCAN is not synchronized to the CAN bus
17 TWRN_INT	If the WRN_EN bit in MCR is asserted, the TWRN_INT bit is set when the TX_WRN flag transition from '0' to '1', meaning that the Tx error counter reached 96. If the corresponding mask bit in the Control Register (TWRN_MSK) is set, an interrupt is generated to the Arm. This bit is cleared by writing it to '1'. When WRN_EN is negated, this flag is masked. Arm must clear this flag before disabling the bit. Otherwise it will be set when the WRN_EN is set again. Writing '0' has no effect. This flag is not generated during "Bus Off" state. This bit is not updated during Freeze mode. 1 The Tx error counter transition from < 96 to >= 96 0 No such occurrence

Table continues on the next page...

FLEXCANx_ESR1 field descriptions (continued)

Field	Description
16 RWRN_INT	If the WRN_EN bit in MCR is asserted, the RWRN_INT bit is set when the RX_WRN flag transition from '0' to '1', meaning that the Rx error counters reached 96. If the corresponding mask bit in the Control Register (RWRN_MSK) is set, an interrupt is generated to the Arm. This bit is cleared by writing it to '1'. When WRN_EN is negated, this flag is masked. Arm must clear this flag before disabling the bit. Otherwise it will be set when the WRN_EN is set again. Writing '0' has no effect. This bit is not updated during Freeze mode. 1 The Rx error counter transition from < 96 to >= 96 0 No such occurrence
15 BIT1_ERR	This bit indicates when an inconsistency occurs between the transmitted and the received bit in a message. This bit is not set by a transmitter in case of arbitration field or ACK slot, or in case of a node sending a passive error flag that detects dominant bits. 1 At least one bit sent as recessive is received as dominant 0 No such occurrence
14 BIT0_ERR	This bit indicates when an inconsistency occurs between the transmitted and the received bit in a message. 1 At least one bit sent as dominant is received as recessive 0 No such occurrence
13 ACK_ERR	This bit indicates that an Acknowledge Error has been detected by the transmitter node, i.e., a dominant bit has not been detected during the ACK SLOT. 1 An ACK error occurred since last read of this register 0 No such occurrence
12 CRC_ERR	This bit indicates that a CRC Error has been detected by the receiver node, i.e., the calculated CRC is different from the received. 1 A CRC error occurred since last read of this register. 0 No such occurrence
11 FRM_ERR	This bit indicates that a Form Error has been detected by the receiver node, i.e., a fixed-form bit field contains at least one illegal bit. 1 A Form Error occurred since last read of this register 0 No such occurrence
10 STF_ERR	This bit indicates that a Stuffing Error has been detected. 1 A Stuffing Error occurred since last read of this register. 0 No such occurrence.
9 TX_WRN	This bit indicates when repetitive errors are occurring during message transmission. 1 TX_Err_Counter ≥ 96 0 No such occurrence
8 RX_WRN	This bit indicates when repetitive errors are occurring during message reception. 1 Rx_Err_Counter ≥ 96 0 No such occurrence
7 IDLE	This bit indicates when CAN bus is in IDLE state.Refer to Table 44-21.

Table continues on the next page...

FLEXCANx_ESR1 field descriptions (continued)

Field	Description
	1 CAN bus is now IDLE 0 No such occurrence
6 TX	This bit indicates if FLEXCAN is transmitting a message.Refer to Table 44-21. 1 FLEXCAN is transmitting a message 0 FLEXCAN is receiving a message
5–4 FLT_CONF	If the LOM bit in the Control Register is asserted, after some delay that depends on the CAN bit timing the FLT_CONF field will indicate "Error Passive". The very same delay affects the way how FLT_CONF reflects an update to ECR register by the Arm. It may be necessary up to one CAN bit time to get them coherent again. Since the Control Register is not affected by soft reset, the FLT_CONF field will not be affected by soft reset if the LOM bit is asserted. This 2-bit field indicates the Confinement State of the FLEXCAN module, as shown in below: 00 Error Active 01 Error Passive 1x Bus off
3 RX	This bit indicates if FlexCAN is receiving a message. Refer to Table 44-21. 1 FLEXCAN is transmitting a message 0 FLEXCAN is receiving a message
2 BOFF_INT	This bit is set when FLEXCAN enters 'Bus Off' state. If the corresponding mask bit in the Control Register (BOFF_MSK) is set, an interrupt is generated to the Arm. This bit is cleared by writing it to '1'. Writing '0' has no effect. 1 FLEXCAN module entered 'Bus Off' state 0 No such occurrence
1 ERR_INT	This bit indicates that at least one of the Error Bits (bits 15-10) is set. If the corresponding mask bit in the Control Register (ERR_MSK) is set, an interrupt is generated to the Arm. This bit is cleared by writing it to '1'.Writing '0' has no effect. 1 Indicates setting of any Error Bit in the Error and Status Register 0 No such occurrence
0 WAK_INT	When FLEXCAN is Stop Mode and a recessive to dominant transition is detected on the CAN bus and if the WAK_MSK bit in the MCR Register is set, an interrupt is generated to the Arm. This bit is cleared by writing it to '1'. When SLF_WAK is negated, this flag is masked. Arm must clear this flag before disabling the bit. Otherwise it will be set when the SLF_WAK is set again. Writing '0' has no effect 1 Indicates a recessive to dominant transition received on the CAN bus when the FLEXCAN module is in Stop Mode 0 No such occurrence

44.9.9 Interrupt Masks 2 Register (FLEXCANx_IMASK2)

This register allows any number of a range of 32 Message Buffer Interrupts to be enabled or disabled. It contains one interrupt mask bit per buffer, enabling the Arm to determine which buffer generates an interrupt after a successful transmission or reception (i.e. when the corresponding IFLAG2 bit is set).

Address: Base address + 24h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BUF63M_BUF32M																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FLEXCANx_IMASK2 field descriptions

Field	Description
BUF63M_ BUF32M	Each bit enables or disables the respective FLEXCAN Message Buffer (MB32 to MB63) Interrupt. Setting or clearing a bit in the IMASK2 Register can assert or negate an interrupt request, if the corresponding IFLAG2 bit is set. 1 The corresponding buffer Interrupt is enabled 0 The corresponding buffer Interrupt is disabled

44.9.10 Interrupt Masks 1 Register (FLEXCANx_IMASK1)

This register allows to enable or disable any number of a range of 32 Message Buffer Interrupts. It contains one interrupt mask bit per buffer, enabling the Arm to determine which buffer generates an interrupt after a successful transmission or reception (i.e., when the corresponding IFLAG1 bit is set).

Address: Base address + 28h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BUF31M_BUF0M																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FLEXCANx_IMASK1 field descriptions

Field	Description
BUF31M_BUF0M	Each bit enables or disables the respective FLEXCAN Message Buffer (MB0 to MB31) Interrupt. Setting or clearing a bit in the IMASK1 Register can assert or negate an interrupt request, if the corresponding IFLAG1 bit is set

FLEXCANx_IMASK1 field descriptions (continued)

Field	Description
	1 The corresponding buffer Interrupt is enabled 0 The corresponding buffer Interrupt is disabled

44.9.11 Interrupt Flags 2 Register (FLEXCANx_IFLAG2)

This register defines the flags for 32 Message Buffer interrupts. It contains one interrupt flag bit per buffer. Each successful transmission or reception sets the corresponding IFLAG2 bit. If the corresponding IMASK2 bit is set, an interrupt will be generated. The interrupt flag must be cleared by writing it to '1'. Writing '0' has no effect.Before updating MCR[MAXMB] field, Arm must treat the IFLAG2 bits which MB value is greater than the MCR[MAXMB] to be updated, otherwise they will keep set and be inconsistent with the amount of MBs available.

Address: Base address + 2Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BUF63I_BUF32I																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FLEXCANx_IFLAG2 field descriptions

Field	Description
BUF63I_BUF32I	Each bit flags the respective FLEXCAN Message Buffer (MB32 to MB63) interrupt. 1 The corresponding buffer has successfully completed transmission or reception 0 No such occurrence

44.9.12 Interrupt Flags 1 Register (FLEXCANx_IFLAG1)

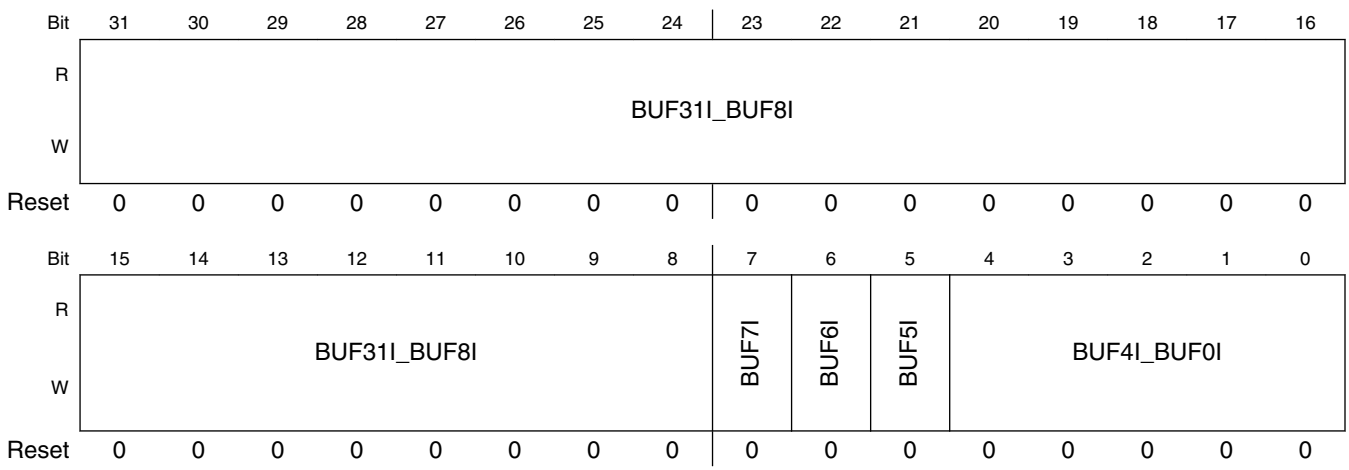
This register defines the flags for 32 Message Buffer interrupts and FIFO interrupts. It contains one interrupt flag bit per buffer. Each successful transmission or reception sets the corresponding IFLAG1 bit. If the corresponding IMASK1 bit is set, an interrupt will be generated. The Interrupt flag must be cleared by writing it to '1'. Writing '0' has no effect.

When the RFEN bit in the MCR is set (Rx FIFO enabled), the function of the 8 least significant interrupt flags (BUF7I - BUF0I) is changed to support the FIFO operation. BUF7I, BUF6I and BUF5I indicate operating conditions of the FIFO, while BUF4I to BUF0I are not used. Before enabling the RFEN, Arm must service the IFLAGS asserted

in the Rx FIFO region (see Rx FIFO). Otherwise, these IFLAGS will mistakenly show the related MBs now belonging to FIFO as having contents to be serviced. When the RFEN is negated, the FIFO flags must be cleared. The same care must be taken when a RFFN value is selected extending Rx FIFO filters beyond MB7 (see Control 2 Register (FLEXCAN_CTRL2)). For example, when RFFN is 0x8, the MB0-23 range is occupied by Rx FIFO filters and related IFLAGS must be cleared.

Before updating MCR[MAXMB] field, Arm must service the IFLAG1 which MB value is greater than the MCR[MAXMB] to be updated, otherwise they will keep set and be inconsistent with the amount of MBs available.

Address: Base address + 30h offset



FLEXCANx_IFLAG1 field descriptions

Field	Description
31–8 BUF31I_BUF8I	Each bit flags the respective FLEXCAN Message Buffer (MB8 to MB31) interrupt. 1 The corresponding MB has successfully completed transmission or reception 0 No such occurrence
7 BUF7I	If the Rx FIFO is not enabled, this bit flags the interrupt for MB7. If the MCR[RFEN] bit is asserted, this flag indicates that a message was lost because Rx FIFO is full. Note that the flag will not be asserted when the Rx FIFO is full and the message was captured by a Mailbox. This flag is cleared by the FlexCAN whenever the bit MCR[RFEN] is changed by Arm writes. 1 MB7 completed transmission/reception or FIFO overflow 0 No such occurrence
6 BUF6I	If the Rx FIFO is not enabled, this bit flags the interrupt for MB6. If the MCR[RFEN] bit is asserted, this flag indicates when the number of unread messages within the Rx FIFO is increased to 5 from 4 due to the reception of a new one, meaning that the Rx FIFO is almost full. Note that if the flag is cleared while the number of unread messages is greater than 4 it will not assert again until the number of unread messages within the Rx FIFO is decreased to equal or less than 4. This flag is cleared by the FlexCAN whenever the bit MCR[RFEN] is changed by Arm writes.

Table continues on the next page...

FLEXCANx_IFLAG1 field descriptions (continued)

Field	Description
	1 MB6 completed transmission/reception or FIFO almost full 0 No such occurrence
5 BUF5I	If the Rx FIFO is not enabled, this bit flags the interrupt for MB5. If the Rx FIFO is enabled, this flag indicates that at least one frame is available to be read from the Rx FIFO. This flag is cleared by the FlexCAN whenever the bit MCR[RFEN] is changed by Arm writes. 1 MB5 completed transmission/reception or frames available in the FIFO 0 No such occurrence
BUF4I_BUF0I	If the Rx FIFO is not enabled, these bits flag the interrupts for MB0 to MB4 . If the Rx FIFO is enabled, these flags are not used and must be considered as reserved locations. These flags are cleared by the FlexCAN whenever the bit MCR[RFEN] is changed by Arm writes. 1 Corresponding MB completed transmission/reception 0 No such occurrence

44.9.13 Control 2 Register (FLEXCANx_CTRL2)

This register contains control bits for CAN errors, FIFO features and mode selection.

Table 44-22. Rx FIFO Filters

RFFN[3:0]	Number of Rx FIFO filters	Message Buffers occupied by Rx FIFO and ID Filter Table	Remaining Available Mailboxes[1]	Rx FIFO ID Filter Table Elements Affected by Rx Individual Masks[2]	Rx FIFO ID Filter Table Elements Affected by Rx FIFO Global Mask[2]
0x0	8	MB 0-7	MB 8-63	Elements 0-7	none
0x1	16	MB 0-9	MB 10-63	Elements 0-9	Elements 10-15
0x2	24	MB 0-11	MB 12-63	Elements 0-11	Elements 12-23
0x3	32	MB 0-13	MB 14-63	Elements 0-13	Elements 14-31
0x4	40	MB 0-15	MB 16-63	Elements 0-15	Elements 16-39
0x5	48	MB 0-17	MB 18-63	Elements 0-17	Elements 18-47
0x6	56	MB 0-19	MB 20-63	Elements 0-19	Elements 20-55
0x7	64	MB 0-21	MB 22-63	Elements 0-21	Elements 22-63
0x8	72	MB 0-23	MB 24-63	Elements 0-23	Elements 24-71
0x9	80	MB 0-25	MB 26-63	Elements 0-25	Elements 26-79
0xA	88	MB 0-27	MB 28-63	Elements 0-27	Elements 28-87
0xB	96	MB 0-29	MB 30-63	Elements 0-29	Elements 30-95
0xC	104	MB 0-31	MB 32-63	Elements 0-31	Elements 32-103
0xD	112	MB 0-33	MB 34-63	Elements 0-31	Elements 32-111
0xE	120	MB 0-35	MB 36-63	Elements 0-31	Elements 32-119
0xF	128	MB 0-37	MB 38-63	Elements 0-31	Elements 32-127

1. The number of the last remaining available mailboxes is defined by the MCR[MAXMB] field.
2. If Rx Individual Mask Registers are not enabled then all Rx FIFO filters are affected by the Rx FIFO Global Mask.

Each group of eight filters occupies a memory space equivalent to two Message Buffers which means that the more filters are implemented the less Mailboxes will be available.

Considering that the Rx FIFO occupies the memory space originally reserved for MB0-5, RFFN should be programmed with a value corresponding to a number of filters not greater than the number of available memory words which can be calculated as follows:

$$(SETUP_MB - 6) \times 4$$

where SETUP_MB is MAXMB.

The number of remaining Mailboxes available will be:

$$SETUP_MB - 8 - (RFFN \times 2)$$

If the Number of Rx FIFO Filters programmed through RFFN exceeds the SETUP_MB value, the exceeding ones will not be functional.Unshaded regions in Table 44-23 indicate the valid combinations of MAXMB, RFEN and RFFN, shaded regions are not functional.

Table 44-23. Valid Combinations of MAXMB, RFEN and RFFN

RFFN	0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RFEN	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
MAXMB																	
0 - 6																	
7 - 8																	
9 - 10																	
11 - 12																	
13 -14																	
15 - 16																	
17 -18																	
19 - 20																	
21 - 22																	

Table continues on the next page...

Table 44-23. Valid Combinations of MAXMB, RFEN and RFFN (continued)

RFFN	0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RFEN	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
MAXMB																	
23 - 24																	
25 - 26																	
27 - 28																	
29 -30																	
31 - 32																	
33 - 34																	
35 - 36																	
37 - 63																	

Address: Base address + 34h offset

FLEXCANx_CTRL2 field descriptions

Field	Description
31 -	must be written as 0
30–29 -	This field is reserved. Reserved

Table continues on the next page...

FLEXCANx_CTRL2 field descriptions (continued)

Field	Description
28 WRMFRZ	Enable unrestricted write access to FlexCAN memory in Freeze mode. This bit can only be written in Freeze mode and has no effect out of Freeze mode. 1 Enable unrestricted write access to FlexCAN memory 0 Keep the write access restricted in some regions of FlexCAN memory
27–24 RFFN	This 4-bit field defines the number of Rx FIFO filters according to Table 44-22 . The maximum selectable number of filters is determined by the Arm. This field can only be written in Freeze mode as it is blocked by hardware in other modes. RFFN defines a number of Message Buffers occupied by Rx FIFO and ID Filter (see Table 44-22) that **may not exceed** the number of available Mailboxes present in module, defined by MCR[MAXMB].Default RFFN value is 0x0, which leads to a total of 8 Rx FIFO filters, occupies the first 8 Message Buffers (MB 0-7) and makes available the next Message Buffers (MB 8-63) for Mailboxes. As a second example, when RFFN is set to 0xD, there will be 112 Rx FIFO filters, located in MB 0-33, and MB 34-63 are available for Mailboxes. Notice that, in this case, individual masks (RXIMR) will just cover Rx FIFO filters in 0-31 range, and filters 32-111 will use RXFGMASK. In case of reducing the number of last Message Buffers, MCR[MAXMB] (see Module Configuration Register (FLEXCAN_MCR)) can be adjusted by the application to minimum of 33, in order to give room to the Rx FIFO and its ID Filter Table defined by RFFN. On the contrary, if the application sets MCR[MAXMB] to 16, for instance, the maximum RFFN is limited to 0x4. RFFN also impacts the definition of the minimum number of peripheral clocks per CAN bit as described in Table 44-19 (see Arbitration and Matching Timing).
23–19 TASD	This 5-bit field indicates how many CAN bits the Tx arbitration process start point can be delayed from the first bit of CRC field on CAN bus. This field can only be written in Freeze mode as it is blocked by hardware in other modes. This field is useful to optimize the transmit performance based on factors such as: peripheral/serial clock ratio, CAN bit timing and number of MBs . The duration of an arbitration process, in terms of CAN bits, is directly proportional to the number of available MBs and CAN baud rate and inversely proportional to the peripheral clock frequency. The optimal arbitration timing is that in which the last MB is scanned right before the first bit of the Intermission field of a CAN frame. Therefore, if there are few MBs and the system/serial clock ratio is high and the CAN baud rate is low then the arbitration can be delayed and vice-versa. If TASD is 0 then the arbitration start is not delayed, thus Arm has less time to configure a Tx MB for the next arbitration, but more time is reserved for arbitration . In the other hand, if TASD is 24 then Arm can configure a Tx MB later and less time is reserved for arbitration. If too little time is reserved for arbitration the FlexCAN may be not able to find winner MBs in time to compete with other nodes for the CAN bus. If the arbitration ends too much time before the first bit of Intermission field then there is a chance that Arm reconfigure some Tx MBs and the winner MB is not the best to be transmitted. The reset value is different on various platforms, according to their peripheral clock frequency, number of MBs and target CAN baud rate. The optimal configuration for TASD can be calculated as:

Table continues on the next page...

FLEXCANx_CTRL2 field descriptions (continued)

Field	Description
	$$TASD = 25 - \left\{ \frac{f_{CANCLK} \times [MAXMB + 3 - (RFEN \times 8) - (RFEN \times RFFN \times 2)] \times 2}{f_{sys} \times [1 + (PSEG1 + 1) + (PSEG2 + 1) + (PROPSEG + 1)] \times (PRESDIV + 1)} \right\}$$ where: f_{CANCLK} is the Protocol Engine (PE) Clock in Hz; PE clock is derrived from CAN_CLK_ROOT in CCM. See the Clock controller module. f_{SYS} is the peripheral clock in Hz; MAXMB is the value in CTRL1[MAXMB] field; RFEN is the value in CTRL1[RFEN] bit; RFFN is the value in CTRL2[RFFN] field; PSEG1 is the value in CTRL1[PSEG1] field; PSEG2 is the value in CTRL1[PSEG2] field; PROPSEG is the value in CTRL1[PROPSEG] field; PRESDIV is the value in CTRL1[PRESDIV] field. Please refer to Arbitration process and Protocol Timing for more details.
18 MRP	If this bit is set the matching process starts from the Mailboxes and if no match occurs the matching continues on the Rx FIFO. This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 1 Matching starts from Mailboxes and continues on Rx FIFO 0 Matching starts from Rx FIFO and continues on Mailboxes
17 RRS	If this bit is asserted Remote Request Frame is submitted to a matching process and stored in the corresponding Message Buffer in the same fashion of a Data Frame. No automatic Remote Response Frame will be generated. If this bit is negated the Remote Request Frame is submitted to a matching process and an automatic Remote Response Frame is generated if a Message Buffer with CODE=0b1010 is found with the same ID. This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 1 Remote Request Frame is stored 0 Remote Response Frame is generated
16 EACEN	This bit controls the comparison of IDE and RTR bits within Rx Mailboxes filters with their corresponding bits in the incoming frame by the matching process. This bit does not affect matching for Rx FIFO. This bit can only be written in Freeze mode as it is blocked by hardware in other modes. 1 Enables the comparison of both Rx Mailbox filter's IDE and RTR bit with their corresponding bits within the incoming frame. Mask bits do apply. 0 Rx Mailbox filter's IDE bit is always compared and RTR is never compared despite mask bits.
-	This field is reserved.

Table continues on the next page...

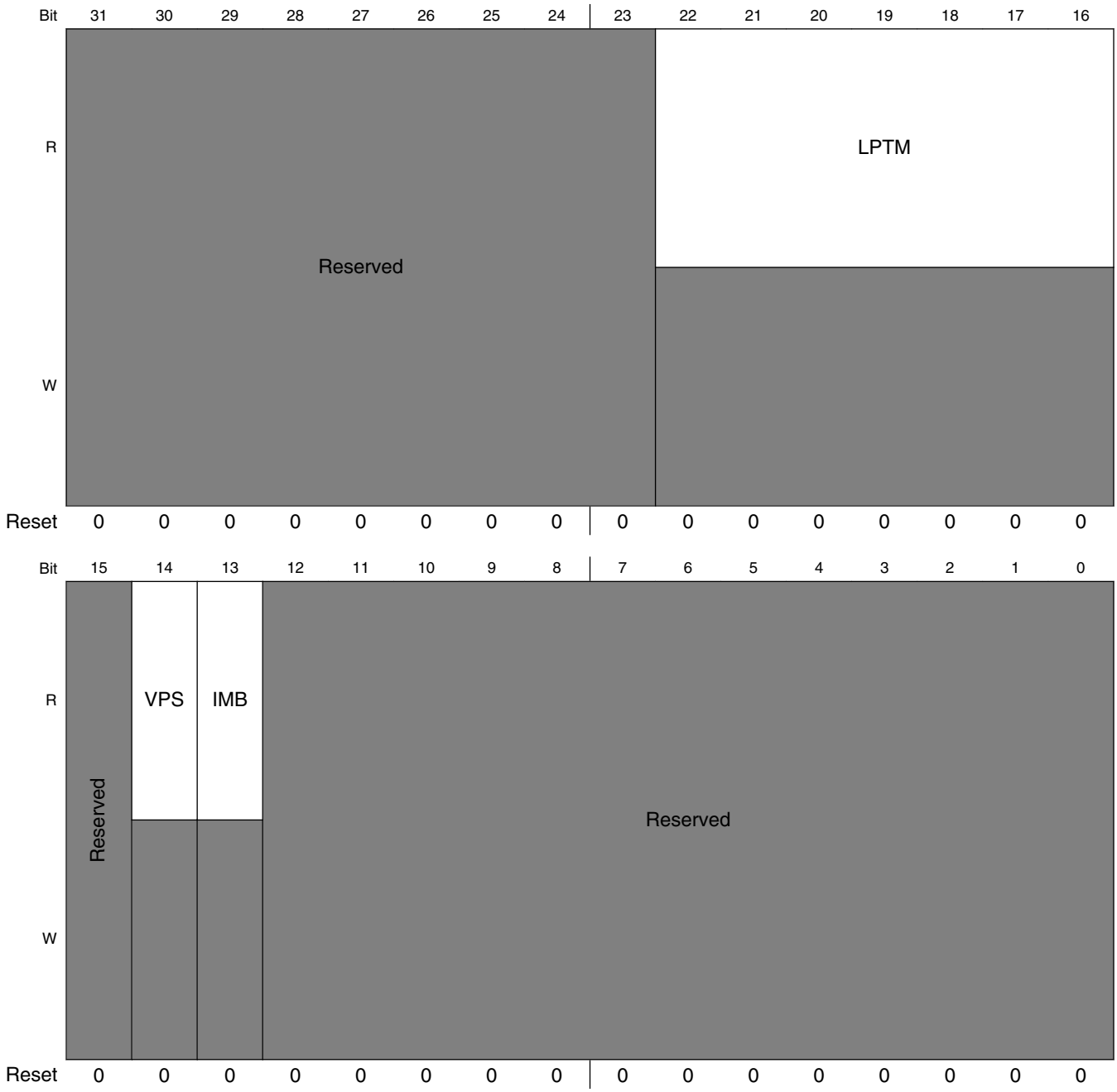
FLEXCANx_CTRL2 field descriptions (continued)

Field	Description
	Reserved

44.9.14 Error and Status 2 Register (FLEXCANx_ESR2)

This register reflects various interrupt flags and some general status.

Address: Base address + 38h offset



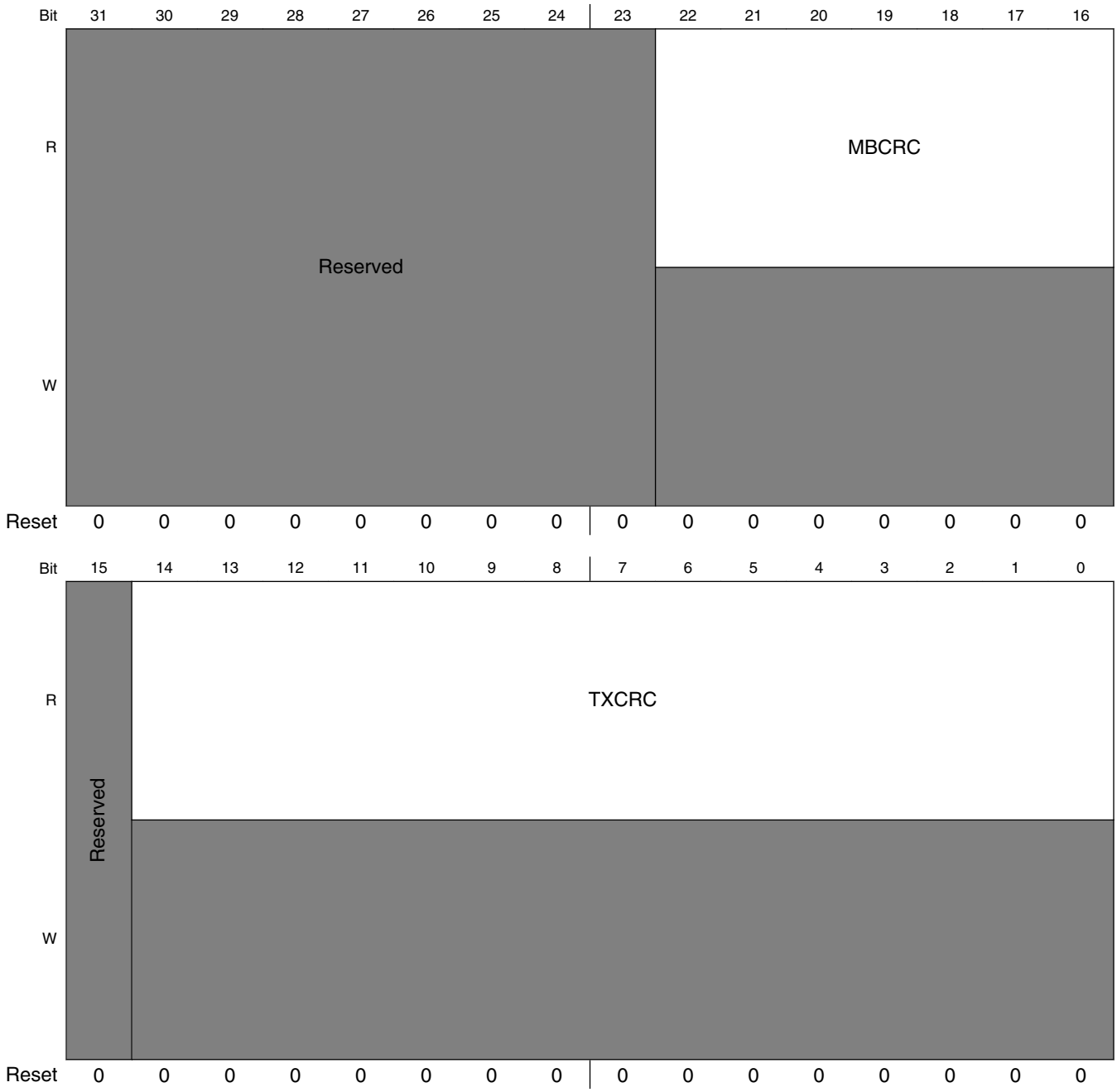
FLEXCANx_ESR2 field descriptions

Field	Description
31–23 -	This field is reserved. Reserved
22–16 LPTM	If ESR2[VPS] is asserted, his 7-bit field indicates the lowest number inactive Mailbox (refer to IMB bit description). If there is no inactive Mailbox then the Mailbox indicated depends on CTRL1[LBUF] bit value . If CTRL1[LBUF] bit is negated then the Mailbox indicated is the one which has the greatest arbitration value (see Highest Mailbox priority first). If CTRL1[LBUF] bit is asserted then the Mailbox indicated is the highest number active Tx Mailbox. If a Tx Mailbox is being transmitted it is not considered in LPTM calculation. If ESR2[IMB] is not asserted and a frame is transmitted successfully, LPTM is updated with its Mailbox number.
15 -	This field is reserved. Reserved
14 VPS	This bit indicates whether IMB and LPTM contents are currently valid or not. VPS is asserted upon every complete Tx arbitration process unless the Arm writes to Control and Status word of a Mailbox that has already been scanned (i.e. it is behind Tx Arbitration Pointer) during the Tx arbitration process . If there is no inactive Mailbox and only one Tx Mailbox which is being transmitted then VPS is not asserted. VPS is negated upon the start of every Tx arbitration process or upon a write to Control and Status word of any Mailbox.ESR2[VPS] is not affected by any Arm write into Control Status (C/S) of a MB which is blocked by abort mechanism. When MCR[AEN] is asserted, the abort code write in C/S of a MB that is been transmitted (pending abort), or any write attempt into a Tx MB with IFLAG set is blocked. 1 Contents of IMB and LPTM are valid 0 Contents of IMB and LPTM are invalid
13 IMB	If ESR2[VPS] is asserted, this bit indicates whether there is any inactive Mailbox (CODE field is either 0b1000 or 0b0000). This bit is asserted in the following cases: (1) During arbitration, if a LPTM is found and it is inactive. (2) If IMB is not asserted and a frame is transmitted successfully. (3) This bit is cleared in all start of arbitration (see Arbitration process). LPTM mechanism have the following behavior: if a MB is successfully transmitted and ESR2[IMB]=0 (no inactive Mailbox), then ESR2[VPS] and ESR2[IMB] are asserted and the index related to the MB just transmitted is loaded into ESR2[LPTM]. 1 If ESR2[VPS] is asserted, there is at least one inactive Mailbox. LPTM content is the number of the first one. 0 If ESR2[VPS] is asserted, the ESR2[LPTM] is not an inactive Mailbox.
-	This field is reserved. Reserved

44.9.15 CRC Register (FLEXCANx_CRCR)

This register provides information about the CRC of transmitted messages

Address: Base address + 44h offset



FLEXCANx_CRCR field descriptions

Field	Description
31–23 -	This field is reserved. Reserved
22–16 MBCRC	This field indicates the number of the Mailbox corresponding to the value in TXCRC field.
15 -	This field is reserved. Reserved
TXCRC	This field indicates the CRC value of the last message transmitted. This field is updated at the same time the Tx Interrupt Flag is asserted.

44.9.16 Rx FIFO Global Mask Register (FLEXCANx_RXFGMASK)

If Rx FIFO is enabled RXFGMASK is used to mask the Rx FIFO ID Filter Table elements that do not have a corresponding RXIMR according to CTRL2[RFFN] field setting.

This register can only be written in Freeze Mode as it is blocked by hardware in other modes.

Table 44-24. Rx FIFO Global Mask usage

Rx FIFO ID Filter Table Elements Format (MCR[IDAM])	Identifier Acceptance Filter fields					
	RTR	IDE	RXIDA	RXIDB	RXIDC	reserved
A	FGM[31]	FGM[30]	FGM[29:1]	-	-	FGM[0]
B	FGM[31] FGM[15]	FGM[30] FGM[14]	-	FGM[29:16] FGM[13:0] 1		-
C	-	-		-	FGM[31:24] FGM[23:16] FGM[15:8] FGM[7:0] 2	

1. If MCR[IDAM] field is equivalent to the format B only the fourteen most significant bits of the Identifier of the incoming frame are compared with the Rx FIFO filter.
2. If MCR[IDAM] field is equivalent to the format C only the eight most significant bits of the Identifier of the incoming frame are compared with the Rx FIFO filter.

Address: Base address + 48h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	FGM31_FGM0																															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

FLEXCANx_RXFGMASK field descriptions

Field	Description
FGM31_FGM0	These bits mask the ID Filter Table elements bits in a perfect alignment.Rx FIFO Global Mask Register (FLEXCAN_RXFGMASK) shows in detail which FGM bits mask each IDAF field. Clear this register has the effect of disabling the ID Filter. 1 The corresponding bit in the filter is checked 0 The corresponding bit in the filter is "don't care"

44.9.17 Rx FIFO Information Register (FLEXCANx_RXFIR)

RXFIR provides information on Rx FIFO.

This register is the port through which Arm accesses the output of the RXFIR FIFO located in RAM. The RXFIR FIFO is written by the FlexCAN whenever a new message is moved into the Rx FIFO as well as its output is updated whenever the output of the Rx FIFO is updated with the next message. Refer to Rx FIFO to find instructions on reading this register.

Address: Base address + 4Ch offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved																							IDHIT								
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FLEXCANx_RXFIR field descriptions

Field	Description
31–9 -	This field is reserved. Reserved
IDHIT	This 9-bit field indicates which Identifier Acceptance Filter (see Rx FIFO Structure) was hit by the received message that is in the output of the Rx FIFO . (refer to Rx FIFO for details) If multiple filters match the incoming message ID then the first matching IDAF found (lowest number) by the matching process is indicated. This field is valid only while the IFLAG[BUF5I] is asserted.

44.9.18 Debug 1 register (FLEXCANx_DBG1)

This register provides useful information only for IP and device debug purpose.

This table provides detailed information for the DBG1[CFSM] field.

Table 44-25. CAN finite state machine (CFSM) codification

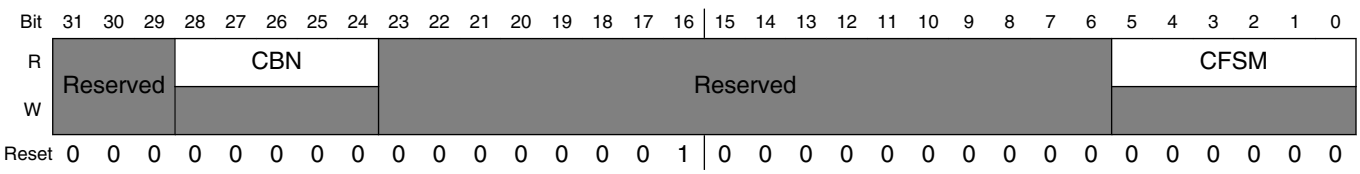
DEC	HEX	State
0	0x0	RESETMODULE
1	0x1	WAITFORBUSIDLE
2	0x2	BUSIDLE
3	0x3	TXSTARTOFFRAME
4	0x4	TXIDENTIFIER
5	0x5	TXSRTRBIT
6	0x6	TXIDEBIT
7	0x7	TXIDEBITR
8	0x8	TXIDEBITE
9	0x9	TXEXIDENTIFIER
10	0xa	TXRTRBIT
11	0xb	TXRESERVEDBITS
12	0xc	TXRESERVEDBITSEX
13	0xd	TXDATALENGTHCODE
14	0xe	TXDATAFIELD
15	0xf	TXCRCSEQUENCE
16	0x10	TXCRCDELIMITER
17	0x11	TXACKSLOT
18	0x12	TXACKDELIMITER
19	0x13	TXENDOFFRAME
20	0x14	TXINTERMISSION
21	0x15	TXSUSPENDTRANSMISSION
22	0x16	TXACTIVEERRORFLAG
23	0x17	TXACTIVEERRORFLAGW1
24	0x18	TXERRORDELIMITER
25	0x19	TXPASSIVEERRORFLAG
26	0x1a	TXPASSIVEERRORFLAGW1
27	0x1b	TXOVERLOADFLAG
28	0x1c	TXOVERLOADFLAGW1
29	0x1d	TXOVERLOADDELIMITER
30	0x1e	LOSTARBITRATION
31	0x1f	LOSTARBINSRTRBIT
32	0x20	LOSTARBINIDEBIT
33	0x21	LOSTARBINIDEBITSTUFFR
34	0x22	LOSTARBINIDEBITSTUFFE
35	0x23	LOSTEARBITRATION
36	0x24	LOSTARBINRTRBIT
37	0x25	RXRESERVEDBITS

Table continues on the next page...

Table 44-25. CAN finite state machine (CFSM) codification (continued)

DEC	HEX	State
38	0x26	RXRESERVEDBITSEX
39	0x27	RXDATALENGTHCODE
40	0x28	RXDATAFIELD
41	0x29	RXCRCSEQUENCEFIRST
42	0x2a	RXCRCSEQUENCE
43	0x2b	RXCRCDELIMITER
44	0x2c	RXACKSLOT
45	0x2d	RXACKDELIMITER
46	0x2e	RXENDOFFRAME
47	0x2f	RXINTERMISSION
48	0x30	RXACTIVEERRORFLAG
49	0x31	RXACTIVEERRORFLAGW1
50	0x32	RXACTIVEERRORFLAGW2
51	0x33	RXERRORDELIMITER
52	0x34	RXPASSIVEERRORFLAG
53	0x35	RXPASSIVEERRORFLAGW1
54	0x36	RXPASSIVEERRORFLAGW2
55	0x37	RXOVERLOADFLAG
56	0x38	RXOVERLOADFLAGW1
57	0x39	RXOVERLOADDELIMITER
58	0x3a	BUSOFF1

Address: Base address + 58h offset



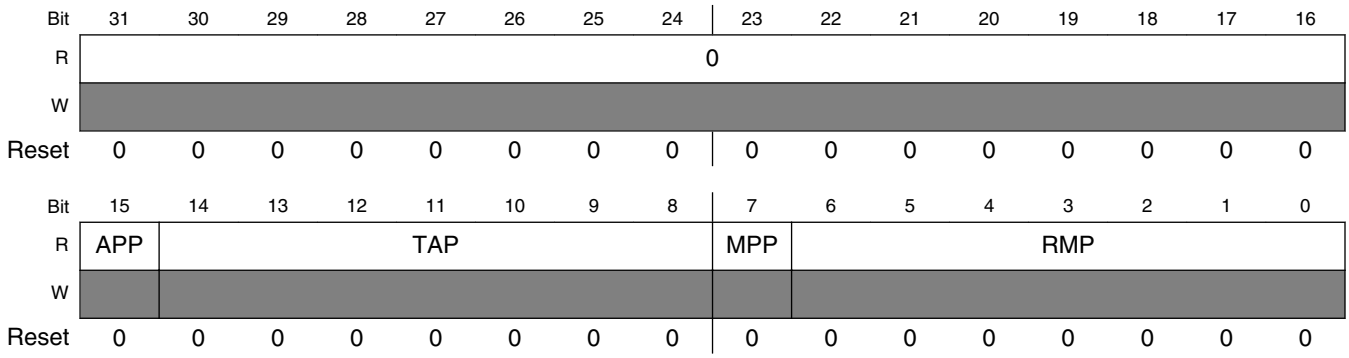
FLEXCANx_DBG1 field descriptions

Field	Description
31–29 -	This field is reserved. Reserved
28–24 CBN	CAN Bit Number This bitfield contains the current CAN bit number.
23–6 -	This field is reserved. Reserved
CFSM	CAN Finite State Machine This bitfield contains the current state of the CAN FSM. See the table above for detailed information.

44.9.19 Debug 2 register (FLEXCANx_DBG2)

This register provides useful information only for IP and device debug purpose.

Address: Base address + 5Ch offset



FLEXCANx_DBG2 field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
15 APP	Arbitration Process in Progress Indicates whether the arbitration process is in progress or not. It is set when the first message buffer is accessed and kept asserted as long as the arbitration is ongoing. APP is negated when the highest priority MB is found or when the last MB is accessed or if the arbitration process is interrupted. 0 No matching process ongoing. 1 Matching process is in progress.
14–8 TAP	Tx Arbitration Pointer This 7-bit field indicates the number of the MB that is currently being accessed by the arbitration process.
7 MPP	Matching Process in Progress Indicates whether the matching process is in progress or not. It is set when the first matching element is accessed and kept asserted when the matching is ongoing. MPP is negated when a match occurs or when the last matching element is accessed or if the matching process is interrupted. 0 No matching process ongoing. 1 Matching process is in progress.
RMP	Rx Matching Pointer This 7-bit field indicates the number of the MB that is currently being accessed by the matching process.

44.9.20 Rx Individual Mask Registers (FLEXCANx_RXIMR*n*)

RXIMR are used as acceptance masks for ID filtering in Rx MBs and the Rx FIFO . If the Rx FIFO is not enabled, one mask register is provided for each available Mailbox, providing ID masking capability on a per Mailbox basis.

When the Rx FIFO is enabled (MCR[RFEN] bit is asserted), up to 32 Rx Individual Mask Registers can apply to the Rx FIFO ID Filter Table elements on a one-to-one correspondence depending on CTRL2[RFFN] setting. Refer to Control 2 Register (FLEXCAN_CTRL2) for details .

RXIMR can only be written by the Arm while the module is in Freeze Mode, otherwise they are blocked by hardware .

The Individual Rx Mask Registers are not affected by reset and must be explicitly initialized prior to any reception.

Address: Base address + 880h offset + (4d × i), where i=0d to 63d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MI31_MI0																															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

FLEXCANx_RXIMR*n* field descriptions

Field	Description
MI31_MI0	These bits mask both Mailbox filter and Rx FIFO ID Filter Table element in distinct ways. For Mailbox filter refer to Rx Mailboxes Global Mask Register (FLEXCAN_RXMGMASK). For Rx FIFO ID Filter Table element refer to Rx FIFO Global Mask Register (FLEXCAN_RXFGMASK). 1 The corresponding bit in the filter is checked 0 the corresponding bit in the filter is "don't care"

44.9.21 Glitch Filter Width Registers (FLEXCANx_GFWR)

The Glitch Filter just takes effects when FLEXCAN enters the STOP mode.

Address: Base address + 9E0h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																								GFWR							
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1

FLEXCANx_GFWR field descriptions

Field	Description
31–8 Reserved	This read-only field is reserved and always has the value 0.
GFWR	It determines the Glitch Filter Width. The width will be divided from Oscillator clock by GFWR values. By default, it is 5.33 µs when the oscillator is 24 MHz. Filter Pulse Width = [(GFWR FIELD + 1) x (1 / Osc. Frequency)]

Chapter 45
Flexible Data-rate Controller Area Network (CANFD/ FlexCAN3)

45.1 Chip-specific FLEXCAN information

Table 45-1. Reference links to related information

Topic	Related module(s)	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
Interrupts, DMA Events and XBAR Assignments	-	Interrupts, DMA Events and XBAR Assignments

45.2 Overview

The FlexCAN module is a communication controller implementing the CAN protocol according to the ISO 11898-1 standard and CAN 2.0 B protocol specifications.

The CAN protocol was primarily designed to be used as a vehicle serial data bus, meeting the specific real-time processing and reliable operation requirements in the EMI environment of a vehicle. The FlexCAN module is a full implementation of the CAN protocol specification, the CAN with Flexible Data rate (CAN FD) protocol, and the CAN 2.0 version B protocol, which supports both standard and extended message frames and long payloads.

NOTE

Rx FIFO cannot be used in FD mode.

45.2.1 Block diagram

A general block diagram is shown in Figure 45-1, which describes the main subblocks implemented in the FlexCAN module.

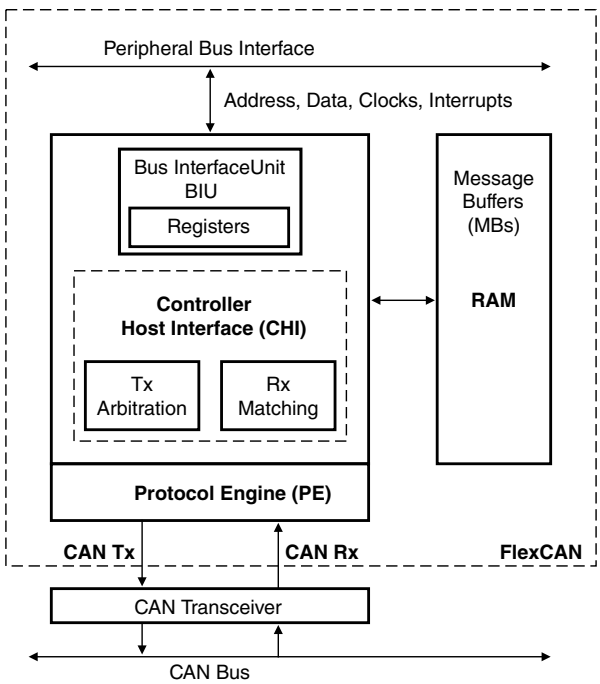


Figure 45-1. FlexCAN block diagram

The Protocol Engine (PE) submodule manages the serial communication on the CAN bus:

- Requesting RAM access for receiving and transmitting message frames
- Validating received messages
- Performing error handling
- Detecting CAN FD messages

The Controller Host Interface (CHI) submodule manages message buffer selection for reception and transmission, taking care of arbitration and ID matching algorithms for both CAN FD and non-CAN FD message formats.

The Bus Interface Unit (BIU) submodule controls access to and from the internal interface bus, in order to establish connection to the CPU and to other blocks. Clocks, address and data buses, interrupt outputs, DMA and test signals are accessed through the BIU.

45.2.2 Features

The FlexCAN module includes these distinctive features:

- Full implementation of the CAN with Flexible Data Rate (CAN FD) protocol specification and CAN protocol specification, Version 2.0 B
  - Standard data frames
  - Extended data frames
  - Zero to sixty four bytes data length
  - Programmable bit rate (see the chip-specific FlexCAN information for the specific maximum rate configuration)
  - Content-related addressing
- Compliant with the ISO 11898-1 standard
- Flexible mailboxes configurable to store 0 to 8, 16, 32, or 64 bytes data length
- Each mailbox configurable as receive or transmit, all supporting standard and extended messages
- Individual Rx Mask registers per mailbox
- Full-featured Rx FIFO with storage capacity for up to six frames and automatic internal pointer handling with DMA support
- Transmission abort capability
- Flexible message buffers (MBs), totaling 64 message buffers of 8 bytes data length each, configurable as Rx or Tx
- RAM not used by reception or transmission structures can be used as general purpose RAM space
- Listen-Only mode capability
- Programmable Loop-Back mode supporting self-test operation
- Programmable transmission priority scheme: lowest ID, lowest buffer number, or highest priority
- Time stamp based on 16-bit free running timer, with an optional external time tick
- Global network time, synchronized by a specific message
- Maskable interrupts
- Independence from the transmission medium (an external transceiver is assumed)
- Short latency time due to an arbitration scheme for high-priority messages
- Low power modes, with programmable wakeup on bus activity
- Transceiver delay compensation feature when transmitting CAN FD messages at faster data rates
- Remote request frames may be managed automatically or by software
- CAN bit time settings and configuration bits can only be written in Freeze mode
- Tx mailbox status (Lowest priority buffer or empty buffer)
- Identifier Acceptance Filter Hit Indicator (IDHIT) register for received frames
- SYNCH bit available in Error in Status 1 register to indicate that the module is synchronous with CAN bus

- CRC status for transmitted message
- Rx FIFO Global Mask register
- Selectable priority between mailboxes and Rx FIFO during matching process
- Powerful Rx FIFO ID filtering, capable of matching incoming IDs against either 128 extended, 256 standard, or 512 partial (8 bit) IDs, with up to 32 individual masking capability
- 100% backward compatibility with previous FlexCAN version

45.3 Functional description

The FlexCAN module is a CAN protocol engine with a very flexible mailbox system for transmitting and receiving CAN frames. The mailbox system is composed of a set of message buffers (MB) that store configuration and control data, time stamp, message ID, and data (see Message buffer structure). The memory corresponding to the first 38 MBs can be configured to support a FIFO reception scheme with a powerful ID filtering mechanism, capable of checking incoming frames against a table of IDs (up to 128 extended IDs or 256 standard IDs or 512 8-bit ID slices), with individual mask register for up to 32 ID filter table elements.

For Classical CAN frames, simultaneous reception through FIFO and mailbox is supported. For CAN FD frames, reception is supported through mailboxes only. For mailbox reception, a matching algorithm makes it possible to store received frames only into MBs that have the same ID programmed in the ID field. A masking scheme makes it possible to match the ID programmed on the MB with a range of IDs on received CAN frames. For transmission, an arbitration algorithm decides the prioritization of MBs to be transmitted based on the message ID (optionally augmented by 3 local priority bits) or the MB ordering.

Before proceeding with the functional description, an important concept must be explained. A message buffer is said to be active at a given time if it can participate in both the matching and arbitration processes. An Rx MB with a 0b code is inactive (see Table 45-27). Similarly, a Tx MB with a 1000b or 1001b code is also inactive (see Table 45-28).

The FlexCAN module is also able to receive and transmit messages in CAN FD format. The message buffers are sized to adequately store the quantity of data bytes selected by the FDCTRL[MBDSRn] fields. The quantity of FD MBs available for a given quantity of data bytes is described in the FDCTRL register. See also FlexCAN memory partition for CAN FD.

45.3.1 Modes of operation

The FlexCAN module has these functional modes:

- Normal mode (User or Supervisor):

  In Normal mode, the module operates receiving and/or transmitting message frames, errors are managed normally, and all CAN Protocol functions are enabled. User and Supervisor modes differ in the access to some restricted control registers.

- Freeze mode:

  Freeze mode is enabled when MCR[FRZ] is asserted. If enabled, Freeze mode is entered when MCR[HALT] is set or when is requested at chip level and MCR[FRZ_ACK] is asserted by the FlexCAN. In this mode, no transmission or reception of frames is done and synchronicity to the CAN bus is lost. See Freeze mode for more information.

- Loop-Back mode:

  The module enters this mode when CTRL1[LPB] is asserted. In this mode, FlexCAN performs an internal loop back that can be used for self-test operation. The bit stream output of the transmitter is internally fed back to the receiver input. The Rx CAN input pin is ignored and the Tx CAN output goes to the recessive state (logic '1'). FlexCAN behaves as it normally does when transmitting and treats its own transmitted message as a message received from a remote node. In this mode, FlexCAN ignores the bit sent during the ACK slot in the CAN frame acknowledge field to ensure proper reception of its own message. Both transmit and receive interrupts are generated.

- Listen-Only mode:

  The module enters this mode when CTRL1[LOM] is asserted. In this mode, transmission is disabled, all error counters are frozen, and the module operates in a CAN Error Passive mode. Only messages acknowledged by another CAN station will be received. If FlexCAN detects a message that has not been acknowledged, it will flag a BIT0 error (without changing the REC), as if it was trying to acknowledge the message.

- CAN FD Active mode:

  In this mode, FlexCAN is capable of transmitting and receiving all messages formatted according to the CAN FD Standard (2.0) and 2.0B Protocol in an interleaved fashion. The CPU can set the FlexCAN into CAN FD Active mode by configuring MCR[FDEN] in Freeze mode.

It is important to know which features are available in CAN FD active mode. This table lists the differences between FD and classical modes.

Table 45-2. Differences between CAN classical and CAN FD

Feature	CAN classical	CAN FD
Rx FIFO DMA	Yes	No
Rx FIFO	Yes	No

For low-power operation, the FlexCAN module has:

- Module Disable mode:

  This low-power mode is entered when MCR[MDIS] is asserted by the CPU and MCR[LPM_ACK] is asserted by FlexCAN. When disabled, the module issues a request to disable the clocks to the CAN Protocol Engine and Controller Host Interface submodules. Exit from this mode is done by negating MCR[MDIS]. See Module Disable mode for more information.

- Doze mode:

  This low power mode is entered when MCR[DOZE] is asserted and Doze mode is requested at chip level, and MCR[LPM_ACK] is asserted by FlexCAN. When in Doze mode, the module issues a request to disable the clocks to the CAN Protocol Engine and the CAN Controller-Host Interface submodules. Exit from this mode happens when MCR[DOZE] is negated, when the chip is removed from Doze mode, or when activity is detected on the CAN bus and the Self Wake Up mechanism is enabled. See Doze mode for more information.

- Stop mode:

  This low power mode is entered when Stop mode is requested at chip level and MCR[LPM_ACK] is asserted by the FlexCAN. When in Stop Mode, the module puts itself in an inactive state and then informs the CPU that the clocks can be shut down globally. Exit from this mode happens when the Stop mode request is removed, or when activity is detected on the CAN bus and the Self Wake Up mechanism is enabled. See Stop mode for more information.

45.3.1.1 Modes of operation details

The FlexCAN module has functional modes and low-power modes. See Modes of operation for an introductory description of all the modes of operation. The following sub-sections contain functional details on Freeze mode and the low-power modes.

CAUTION

"Permanent Dominant" failure on CAN Bus line is not supported by FlexCAN. If a Low-Power request or Freeze mode request is done during a "Permanent Dominant", the corresponding acknowledge can never be asserted.

45.3.1.1.1 Freeze mode

This mode is requested either by the CPU through the assertion of MCR[HALT] or when the chip is put into Debug mode. In both cases it is also necessary that MCR[FRZ] be asserted and the module not be in a low-power mode.

The acknowledgement is obtained through the assertion by the FlexCAN of MECR[FRZ_ACK]. The CPU must only consider the FlexCAN in Freeze mode when both request and acknowledgement conditions are satisfied.

When Freeze mode is requested, FlexCAN does the following:

- Waits to be in either Intermission, Passive Error, Bus Off, or Idle state.
- Waits for all internal activities like arbitration, matching, move-in, and move-out to finish. A pending move-in does not prevent entering Freeze mode.
- Ignores the Rx input pin and drives the Tx pin as recessive.
- Stops the prescaler, thus halting all CAN protocol activities.
- Grants write access to the Error Counters register, which is read-only in other modes.
- Sets MCR[NOTRDY] and MCR[FRZACK].

After requesting Freeze mode, the user must wait for MCR[FRZ_ACK] to be asserted before executing any other action; otherwise FlexCAN may operate in an unpredictable way. In Freeze mode, all memory-mapped registers are accessible.

Exiting Freeze mode is done in one of the following ways:

- CPU negates MCR[FRZ].
- The chip is removed from Debug mode and/or the HALT bit is negated.

MCR[FRZ_ACK] is negated after the protocol engine recognizes the negation of the freeze request. When out of Freeze mode, FlexCAN tries to resynchronize to the CAN bus by waiting for 11 consecutive recessive bits.

45.3.1.1.2 Module Disable mode

This low-power mode is normally used to temporarily disable a complete FlexCAN block, with no power consumption. It is requested by the CPU through the assertion of MCR[MDIS], and the acknowledgement is obtained through the assertion by the FlexCAN of MCR[LPMACK]. The CPU must only consider the FlexCAN in Disable mode when both request and acknowledgement conditions are satisfied.

If the module is disabled during Freeze mode, it requests to disable the clocks to the PE and CHI submodules, sets MCR[LPMACK] and negates MCR[FRZACK].

If the module is disabled during transmission or reception, FlexCAN does the following:

- Waits to be in either Idle or Bus Off state, or else waits for the third bit of Intermission and then checks it to be recessive.
- Waits for all internal activities like arbitration, matching, move-in, and move-out to finish. A pending move-in is not taken into account.
- Ignores its Rx input pin and drives its Tx pin as recessive.
- Shuts down the clocks to the PE and CHI submodules
- Sets MCR[NOTRDY] and MCR[LPMACK].

The Bus Interface Unit continues to operate, enabling the CPU to access memory-mapped registers, except the Rx Mailboxes Global Mask registers, the Rx Buffer 14 Mask register, the Rx Buffer 15 Mask register, the Rx FIFO Global Mask register. The Rx FIFO Information register, the message buffers, the Rx Individual Mask registers, and the reserved words within RAM may not be accessed when the module is in Disable mode. Exiting from this mode is done by negating the MDIS bit by the CPU, which causes the FlexCAN to request to resume the clocks and negate the LPMACK bit after the CAN protocol engine recognizes the negation of disable mode requested by the CPU.

45.3.1.1.3 Doze mode

This is a system low power mode in which the CPU bus is kept alive and a global Doze mode request is sent to all peripherals asking them to enter low-power mode. When Doze mode is globally requested, MCR[DOZE] needs to have been asserted previously for

Doze mode to be triggered. The acknowledgement is obtained through the assertion by the FlexCAN of MCR[LPMACK]. The CPU must only consider the FlexCAN to be in Doze mode when both request and acknowledgement conditions are satisfied.

If Doze mode is triggered during Freeze mode, FlexCAN requests to shut down the clocks to the PE and CHI submodules, sets MCR[LPMACK] and negates MCR[FRZACK]. If Doze mode is triggered during transmission or reception, FlexCAN does the following:

- Waits to be in either Idle or Bus Off state, or else waits for the third bit of Intermission and checks it to be recessive.
- Waits for all internal activities like arbitration, matching, move-in, and move-out to finish. A pending move-in is not taken into account.
- Ignores its Rx input pin and drives its Tx pin as recessive.
- Shuts down the clocks to the PE and CHI submodules.
- Sets MCR[NOTRDY] and MCR[LPMACK].

The Bus Interface Unit continues to operate, enabling the CPU to access memory-mapped registers, except the Rx Mailboxes Global Mask registers, the Rx Buffer 14 Mask register, the Rx Buffer 15 Mask register, the Rx FIFO Global Mask register. The Rx FIFO Information register, the message buffers, the Rx Individual Mask registers, and the reserved words within RAM may not be accessed when the module is in Doze mode.

Exiting Doze mode is done in one of the following ways:

- CPU removing the Doze mode request
- CPU negating MCR [DOZE]
- Self Wake mechanism

In the Self Wake mechanism, if MCR[SLFWAK] was set at the time FlexCAN entered Doze mode, then upon detection of a recessive to dominant transition on the CAN bus, FlexCAN negates the DOZE bit, requests to resume its clocks and negates the LPMACK after the CAN protocol engine recognizes the negation of the Doze mode request. It also sets ESR [WAKINT] and, if enabled by MCR[WAKMSK], generates a Wake-Up interrupt to the CPU. FlexCAN will then wait for 11 consecutive recessive bits to synchronize to the CAN bus. As a consequence, it will not receive the frame that woke it up. The following table details the effect of SLFWAK and WAKMSK upon wakeup from Doze mode.

Table 45-3. Wakeup from Doze mode

SLFWAK	WAKINT	WAKMSK	FlexCAN clocks enabled	Wakeup interrupt generated
0	—	—	No	No
0	—	—	No	No
1	0	0	No	No
1	0	1	No	No
1	1	0	Yes	No
1	1	1	Yes	Yes

The sensitivity to CAN bus activity can be modified by applying a low-pass filter function to the Rx CAN input line when in Doze mode. (See MCR[WAKSRC].) This feature can be used to protect FlexCAN from waking up due to short glitches on the CAN bus lines. Such glitches can result from electromagnetic interference within noisy environments.

45.3.1.1.4 Stop mode

This is a system low-power mode in which all chip clocks can be stopped for maximum power savings. The Stop mode is globally requested by the CPU and acknowledgement is obtained through the assertion by the FlexCAN of a Stop Acknowledgement signal. The CPU must only consider the FlexCAN in Stop mode when both request and acknowledgement conditions are satisfied.

If FlexCAN receives the global Stop mode request during Freeze mode, it sets MCR[LPMACK], negates MCR[FRZACK], and then sends the Stop Acknowledge signal to the CPU, in order to shut down the clocks globally.

If Stop mode is requested during transmission or reception, FlexCAN does the following:

- Waits to be in either Idle or Bus Off state, or else waits for the third bit of Intermission and checks it to be recessive
- Waits for all internal activities like arbitration, matching, move-in, and move-out to finish. A pending move-in is not taken into account.
- Ignores its Rx input pin and drives its Tx pin as recessive
- Sets MCR[NOTRDY] and MCR[LPMACK]
- Sends a Stop Acknowledge signal to the CPU, so that it can shut down the clocks globally

Stop mode is exited when the CPU resumes the clocks and removes the Stop mode request. This can be as a result of the Self Wake mechanism.

In the Self Wake mechanism, if MCR[SLFWAK] was set at the time FlexCAN entered Stop mode, then upon detection of a recessive to dominant transition on the CAN bus, FlexCAN sets ESR[WAKINT] and, if enabled by MCR[WAKMSK], generates a Wake Up interrupt to the CPU. Upon receiving the interrupt, the CPU should resume the clocks and remove the Stop mode request. FlexCAN will then wait for 11 consecutive recessive bits to synchronize to the CAN bus. As a consequence, it will not receive the frame that woke it up. The following table details the effect of MCR[SLFWAK] and MCR[WAKMSK] upon wakeup from Stop mode. Note that wakeup from Stop mode only works when both bits are asserted.

After the CAN protocol engine recognizes the negation of the Stop mode request, the FlexCAN negates MCR[LPMACK]. FlexCAN will then wait for 11 consecutive recessive bits to synchronize to the CAN bus. As a consequence, it will not receive the frame that woke it up.

Table 45-4. Wakeup from Stop mode

SLFWAK	WAKINT	WAKMSK	Chip clocks enabled	Wakeup interrupt generated
0	—	—	No	No
0	—	—	No	No
1	0	0	No	No
1	0	1	No	No
1	1	0	No	No
1	1	1	Yes	Yes

The sensitivity to CAN bus activity can be modified by applying a low-pass filter function to the Rx CAN input line when in Stop mode. (See the description in MCR[WAKSRC].) This feature can be used to protect FlexCAN from waking up due to short glitches on the CAN bus lines. Such glitches can result from electromagnetic interference within noisy environments.

45.3.2 Transmit process

To transmit a CAN frame, the CPU must prepare a message buffer for transmission by executing the following procedure:

1. Check whether the respective interrupt bit is set and clear it.
2. If the MB is active (transmission pending), write the ABORT code (1001b) to the CODE field of the Control and Status word to request an abort of the transmission.
3. Wait for the corresponding IFLAG bit to be asserted by polling the IFLAG register, or by the interrupt request if enabled by the respective IMASK bit.

4. Read back the CODE field to check if the transmission was aborted or transmitted (see Transmission abort mechanism).
5. Clear the corresponding interrupt flag.
6. Write the ID register (containing the local priority if enabled via MCR[LPRIO_EN]).
7. Write payload data bytes.
8. Configure the Control and Status word with the desired configuration.
   a. Set ID type via MB_CS[IDE].
   b. Set Remote Transmission Request (if needed) via MB_CS[RTR].
   c. If MCR[FDEN] is enabled, also configure the MB_CS[EDL] and MB_CS[BRS] fields. For details about the relationship between the written value and transmitted value of the MB_CS[ESI] field, see Table 45-9.[1]
   d. Set Data Length Code in bytes via MB_CS[DLC]. See Table 45-29 for detailed information.
   e. Activate the message buffer to transmit the CAN frame by setting MB_CS[CODE] to Ch.

NOTE

It is strongly recommended that all the fields in MB_CS word be configured in only one 32-bit write operation to maximize software performance. If the fields are configured in separate writes, the MB_CS[CODE] must be the last write in the C/S word.

When the MB is activated, it participates in the arbitration process and is eventually transmitted according to its priority. When the DLC value stored in the MB selected for transmission is larger than the respective MB payload size, FlexCAN adds the necessary number of bytes with constant CCh pattern to complete the expected DLC.

At the end of the successful transmission:

- The value of the free running timer is written into the Time Stamp field.
- The CODE field in the Control and Status word is updated.
- Both CRC and FDCRC registers are updated.
- A status flag is set in the Interrupt Flag register.
- An interrupt is generated if allowed by the corresponding Interrupt Mask Register bit.

The new CODE field after transmission depends on the code that was used to activate the MB (see Table 45-27 and Table 45-28 in Message buffer structure).

1. The ESI field does not need to be written, and will automatically be transmitted dominant by error active nodes and recessive by error passive nodes. Note that there is an exception if the FlexCAN is operating as a network gateway: In that case, the CPU writes the MB_CS[ESI] bit according to the error status of the node which sent the message.

When the Abort feature is enabled (MCR[AEN] is asserted), after the Interrupt flag is asserted for a mailbox configured as transmit buffer, the mailbox is blocked. Therefore the CPU is not able to update it until the Interrupt Flag is negated by the CPU. This means that the CPU must clear the corresponding IFLAG bit before starting to prepare this MB for a new transmission or reception.

NOTE

If backwards compatibility is desired (MCR[AEN] is negated), write the INACTIVE code (1000b) to the CODE field to inactivate the MB. However, in this case the pending frame may be transmitted without notification (see Mailbox inactivation).

45.3.3 Arbitration process

The arbitration process scans the mailboxes, searching for the Tx mailbox that holds the message to be sent in the next opportunity. This mailbox is called the arbitration winner.

The scan starts from the lowest number mailbox and runs toward the higher ones.

The arbitration process is triggered in the following events:

- From the CRC field of the CAN frame. The start point depends on the value of CTRL2[TASD].
- During the Error Delimiter field of a CAN frame.
- During the Overload Delimiter field of a CAN frame.
- When the winner is inactivated and the CAN bus has still not reached the first bit of the Intermission field.
- When there is a CPU write to the C/S word of a winner MB and the CAN bus has still not reached the first bit of the Intermission field.
- When CHI is in Idle state and the CPU writes to the C/S word of any MB.
- When FlexCAN exits Bus Off state.
- Upon leaving Freeze mode or Low Power mode.

If the arbitration process does not manage to evaluate all mailboxes before the CAN bus has reached the first bit of the Intermission field, the temporary arbitration winner is invalidated and the FlexCAN will not compete for the CAN bus in the next opportunity.

The arbitration process selects the winner among the active Tx mailboxes at the end of the scan according to the settings of both CTRL1[LBUF] and MCR[LPRIOEN].

45.3.3.1 Lowest-number mailbox first

If CTRL1[LBUF] is asserted, the first (lowest number) active Tx mailbox found is the arbitration winner. MCR[LPRIOEN] has no effect when CTRL1[LBUF] is asserted.

45.3.3.2 Highest-priority mailbox first

If CTRL1[LBUF] is negated, then the arbitration process searches the active Tx mailbox with the highest priority, which means that this mailbox's frame would have a higher probability to win the arbitration on CAN bus when multiple external nodes compete for the bus at the same time.

The sequence of bits considered for this arbitration is called the *arbitration value* of the mailbox. The highest-priority Tx mailbox is the one that has the lowest arbitration value among all Tx mailboxes.

If two or more mailboxes have equivalent arbitration values, the mailbox with the lowest number is the arbitration winner.

The composition of the arbitration value depends on the MCR[LPRIOEN] setting.

45.3.3.2.1 Local priority disabled

If MCR[LPRIOEN] is negated the arbitration value is built in the exact sequence of bits as they would be transmitted in a CAN frame (see the following table) in such a way that the local priority is disabled.

Table 45-5. Composition of the arbitration value when local priority is disabled

Format	Mailbox arbitration value (32 bits)				
Standard (IDE = 0)	Standard ID (11 bits)	RTR (1 bit)	IDE (1 bit)	- (18 bits)	- (1 bit)
Extended (IDE = 1)	Extended ID[28:18] (11 bits)	SRR (1 bit)	IDE (1 bit)	Extended ID[17:0] (18 bits)	RTR (1 bit)

45.3.3.2.2 Local priority enabled

To enable local priority, MCR[LPRIOEN] must be asserted. In this case the mailbox PRIO field is included at the very left of the arbitration value (see the following table).

Table 45-6. Composition of the arbitration value when local priority is enabled

Format	Mailbox arbitration value (35 bits)					
Standard (IDE = 0)	PRIO (3 bits)	Standard ID (11 bits)	RTR (1 bit)	IDE (1 bit)	- (18 bits)	- (1 bit)
Extended (IDE = 1)	PRIO (3 bits)	Extended ID[28:18] (11 bits)	SRR (1 bit)	IDE (1 bit)	Extended ID[17:0] (18 bits)	RTR (1 bit)

Because the PRIO field is the most significant part of the arbitration value, mailboxes with low PRIO values have higher priority than mailboxes with high PRIO values regardless of the rest of their arbitration values.

Note that the PRIO field is not part of the frame on the CAN bus. Its purpose is only to affect the internal arbitration process.

45.3.3.3 Arbitration process (continued)

After the arbitration winner is found, its content is copied to a hidden auxiliary MB called Tx Serial Message Buffer (Tx SMB), which has the same structure as a normal MB but is not user accessible. This operation is called move-out and after it is done, write access to the C/S word of the corresponding MB is blocked (if MCR[AEN] is asserted). Write access is restored in the following events:

- After the MB is transmitted and the corresponding IFLAG bit is cleared by the CPU.
- FlexCAN enters Freeze mode or Bus Off.
- FlexCAN loses the bus arbitration or there is an error during the transmission.

At the first opportunity window on the CAN bus, the message on the Tx SMB is transmitted according to the CAN protocol rules.

Arbitration process can be triggered in the following situations:

- During Rx and Tx frames from CAN CRC field to end of frame. CTRL2[TASD] value may be changed to optimize the arbitration start point.
- During CAN Bus Off state from TX_ERR_CNT=124 to 128. CTRL2[TASD] value may be changed to optimize the arbitration start point.
- During C/S write by CPU in Bus Idle. First C/S write starts arbitration process, and a second C/S write during this same arbitration restarts the process. If other C/S writes are performed, Tx arbitration process is pending. If there is no arbitration winner after the arbitration process has finished, then the TX arbitration machine begins a new arbitration process. If there is a pending arbitration and Bus Idle state starts, then an arbitration process is triggered. In this case the first and second C/S write in Bus Idle will not restart the arbitration process. It is possible that there is not enough time

to finish arbitration in Wait For Bus Idle state and the next state is Idle. In this case the scan is not interrupted, and it is completed during Bus Idle state. During this arbitration C/S write does not cause arbitration restart.
- Arbitration winner deactivation during a valid arbitration window.
- Upon exiting Freeze mode (first bit of the Wait For Bus Idle state). If there is a re-synchronization during Wait For Bus Idle, the arbitration process is restarted.

Arbitration process stops in the following situations:

- All mailboxes were scanned.
- A Tx active mailbox is found if lowest buffer feature is enabled.
- Arbitration winner inactivation or abort during any arbitration process.
- There was not enough time to finish Tx arbitration process (for instance, when a deactivation was performed near the end of frame). In this case arbitration process is pending.
- Error or overload flag in the bus.
- Low Power or Freeze mode request in Idle state.

Arbitration is considered pending as described below:

- It was not possible to finish arbitration process in time.
- C/S write during arbitration if write is performed in a MB whose number is lower than the Tx arbitration pointer.
- Any C/S write if there is no Tx arbitration process in progress.
- Rx Match has just updated a Rx code to Tx code.
- Entering Bus Off state.

C/S write during arbitration has the following effect:

- If C/S write is performed in the arbitration winner, a new process is restarted immediately.
- If C/S write is performed in a MB whose number is higher than the Tx arbitration pointer, the ongoing arbitration process will scan this MB as normal.

45.3.4 Receive process

To be able to receive CAN frames into a mailbox, the CPU must prepare it for reception by executing the following steps:

1. If the mailbox is active (either Tx or Rx) inactivate the mailbox (see Mailbox inactivation), preferably with a safe inactivation (see Transmission abort mechanism).
2. Write the ID word into the mailbox.

3. Write the EMPTY code (0100b) to the CODE field of the Control and Status word to activate the mailbox. No setup is required for EDL, BRS, and ESI bits—they are overwritten by the respective bit fields in the received message.

After the MB is activated, it will be able to receive frames that match the programmed filter. At the end of a successful reception, the mailbox is updated by the move-in process (see Move-in) as follows:

1. The received data field (8 bytes at most for Classical CAN message format and up to 64 bytes for CAN FD message format) is stored.
2. The received Identifier field is stored.
3. The value of the Free Running Timer at the time of the second bit of frame's Identifier field is written into the mailbox's Time Stamp field.
4. The received SRR, IDE, RTR, EDL, BRS, ESI, and DLC fields are stored.
5. The CODE field in the Control and Status word is updated (see Table 45-27 and Table 45-28 in Section Message buffer structure).
6. A status flag is set in the Interrupt Flag Register and an interrupt is generated if allowed by the corresponding Interrupt Mask Register bit.

The recommended way for the CPU to service (read) the frame received in a mailbox is by the following procedure:

1. Read the Control and Status word of that mailbox.
2. Check if the BUSY bit is deasserted, indicating that the mailbox is locked. Repeat step 1) while it is asserted. See Mailbox lock mechanism.
3. Read the contents of the mailbox. After the mailbox is locked, its contents won't be modified by FlexCAN move-in processes. See Move-in.
4. Acknowledge the proper flag at IFLAG registers.
5. Read the free running timer to unlock the mailbox.

The CPU should poll for frame reception by the status flag bit for the specific mailbox in one of the IFLAG registers and not by the CODE field of that mailbox. Polling the CODE field does not work because after a frame is received and the CPU services the mailbox (by reading the C/S word followed by unlocking the mailbox), the CODE field will not return to EMPTY. It will remain FULL, as explained in Table 45-27. If the CPU tries to work around this behavior by writing to the C/S word to force an EMPTY code after reading the mailbox without a prior safe inactivation, a newly received frame matching the filter of that mailbox may be lost.

CAUTION

In summary: never do polling by reading directly the C/S word of the mailboxes. Instead, read the IFLAG registers.

Note that the received frame's Identifier field is always stored in the matching mailbox, thus the contents of the ID field in a mailbox may change if the match was due to masking. When MCR[SRXDIS] is asserted, FlexCAN will not store frames transmitted by itself in any MB, even if it contains a matching Rx mailbox, and no interrupt flag or interrupt signal will be generated. Otherwise, when MCR[SRXDIS] is deasserted, FlexCAN can receive frames transmitted by itself if a matching Rx mailbox exists.

To be able to receive CAN frames through the Rx FIFO, the CPU must enable and configure the Rx FIFO during Freeze mode (see Rx FIFO). Upon receiving the Frames Available in Rx FIFO interrupt (see the description of IFLAG1[BUF5I] "Frames available in Rx FIFO"), the CPU should service the received frame using the following procedure:

1. Read the Control and Status word (optional: needed only if a mask was used for IDE and RTR bits).
2. Read the ID field (optional: needed only if a mask was used).
3. Read the data field.
4. Read the RXFIR register. (This step is optional, and is executed only if the IDHIT (Identifier Acceptance Filter Hit Indicator) is required in the application.)
5. Clear the Frames Available in Rx FIFO interrupt by writing one to IFLAG1[BUF5I] (mandatory: releases the MB and allows the CPU to read the next Rx FIFO entry).

When MCR[DMA] is asserted, upon receiving a frame in FIFO, IFLAG1[BUF5I] generates a DMA request and does not generate a CPU interrupt (see Rx FIFO under DMA operation). The IMASK1 bits in Rx FIFO region are not used.

The DMA controller must service the received frame using the following procedure:

1. Read the Control and Status word (read 80h address, optional).
2. Read the ID field (read 84h address, optional).
3. Read all data bytes (start read at 88h address, optional).
4. Read the last data bytes (read 8Ch address is mandatory).

45.3.5 Matching process

The matching process scans the MB memory looking for Rx MBs programmed with the same ID as the one received from the CAN bus. If the FIFO is enabled, the priority of scanning can be selected between mailboxes and FIFO filters. The matching starts from the lowest number message buffer toward the higher ones. If no match is found within the first structure then the other is scanned subsequently. In the event that the FIFO is full, the matching algorithm always looks for a matching MB outside the FIFO region.

For Rx FIFO see Rx FIFO.

As the frame is being received, it is stored in a hidden auxiliary MB called Rx Serial Message Buffer (Rx SMB).

The matching process start point depends on the following conditions:

- If the received frame is a remote frame, the start point is the CRC field of the frame.
- If the received frame is a data frame with DLC field equal to zero, the start point is the CRC field of the frame.
- If the received frame is a data frame with DLC field different than zero, the start point is the DATA field of the frame.

If a matching ID is found in the FIFO table or in one of the mailboxes, the contents of the Rx SMB are transferred to the FIFO or to the matched mailbox by the move-in process. If any CAN protocol error is detected then no match results are transferred to the FIFO or to the matched mailbox at the end of reception.

The matching process scans all matching elements of both Rx FIFO (if enabled) and the active Rx mailboxes (CODE is EMPTY, FULL, OVERRUN, or RANSWER) in search of a successful comparison with the matching elements of the Rx SMB that is receiving the frame on the CAN bus. The Rx SMB has the same structure as a mailbox. The reception structures (Rx FIFO or mailboxes) associated with the matching elements that had a successful comparison are the *matched structures*. The *matching winner* is selected at the end of the scan among those matched structures and depends on conditions described ahead. See the following table.

Table 45-7. Matching architecture

Structure	SMB[RTR]	CTRL2[RRS]	CTRL2[EACEN]	MB[IDE]	MB[RTR]	MB[ID[1]]	MB[CODE]
Mailbox	0	—	0	cmp[2]	no_cmp[3]	cmp_msk[4]	EMPTY or FULL or OVERRUN
Mailbox	0	—	1	cmp_msk	cmp_msk	cmp_msk	EMPTY or FULL or OVERRUN
Mailbox	1	0	—	cmp	no_cmp	cmp	RANSWER
Mailbox	1	1	0	cmp	no_cmp	cmp_msk	EMPTY or FULL or OVERRUN
Mailbox	1	1	1	cmp_msk	cmp_msk	cmp_msk	EMPTY or FULL or OVERRUN
FIFO[5]	—	—	—	cmp_msk	cmp_msk	cmp_msk	—

1. For mailbox structure, If SMB[IDE] is asserted, the ID is 29 bits (ID Standard + ID Extended). If SMB[IDE] is negated, the ID is only 11 bits (ID Standard). For FIFO structure, the ID depends on IDAM.
2. cmp: Compares the Rx SMB contents with the MB contents regardless the masks.

3. no_cmp: The Rx SMB contents are not compared with the MB contents.
4. cmp_msk: Compares the Rx SMB contents with MB contents taking into account the masks.
5. SMB[IDE] and SMB[RTR] are not taken into account when IDAM is type C.

A reception structure is free-to-receive when any of the following conditions is satisfied:

- The CODE field of the mailbox is EMPTY.
- The CODE field of the mailbox is either FULL or OVERRUN and it has already been serviced (the C/S word was read by the CPU and unlocked as described in Mailbox lock mechanism).
- The CODE field of the mailbox is either FULL or OVERRUN and an inactivation (see Mailbox inactivation) is performed.
- The Rx FIFO is not full.

The scan order for mailboxes and Rx FIFO is from the matching element with lowest number to the higher ones.

The matching winner search for mailboxes is affected by MCR[IRMQ]. If it is negated, the matching winner is the first matched mailbox regardless if it is free-to-receive or not. If it is asserted, the matching winner is selected according to the priority below:

1. the first free-to-receive matched mailbox;
2. the last non free-to-receive matched mailbox.

It is possible to select the priority of scan between mailboxes and Rx FIFO with CTRL2[MRP].

If the selected priority is Rx FIFO first:

- If the Rx FIFO is a matched structure and is free-to-receive, then the Rx FIFO is the matching winner regardless of the scan for mailboxes.
- Otherwise (the Rx FIFO is not a matched structure or is not free-to-receive), then the matching winner is searched among mailboxes as described above.

If the selected priority is mailboxes first:

- If a free-to-receive matched mailbox is found, it is the matching winner regardless of the scan for Rx FIFO.
- If no matched mailbox is found, then the matching winner is searched in the scan for the Rx FIFO.
- If both conditions above are not satisfied and a non–free-to-receive matched mailbox is found, then the matching winner determination is conditioned by MCR[IRMQ]:
  - If MCR[IRMQ] is negated, the matching winner is the first matched mailbox.
  - If MCR[IRMQ] is asserted, the matching winner is the Rx FIFO if it is a free-to-receive matched structure; otherwise, the matching winner is the last non–free-to-receive matched mailbox.

See the following table for a summary of matching possibilities.

Table 45-8. Matching possibilities and resulting reception structures

RFEN	IRMQ	MRP	Matched in MB	Matched in FIFO	Reception structure	Description
No FIFO, only MB, match is always MB first						
0	0	X[1]	None[2]	—[3]	None	Frame lost by no match
0	0	X	Free[4]	—	First MB	
0	1	X	None	—	None	Frame lost by no match
0	1	X	Free	—	First MB	
0	1	X	Not free	—	Last MB	Overrun
FIFO enabled, no match in FIFO is as if FIFO does not exist						
1	0	X	None	None[5]	None	Frame lost by no match
1	0	X	Free	None	First MB	
1	1	X	None	None	None	Frame lost by no match
1	1	X	Free	None	First MB	
1	1	X	Not free	None	Last MB	Overrun
FIFO enabled, Queue disabled						
1	0	0	X	Not full[6]	FIFO	
1	0	0	None	Full[7]	None	Frame lost by FIFO full (FIFO overflow)
1	0	0	Free	Full	First MB	
1	0	0	Not free	Full	First MB	
1	0	1	None	Not full	FIFO	
1	0	1	None	Full	None	Frame lost by FIFO full (FIFO overflow)
1	0	1	Free	X	First MB	
1	0	1	Not free	X	First MB	Overrun
FIFO enabled, queue enabled						
1	1	0	X	Not full	FIFO	
1	1	0	None	Full	None	Frame lost by FIFO full (FIFO overflow)
1	1	0	Free	Full	First MB	
1	1	0	Not free	Full	Last MB	Overrun
1	1	1	None	Not full	FIFO	
1	1	1	Free	X	First MB	
1	1	1	Not free	Not full	FIFO	
1	1	1	Not free	Full	Last MB	Overrun

1. This is a don't care condition.
2. Matched in MB "None" means that the frame has not matched any MB (free-to-receive or non–free-to-receive).
3. This is a forbidden condition.
4. Matched in MB "Free" means that the frame matched at least one MB free-to-receive regardless of whether it has matched MBs non-free-to-receive.
5. Matched in FIFO "None" means that the frame has not matched any filter in FIFO. It is as if the FIFO didn't exist (CTRL2[RFEN]=0).
6. Matched in FIFO "Not full" means that the frame has matched a FIFO filter and has empty slots to receive it.
7. Matched in FIFO "Full" means that the frame has matched a FIFO filter but couldn't store it because it has no empty slots to receive it.

If a non-safe mailbox inactivation (see Mailbox inactivation) occurs during matching process and the mailbox inactivated is the temporary matching winner, then the temporary matching winner is invalidated. The matching elements scan is not stopped nor restarted—it continues normally. The consequence is that the current matching process works as if the matching elements compared before the inactivation did not exist, therefore a message may be lost.

Suppose, for example, that the FIFO is disabled, IRMQ is enabled, there are two MBs with the same ID, and FlexCAN starts receiving messages with that ID. Let us say that these MBs are the second and the fifth in the array. When the first message arrives, the matching algorithm finds the first match in MB number 2. The code of this MB is EMPTY, so the message is stored there. When the second message arrives, the matching algorithm finds MB number 2 again, but it is not "free-to-receive", so it keeps looking, finds MB number 5 and stores the message there. If yet another message with the same ID arrives, the matching algorithm finds out that there are no matching MBs that are "free-to-receive", so it decides to overwrite the last matched MB, which is number 5. In doing so, it sets the CODE field of the MB to indicate OVERRUN.

The ability to match the same ID in more than one MB can be exploited to implement a reception queue (in addition to the full featured FIFO) to allow more time for the CPU to service the MBs. By programming more than one MB with the same ID, received messages are queued into the MBs. The CPU can examine the Time Stamp field of the MBs to determine the order in which the messages arrived.

Matching to a range of IDs is possible by using ID acceptance masks. FlexCAN supports individual masking per MB (see the description of Rx Individual Mask Registers (RXIMR0 - RXIMR63)). During the matching algorithm, if a mask bit is asserted, then the corresponding ID bit is compared. If the mask bit is negated, the corresponding ID bit is a "don't care". Note that the Individual Mask Registers are implemented in RAM, so they are not initialized out of reset. Also, they can only be programmed when the module is in Freeze mode; otherwise, they are blocked by hardware.

FlexCAN also supports an alternate masking scheme with only four mask registers (RXFGMASK, RXMGMASK, RX14MASK, and RX15MASK) for backwards compatibility with legacy applications. This alternate masking scheme is enabled when the IRMQ bit in the MCR register is negated.

45.3.6 Move process

There are two types of move process: move-in and move-out.

45.3.6.1 Move-in

The move-in process is the copy of a message received by an Rx SMB to an Rx mailbox or FIFO that has matched it. If the move destination is the Rx FIFO, attributes of the message are also copied to the CAN_RXFIR FIFO. Each Rx SMB has its own move-in process, but only one is performed at a given time as described ahead. The move-in starts only when the message held by the Rx SMB has a corresponding matching winner (see Matching process) and all of the following conditions are true:

- The CAN bus has either reached or already gone past:
  - The second bit of Intermission field next to the frame that carried the message that is in the Rx SMB.
  - The first bit of an overload frame next to the frame that carried the message that is in the Rx SMB.
- There is no ongoing matching process.
- The destination mailbox is not locked by the CPU.
- There is no ongoing move-in process from another Rx SMB. If more than one move-in process is to be started at the same time, both are performed and the newest substitutes for the oldest.

The term pending move-in is used throughout the documentation and stands for a move-to-be that still does not satisfy all of the aforementioned conditions.

The move-in is cancelled and the Rx SMB is able to receive another message if any of the following conditions is satisfied:

- The destination mailbox is inactivated after the CAN bus has reached the first bit of Intermission field next to the frame that carried the message and its matching process has finished.
- There is a previous pending move-in to the same destination mailbox.
- The Rx SMB is receiving a frame transmitted by the FlexCAN itself and self-reception is disabled (MCR[SRXDIS] is asserted).
- Any CAN protocol error is detected.

Note that the pending move-in is not cancelled if the module enters Freeze or Low-Power mode. It stays on hold, only waiting for Freeze and Low-Power mode to be exited and the module to be unlocked. If an MB is unlocked during Freeze mode, the move-in happens immediately.

The move-in process is the execution by the FlexCAN of the following steps:

1. Push IDHIT into the RXFIR FIFO if the message is destined for the Rx FIFO.
2. Read all data words from the Rx SMB in accordance with the selected payload size for the Rx storage element.
3. Write all data words to the Rx mailbox in accordance with the selected payload size for the Rx storage element. If the data size of the storage element is smaller than the original payload size described in the message's DLC field, the payload is truncated and the high order bytes that do not fit the destination size are lost.
4. Read the Control/Status and ID words from the Rx SMB.
5. Write Control/Status and ID words to the Rx mailbox, and update the CODE field.

The move-in process is not atomic, in such a way that it is immediately cancelled by the inactivation of the destination mailbox (see Mailbox inactivation). In this case the mailbox may remain partially updated, thus incoherent. The exception is if the move-in destination is a Rx FIFO message buffer; then the process cannot be cancelled.

The BUSY Bit (least significant bit of the CODE field) of the destination message buffer is asserted while the move-in is being performed to alert the CPU that the message buffer content is temporarily incoherent.

45.3.6.2 Move-out

The move-out process is the copy of the content from a Tx Mailbox to the Tx SMB when a message for transmission is available (see Arbitration process). The move-out occurs in the following conditions:

- The first bit of Intermission field
- During Bus Off state when TX Error Counter is in the 124 to 128 range
- During Bus Idle state
- During Wait For Bus Idle state

The move-out process is not atomic. Only the CPU has priority to access the memory concurrently out of Bus Idle state. In Bus Idle, the move-out has the lowest priority to the concurrent memory accesses.

45.3.7 Data coherence

In order to maintain data coherency and FlexCAN proper operation, the CPU must obey the rules described in Transmit process and Receive process.

45.3.7.1 Transmission abort mechanism

The abort mechanism provides a safe way to request the abort of a pending transmission. A feedback mechanism is provided to inform the CPU if the transmission was aborted or if the frame could not be aborted and was transmitted instead.

Two primary conditions must be fulfilled in order to abort a transmission:

- MCR[AEN] must be asserted.
- The first CPU action must be the writing of abort code (1001b) into the CODE field of the Control and Status word.

Active MBs configured for transmission must be aborted first before they can be updated. If the abort code is written to a Mailbox that is currently being transmitted or to a Mailbox that was already loaded into the Tx SMB for transmission, the write operation is blocked and the transmission is not disturbed. However, the abort request is captured and kept pending until one of the following conditions is satisfied:

- The module loses the bus arbitration.
- There is an error during the transmission.
- The module is put into Freeze mode.
- The module enters Bus Off state.
- There is an overload frame.

If none of the conditions above are reached:

- The MB is transmitted correctly.
- The interrupt flag is set in the IFLAG register.
- An interrupt to the CPU is generated (if enabled).

The abort request is automatically cleared when the interrupt flag is set. On the other hand, if only one of the above conditions is reached, the frame is not transmitted; therefore:

- The abort code is written into the CODE field.
- The interrupt flag is set in the IFLAG.
- An interrupt is (optionally) generated to the CPU.

If the CPU writes the abort code before the transmission begins internally, then the write operation is not blocked; therefore, the MB is updated and the interrupt flag is set. In this way the CPU only needs to read the abort code to make sure the active MB was safely

inactivated. Although the AEN bit is asserted and the CPU wrote the abort code, in this case the MB is inactivated and not aborted, because the transmission did not start yet. One Mailbox is aborted only when the abort request is captured and kept pending until one of the previous conditions is satisfied.

45.3.7.2 Mailbox inactivation

Inactivation is a mechanism provided to protect the mailbox against updates by the FlexCAN internal processes, thus allowing the CPU to rely on mailbox data coherence after having updated it, even in Normal mode.

Inactivation of transmission mailboxes must be performed just when MCR[AEN] is deasserted.

If a mailbox is inactivated, it participates in neither the arbitration process nor the matching process until it is reactivated. See Transmit process and Receive process for more detailed instructions on how to inactivate and reactivate a mailbox.

To inactivate a mailbox, the CPU must update its CODE field to INACTIVE (either 0b or 1000b).

Because you will not be able to synchronize the CODE field update with the FlexCAN internal processes, an inactivation can have the following consequences:

- A frame in the bus that matches the filtering of the inactivated Rx mailbox may be lost without notice, even if there are other mailboxes with the same filter.
- A frame containing the message within the inactivated Tx mailbox may be transmitted without setting the respective IFLAG.

In order to perform a safe inactivation and avoid the above consequences for Tx mailboxes, the CPU must use the transmission abort mechanism (see Transmission abort mechanism).

The inactivation automatically unlocks the mailbox (see Mailbox lock mechanism).

NOTE

Message buffers that are part of the Rx FIFO cannot be inactivated. There is no write protection on the FIFO region by FlexCAN. CPU must maintain data coherency in the FIFO region when RFEN is asserted.

45.3.7.3 Mailbox lock mechanism

Other than mailbox inactivation, FlexCAN has another data coherence mechanism for the receive process. When the CPU reads the Control and Status word of an Rx MB with codes FULL or OVERRUN, FlexCAN assumes that the CPU wants to read the whole MB in an atomic operation, and therefore it sets an internal lock flag for that MB. The lock is released when the CPU reads the free running timer (global unlock operation), or when it reads the Control and Status word of another MB regardless of its code. A CPU write into the C/S word also unlocks the MB, but this procedure is not recommended for normal unlock use because it cancels a pending move and potentially may lose a received message. The MB locking prevents a new frame from being written into the MB while the CPU is reading it.

NOTE

The locking mechanism applies only to Rx MBs that are not part of the FIFO and have a code different than INACTIVE (0b) or EMPTY[1] (0100b). Also, Tx MBs cannot be locked.

Suppose, for example, that the FIFO is disabled and the second and the fifth MBs of the array are programmed with the same ID, and FlexCAN has already received and stored messages into these two MBs. Suppose now that the CPU decides to read MB number 5 and at the same time another message with the same ID is arriving. When the CPU reads the Control and Status word of MB number 5, this MB is locked. The new message arrives and the matching algorithm finds out that there are no free-to-receive MBs, so it decides to override MB number 5. However, this MB is locked, so the new message cannot be written there. It will remain in the Rx SMB waiting for the MB to be unlocked, and only then will be written to the MB.

If the MB is not unlocked in time and yet another new message with the same ID arrives, then the new message overwrites the one on the Rx SMB and there will be no indication of lost messages either in the CODE field of the MB or in the Error and Status Register.

While the message is being moved in from the Rx SMB to the MB, the BUSY bit on the CODE field is asserted. If the CPU reads the Control and Status word and finds out that the BUSY bit is set, it should defer accessing the MB until the BUSY bit is negated.

Note

If the BUSY bit is asserted or if the MB is empty, then reading the Control and Status word does not lock the MB.

1. In previous FlexCAN versions, reading the C/S word locked the MB even if it was EMPTY. This behavior is maintained when the IRMQ bit is negated.

Inactivation takes precedence over locking. If the CPU inactivates a locked Rx MB, then its lock status is negated and the MB is marked as invalid for the current matching round. Any pending message on the Rx SMB will not be transferred anymore to the MB. An MB is unlocked when the CPU reads the Free Running Timer Register (see Free Running Timer (TIMER)), or the C/S word of another MB.

Lock and unlock mechanisms have the same functionality in both Normal and Freeze modes.

An unlock during Normal or Freeze mode results in the move-in of the pending message. However, the move-in is postponed if an unlock occurs during a low power mode (see Modes of operation), and it takes place only when the module returns to Normal or Freeze modes.

45.3.8 Rx FIFO

The Rx FIFO is receive-only and is enabled by asserting MCR[RFEN]. The reset value of this bit is zero to maintain software backward compatibility with previous versions of the module that did not have the FIFO feature.

CAUTION

Rx FIFO must not be enabled when CAN FD feature is enabled.

The FIFO is 6-message deep. The memory region occupied by the FIFO structure (both message buffers and FIFO engine) is described in Rx FIFO structure. The CPU can read the received messages sequentially, in the order they were received, by repeatedly reading a message buffer structure at the output of the FIFO.

IFLAG1[BUF5I] (Frames available in Rx FIFO) is asserted when there is at least one frame available to be read from the FIFO. An interrupt is generated if it is enabled by the corresponding mask bit. Upon receiving the interrupt, the CPU can read the message (accessing the output of the FIFO as a message buffer) and the RXFIR register, and then clear the interrupt. If there are more messages in the FIFO the act of clearing the interrupt updates the output of the FIFO with the next message and updates RXFIR with the attributes of that message, reissuing the interrupt to the CPU. Otherwise, the flag remains negated. The output of the FIFO is valid only while IFLAG1[BUF5I] is asserted.

IFLAG1[BUF6I] (Rx FIFO Warning) is asserted when the number of unread messages within the Rx FIFO is increased to five from four due to the reception of a new one, meaning that the Rx FIFO is almost full. The flag remains asserted until the CPU clears it.

IFLAG1[BUF7I] (Rx FIFO Overflow) is asserted when an incoming message was lost because the Rx FIFO is full. Note that the flag will not be asserted when the Rx FIFO is full and the message was captured by a Mailbox. The flag remains asserted until the CPU clears it.

Clearing one of those three flags does not affect the state of the other two.

An interrupt is generated if an IFLAG bit is asserted and the corresponding mask bit is asserted too.

A powerful filtering scheme is provided to accept only frames intended for the target application, reducing the interrupt servicing workload. The filtering criteria is specified by programming a table of up to 128 32-bit registers, according to CTRL2[RFFN] setting, that can be configured to one of the following formats (see also Rx FIFO structure):

- Format A: 128 IDAFs (extended or standard IDs including IDE and RTR)
- Format B: 256 IDAFs (standard IDs or extended 14-bit ID slices including IDE and RTR)
- Format C: 512 IDAFs (standard or extended 8-bit ID slices)

Note

A chosen format is applied to all entries of the filter table. It is not possible to mix formats within the table.

Every frame available in the FIFO has a corresponding IDHIT (Identifier Acceptance Filter Hit Indicator) that can be read in the IDHIT field from C/S word, as shown in the Rx FIFO Structure description. Another way the CPU can obtain this information is by accessing the RXFIR register. RXFIR[IDHIT] refers to the message at the output of the FIFO and is valid while the IFLAG1[BUF5I] flag is asserted. The RXFIR register must be read only before clearing the flag, which guarantees that the information refers to the correct frame within the FIFO.

Up to 32 elements of the filter table are individually affected by the Individual Mask Registers (RXIMRx), according to the setting of CTRL2[RFFN], allowing very powerful filtering criteria to be defined. If MCR[IRMQ] is negated, then the FIFO filter table is affected by RXFGMASK.

NOTE

For more information about the difference between FD and non-FD regarding this feature, see Table 45-2.

45.3.8.1 Rx FIFO under DMA operation

The receive-only FIFO can support DMA. This feature is enabled by asserting both MCR[RFEN] and MCR[DMA]. The reset value of MCR[DMA] is zero to maintain backward compatibility with previous versions of the module that did not have the DMA feature.

The DMA controller can read the received message by reading a message buffer structure at the FIFO output port at the 80h–8Ch address range.

NOTE

FlexCAN supports 32-bit access only for DMA transfers.

When MCR[DMA] is asserted the CPU must not access the FIFO output port address range. Before enabling MCR[DMA], the CPU must service the IFLAGs asserted in the Rx FIFO region. Otherwise, these IFLAGs may show that the FIFO has data to be serviced, and mistakenly generate a DMA request. Before disabling MCR[DMA], the CPU must perform a clear FIFO operation.

IFLAG1[BUF5I] (Frames available in Rx FIFO) is asserted when there is at least one frame available to be read from the FIFO. Consequently a DMA request is generated simultaneously. Upon receiving the request, the DMA controller can read the message (accessing the output of the FIFO as a message buffer). The DMA reading process must end by reading address 8Ch, which clears IFLAG1[BUF5I] and updates both the FIFO output with the next message (if FIFO is not empty) and the RXFIR register with the attributes of the new message. If there are more messages stored in the FIFO, IFLAG1[BUF5I] will be re-asserted and another DMA request is issued. Otherwise, the flag remains negated.

NOTE

RXFIR register contents cannot be read after DMA completes the FIFO read. The IDHIT information is also available in the C/S word at address 080h (see Rx FIFO structure.

IFLAG1[BUF6I] and IFLAG1[BUF7I] are not used when the DMA feature is enabled.

When FlexCAN is working with DMA, the CPU does not receive any Rx FIFO interruption and must not clear the related IFLAGs. In addition, the related IMASKs are not used to mask the generation of DMA requests.

NOTE

For more information about the difference between FD and non-FD regarding this feature, see Table 45-2.

45.3.8.2 Clear FIFO operation

When MCR[RFEN] is asserted, the clear FIFO operation is a feature used to empty FIFO contents. With MCR[RFEN] asserted the Clear FIFO occurs when the CPU writes one in IFLAG1[BUF0I]. This operation can only be performed in Freeze mode and is blocked by hardware in other modes. This operation does not clear the FIFO IFLAGs; consequently the CPU must service all FIFO IFLAGs before executing the clear FIFO task.

When Rx FIFO is working with DMA, the clear FIFO operation clears IFLAG1[BUF5I] and the DMA request is canceled.

CAUTION

Clear FIFO operation does not clear IFLAGs, except when MCR[DMA] is asserted; in this case only IFLAG1[BUF5I] is cleared.

45.3.9 CAN protocol related features

This section describes the CAN protocol related features.

45.3.9.1 CAN FD ISO compliance

The CAN FD protocol has been improved to increase the failure detection capability that was in the original CAN FD protocol, which is also called non-ISO CAN FD, by CAN in Automation (CiA). A three-bit stuff counter and a parity bit have been introduced in the improved CAN FD protocol, now called ISO CAN FD. The CRC calculation has also been modified. All these improvements make the ISO CAN FD protocol incompatible with the non-FD CAN FD protocol. The non-ISO CAN FD is still supported by FlexCAN so that it can be used mainly during an intermediate phase, for evaluation and development purposes.

Therefore, it is strongly recommended to configure FlexCAN to the ISO CAN FD protocol by setting the ISOCANFDEN field in the CTRL2 register.

45.3.9.2 CAN FD frames

The ISO 11898-1 standard specifies the Classical Frame format compliant to ISO 11898-1 (2003) and introduces the CAN Flexible Data Rate Frame format. The Classical Frame format allows bit rates up to 1 Mbit/s and payloads up to 8 bytes per frame. The

Flexible Data Rate Frame format allows bit rates higher than 1 Mbit/s and payloads longer than 8 bytes per frame. FlexCAN can receive and transmit CAN FD messages interleaved with Classical CAN messages.

There are three additional control bits in the CAN FD frame. The Extended Data Length (EDL) bit enables a longer data payload with different data length coding. The Bit Rate Switch (BRS) bit decides whether the bit rate is switched inside a CAN FD format frame. The Error State Indicator (ESI) flag is transmitted dominant by error active nodes, and recessive by error passive nodes. There are no Remote Frames (see Remote frames) in the CAN FD format. A message configured to transmit a Remote Frame is always sent out in the Classical CAN format. When an FD frame is received and matches a mailbox, the RTR bit in the receiving message buffer is negated. The RTR bit must be considered in classical frames only.

CAN FD messages may be formatted as long frames where the data field exceeds 8 bytes, and may range from 12 up to 64 bytes. They can also be configured to support bit rate switching, where the control field, the data field, and the CRC field of a CAN frame are transmitted with a higher bit rate than the beginning and the end of the frame. Messages in Classical CAN format are limited to transport a maximum payload of 8 bytes at nominal rate. The following figure illustrates the message formats for Classical and FD frames with either standard or extended ID.

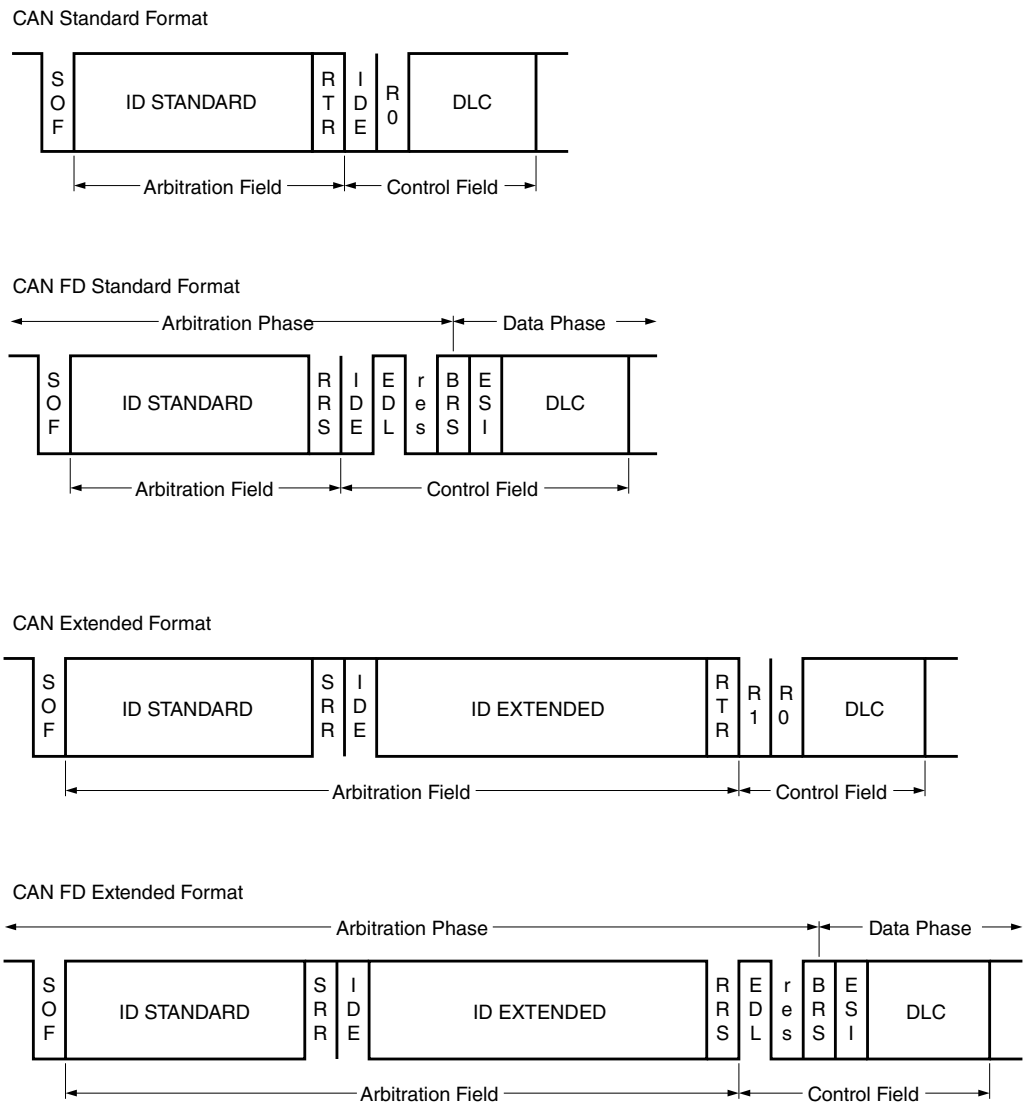


Figure 45-2. CAN message formats

The ability to receive and transmit CAN FD messages is enabled by MCR[FDEN]. Either a recessive R0 bit in CAN frames with 11-bit identifiers or a recessive R1 bit in CAN frames with 29-bit identifiers are decoded as an EDL bit (not a reserved one). A CAN FD frame is recognized by a recessive EDL bit, and a Classical CAN frame is recognized by a dominant EDL bit. The BRS bit specifies whether this frame switches the bit rate in its data phase. A long frame is decoded in accordance with the DLC field value (see DLC definition in Message buffer structure).

CAN FD messages can be transmitted with two different bit rates. The first part of a CAN FD frame, from the Start Of Frame (SOF) bit until the Bit Rate Switch (BRS) bit, also called the arbitration phase, is transmitted with the nominal bit rate based on a set of nominal CAN bit timing configuration values. The second part, from the BRS bit until the CRC Delimiter bit, also named the data phase, is transmitted with the data bit rate

defined by a second set of CAN data bit timing configuration values. Finally, from the CRC Delimiter until the Intermission bits, the transmission returns to nominal bit rate. In CAN FD frames with bit rate switching, the bit timing is changed inside the frame at the sample point of the BRS bit if this bit is recessive. Before the BRS bit, in the CAN FD arbitration phase, the nominal CAN bit timing is used as defined by the CBT register (also by CTRL1 register for backward compatibility). Upon detecting a recessive BRS bit, the CAN data bit timing is used as defined by the FDCBT register.

NOTE

If the length of the time quantum in the nominal bit timing and the length of the time quantum in the data bit timing are not identical, a quantization error of up to one time quantum of the arbitration phase may be present as a phase error. This situation can occur after the switch from arbitration to data phase and will last until the next synchronization event. Thus, the length of the time quantum should be the same in nominal and data bit timing in order to minimize the chance of error frames on the CAN bus, and to optimize the clock tolerance in networks that use FD frames.

FDCTRL[FDRATE] enables the transmission of all frames with bit rate switching if the BRS bit in the selected Tx MB is set. If FDRATE is negated, the transmission is performed at nominal rate regardless of the BRS bit value. FDCTRL[FDRATE] can be written any time but takes effect only for the next message transmitted or received.

The nominal bit timing is resumed at either the sample point of the CRC Delimiter bit or when an error is detected, whichever occurs first. The following figure describes the mechanism for entering and leaving the data phase when BRS bit is recessive.

CAN FD Frame

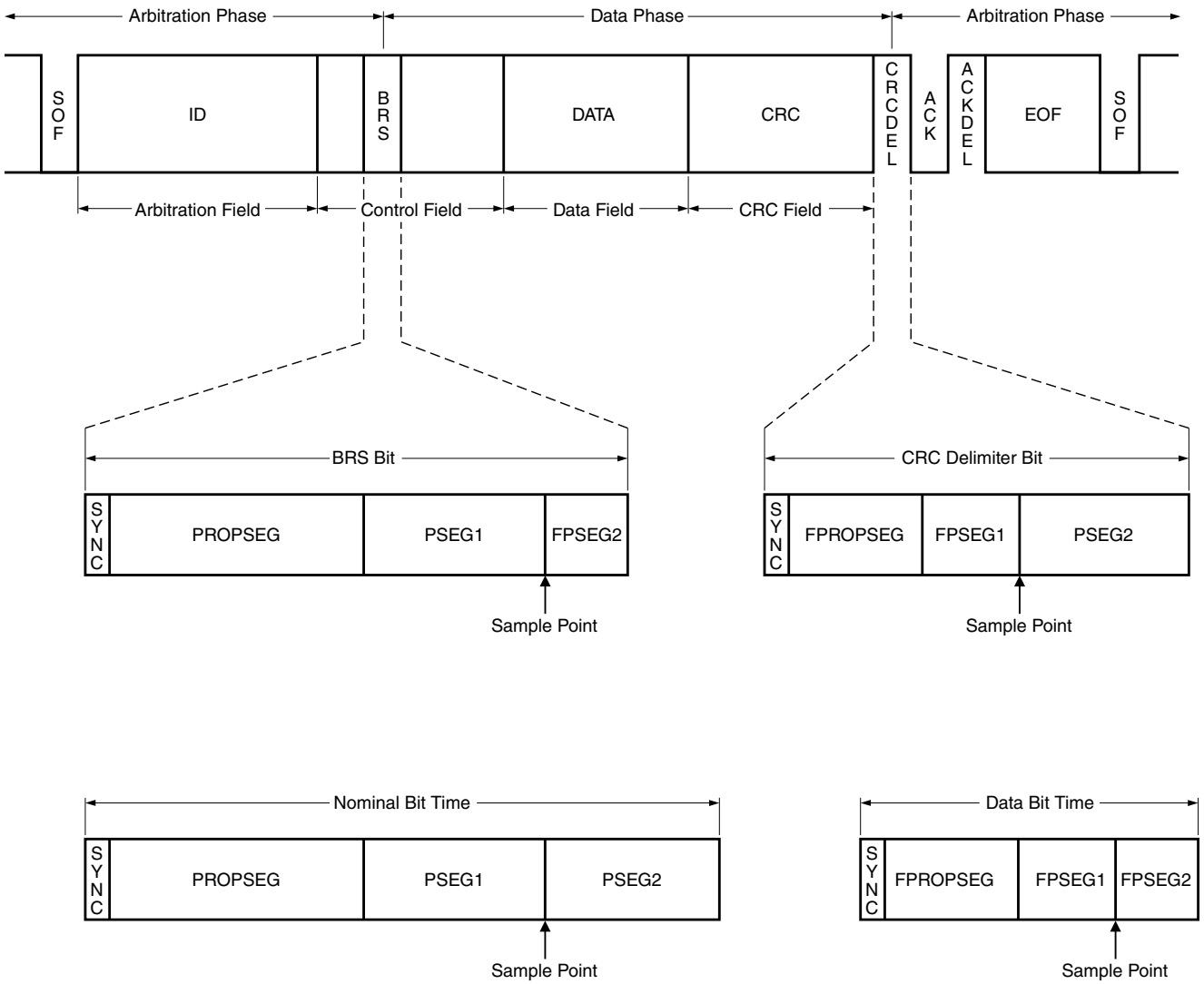


Figure 45-3. Bit rate switching mechanism for CAN FD messages

NOTE

In Classical CAN frames, the CRC delimiter is one single recessive bit. In CAN FD frames, the CRC delimiter may consist of one or two recessive bits. FlexCAN sends only one recessive bit as the CRC delimiter, but it accepts two recessive bits before the edge from recessive to dominant that starts the acknowledge slot. As a receiver, FlexCAN sends its acknowledge bit after the first CRC delimiter bit. In CAN FD frames, FlexCAN accepts a two-bit dominant ACK slot as a valid ACK to compensate for phase shifts between the receivers.

The maximum configurable bit rate in the CAN FD data phase depends on the clock frequency of CAN_PE subblock. For example, with a CAN_PE clock frequency of 40 MHz and the shortest configurable bit time of 5 time quanta, the bit rate in the data phase is 8 Mbit/s.[3]

The value of the ESI bit is determined either by the transmitter's error state at the start of the transmission, if the frame is originated in the FlexCAN node, or by the original transmitting node in case FlexCAN is acting as a gateway for the message. If the transmitter is error passive, ESI is transmitted recessive; otherwise, it is transmitted dominant. The permutations of the relationship between the written value and the transmitted value of the ESI are shown in this table.

Table 45-9. Written versus transmitted values of ESI field

FlexCAN fault confinement status at start of frame	ESI bit Of Tx MB	Transmitted ESI
Error active	0	0 (Error Active)
Error passive	0	1 (Error Passive)
Error active	1	1 (Error Passive)
Error passive	1	1 (Error Passive)

There are different CRC polynomials for different CAN frame formats. The first polynomial, CRC_15, is used for all frames in Classical CAN format. The second, CRC_17, is used for frames in CAN FD format with a data field up to sixteen bytes long. The third, CRC_21, is used for frames in CAN FD format with a data field longer than sixteen bytes. Each polynomial results in a Hamming distance of 6. At the start of the frame, all three CRC polynomials are calculated concurrently. The CRC sequence to be transmitted is selected by the values of the EDL bit and the DLC bit field. When receiving a message, FlexCAN decodes EDL and DLC to select the adequate CRC polynomial to check for a CRC error.

In CAN FD format frames, stuff bits are included in the bit stream for CRC calculation. In Classical CAN format frames, stuff bits are not included. After the transmission of the last bit relevant to the CRC calculation, the FDCRC register stores the calculated CRC for the transmitted message, with the adequate length in accordance to the type of message, for both CAN FD and non-FD messages. The CRCR register reports a valid CRC for Classical CAN messages only.

In CAN FD format frames, the CAN bit stuffing method is changed for the CRC sequence so that the stuff bits are inserted at fixed positions. When FlexCAN is transmitting a CAN FD frame, a fixed stuff bit is inserted just before the first bit of the CRC sequence, even if the last bits of the preceding field do not fulfill the CAN stuff

3. The frequency used in this example may not be supported on this chip; it is shown only to demonstrate how the maximum configurable bit rate is calculated.

condition. Additional stuff bits are inserted after each fourth bit of the CRC sequence. The value of any fixed stuff bit is the inverse value of its preceding bit. When FlexCAN is receiving a CAN FD frame, it discards the fixed stuff bits from the bit stream for the CRC check. A stuff error is detected if the fixed stuff bit has the same value as its preceding bit.

FlexCAN detects errors in CAN FD frames the same way as in Classical CAN frames. The error counters RXERRCNT and TXERRCNT in the ECR register accumulate the counts of Rx and Tx errors, respectively, for both FD and non-FD frames indiscriminately. There are two extra error counters (RXERRCNT_FAST and TXERRCNT_FAST) that accumulate Rx and Tx errors occurring in the data phase of CAN FD frames with the BRS bit set only. The rules for updating the error counters are the same for both CAN FD and non-FD frames (see ECR register).

Error Flags BITERR1, BITERR0, ACKERR, CRCERR, FRMERR, and STFERR in the ESR1 register report errors in both CAN FD and non-FD frames. They also generate the ERRINT interrupt if CTRL1[ERRMSK] is asserted. The ESR1 register has additional error flags (BITERR1_FAST, BITERR0_FAST, CRCERR_FAST, FRMERR_FAST, and STFERR_FAST) to individually indicate the occurrence of errors in the data phase of CAN FD frames with the BRS bit set. There is no ACKERR detected in the data phase of a CAN FD frame. Fault confinement status reported in ESR1[FLTCONF] is the same for both CAN FD and Classical CAN frames, and is based on RXERRCNT and TXERRCNT error counters only. Information contained in RXERRCNT_FAST and TXERRCNT_FAST counters may be considered as status to help detect the error nature related to the bit rate value.

When FlexCAN is in the data phase, either transmitting or receiving a CAN FD message, and detects an error, it immediately switches back to the arbitration phase and to the nominal rate to start an error flag.

Resynchronization and hard synchronization occur in CAN FD frames in the same way as in Classical CAN ones. Additionally, a hard synchronization is also performed at the recessive to dominant edge from EDL to R0 in CAN FD format frames. FlexCAN does not resynchronize while transmitting in the CAN FD data phase.

45.3.9.3 Transceiver delay compensation

The CAN FD protocol allows the transmission and reception of data at a higher bit rate than the nominal rate used in the arbitration phase when the message's BRS bit is set. This feature enables the use of rates up to 8 Mbps.

During the data phase of a CAN FD frame, the transmitter detects a bit error if it cannot receive its own latest transmitted bit at the sample point of that bit. When bit rate switching is enabled (BRS bit is asserted), the length of the CAN bit time in the data phase can become shorter than the transceiver's loop delay, thus impeding the correct comparison between the transmitted bit and the received bit within the current CAN bit time interval.

Note that the TDC process defines a secondary sample point where the transmitted bit is correctly compared with the received bit in order to check for bit errors.

The TDC mechanism can be enabled by FDCTRL[TDCEN] and is effective only during the data phase of FD frames having the BRS bit set. It has no effect either on non-FD frames, or on FD frames transmitted at normal bit rate. The TDC is active from the sample point of the BRS bit until the sample point of the CRC Delimiter bit, provided the respective message under transmission has the BRS bit set. When it is active, a comparison is done between the real received bit and the delayed transmitted bit, where the delay is calculated based on the measured transceiver loop delay.

NOTE

The actual value of the CRC Delimiter bit is disregarded by transmitters using the transceiver delay compensation mechanism. A global error at the end of the CRC field will cause the receivers to send error frames that the transmitter will detect during Acknowledge or End of Frame.

For every transmitted FD frame having the BRS bit set, the delay measurement is triggered by the transition from the recessive EDL bit to the dominant R0 bit (as shown in the next figure). The loop delay is measured in Protocol Engine (PE) clock periods (CANCLK, see Protocol timing), from the transmitted EDL-R0 edge to the received EDL-R0 edge. The position of the secondary sample point is defined by the measured loop delay time added to an offset value specified in FDCTRL[TDCOFF]. FDCTRL[TDCVAL] stores the result of this calculation. The TDCVAL value saturates at its maximum value of 63 CANCLK when the delay measurement is too long.

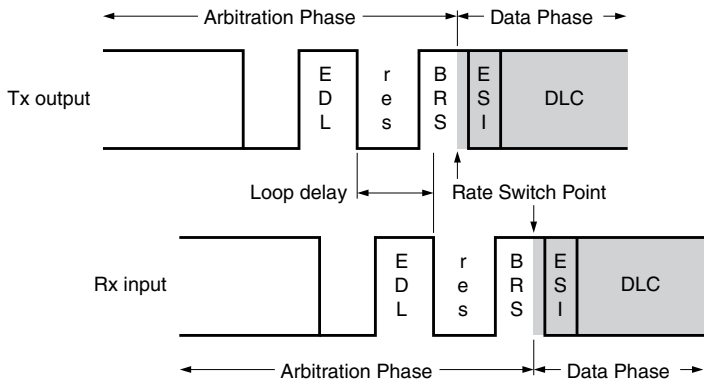


Figure 45-4. Transceiver loop delay measurement

The measured loop delay is not enough to be used to define the secondary sample point because it relates to the CAN bit edges. The transceiver delay compensation offset TDCOFF is used to shift the secondary sample point from the edge to an intermediate point inside the bit time, far away from its edges. Therefore, the TDCOFF value cannot be larger than the CAN bit duration in the data phase.

If the secondary sample point is set very near the CAN bit edge (SYNC field), then problems may occur during the bit sampling in the data phase. For the TDC to work reliably, the offset has to use optimal settings. To be sure the bit sampling is performed in the best region, the TDC offset should be configured as shown in this equation:

$$Offset = (FPSEG1 + FPROPSEG + 2) \times (FPRESDIV + 1)$$

The following figure shows the SSP position when these settings are used.

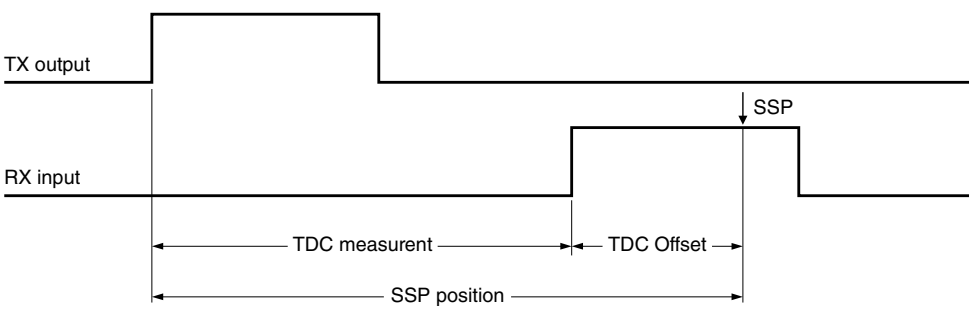


Figure 45-5. SSP position with optimal values

During the data phase of CAN FD frames with bit rate switching enabled, at the onset of every Tx CAN bit, the transmitted Tx bit value is temporarily stored in a buffer and a time countdown based on FDCTRL[TDCVAL] is started which ends with the comparison of the received Rx bit (delayed by the external loop delay plus the specified offset) with the stored Tx bit. If a bit error is detected at the secondary sample point, the FlexCAN issues an error flag to the CAN bus at the next sample point.

During the arbitration phase the delay compensation is always disabled. The maximum delay which can be compensated by the FlexCAN's transceiver delay compensation during the data phase is 3 CAN bit times – 2 Tq. Beyond this limit, the FDCTRL[TDCFAIL] flag is set to indicate when the transceiver delay compensation mechanism is out of range, unable to compensate the transceiver loop delay.

45.3.9.4 Remote frames

A remote frame is a special kind of frame. You can program a mailbox to be a remote request frame by configuring the mailbox as Transmit with the RTR bit set to one. After the remote request frame is transmitted successfully, the mailbox becomes a receive message buffer, with the same ID as before.

When a remote request frame is received by FlexCAN, it can be treated in three ways, depending on remote request storing (CTRL2[RRS]) and Rx FIFO Enable (MCR[RFEN]):

- If RRS is negated the frame's ID is compared to the IDs of the transmit message buffers with the CODE field 1010b. If there is a matching ID, then this mailbox frame will be transmitted. Note that if the matching mailbox has the RTR bit set, then FlexCAN will transmit a remote frame as a response. The received remote request frame is not stored in a receive buffer. It is only used to trigger a transmission of a frame in response. The mask registers are not used in remote frame matching, and all ID bits (except RTR) of the incoming received frame should match. In the case that a remote request frame is received and matches a mailbox, this message buffer immediately enters the internal arbitration process, but is considered as a normal Tx mailbox, with no higher priority. The data length of this frame is independent of the DLC field in the remote frame that initiated its transmission.
- If RRS is asserted the frame's ID is compared to the IDs of the receive mailboxes with the CODE field 0100b, 0010b, or 0110b. If there is a matching ID, then this mailbox will store the remote frame in the same fashion of a data frame. No automatic remote response frame will be generated. The mask registers are used in the matching process.
- If RFEN is asserted FlexCAN will not generate an automatic response for remote request frames that match the FIFO filtering criteria. If the remote frame matches one of the target IDs, it will be stored in the FIFO and presented to the CPU. Note that for filtering formats A and B, it is possible to select whether remote frames are accepted or not. For format C, remote frames are always accepted (if they match the ID). Remote request frames are considered as normal frames, and generate a FIFO overflow when a successful reception occurs and the FIFO is already full.

NOTE

There is no remote frame in the CAN FD format. The RTR bit is replaced by a fixed dominant RRS bit. FlexCAN receives and transmits remote frames in the Classical CAN format.

45.3.9.5 Overload frames

FlexCAN does transmit overload frames when the following conditions are detected on the CAN bus:

- Detection of a dominant bit in the first/second bit of Intermission
- Detection of a dominant bit at the 7th bit (last) of End of Frame field (Rx frames)
- Detection of a dominant bit at the 8th bit (last) of Error Frame Delimiter or Overload Frame Delimiter

45.3.9.6 Time stamp

The value of the free running timer is sampled at the beginning of the Identifier field on the CAN bus, and is stored at the end of move-in in the TIME STAMP field, providing network behavior with respect to time.

When CTRL2[TIMER_SRC] is asserted, the free running timer is continuously clocked by an external time tick.

When CTRL2[TIMER_SRC] is negated, the free running timer is clocked by the FlexCAN bit-clock, which defines the baud rate on the CAN bus. During a message transmission/reception, it increments by one for each bit that is received or transmitted. When there is no message on the bus, it counts using the previously programmed baud rate.

The free running timer is not incremented during Disable, Doze, Stop, and Freeze modes. It can be reset upon a specific frame reception, enabling network time synchronization. See the TSYN description in Control 1 register (CTRL1).

45.3.9.7 Protocol timing

The following figure shows the structure of the clock generation circuitry that feeds the CAN Protocol Engine (PE) submodule.

NOTE

Please see the clock distribution chapter (module clocks table) to identify the proper clock source.

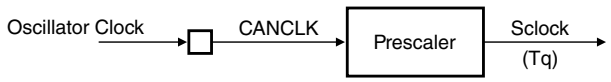


Figure 45-6. CAN engine clocking scheme

The FlexCAN module supports a variety of means to set up bit timing parameters that are required by the CAN protocol. The Control 1 register (CTRL1) has various fields used to control bit timing parameters: PRESDIV, PROPSEG, PSEG1, PSEG2, and RJW.

The CAN Bit Timing register (CBT) extends the range of the CAN bit timing variables in CTRL1. The CAN FD Bit Timing register (FDCBT) provides a second set of CAN bit timing variables to be applied at the data phase of CAN FD frames with the Bit Rate Switch (BRS) set.

NOTE

When the CAN FD feature is enabled, always set CBT[BTF] and configure the CAN bit timing variables in CBT. See CAN Bit Timing Register (CBT).

The PRESDIV field (as well as its extended range EPRESDIV and FDPRESDIV for the data phase bits of CAN FD messages) defines the prescaler value (see the equation below) that generates the serial clock (Sclock), whose period defines the time quantum used to compose the CAN waveform. A time quantum (Tq) is the atomic unit of time managed by the CAN engine.

$$Tq = \frac{(PRESDIV + 1)}{f_{CANCLK}}$$

Figure 45-7. Time quantum

The bit rate, which defines the rate the CAN message is either received or transmitted, is given by the formula:

$$\text{CAN Bit Time} = (\text{Number of Time Quanta in 1 bit time}) * Tq$$

$$\text{Bit Rate} = \frac{1}{\text{CAN Bit Time}}$$

Figure 45-8. CAN bit time and baud rate

A bit time is subdivided into three segments[1] (see Figure 45-9, Figure 45-10 and Table 45-10):

- SYNC_SEG: This segment has a fixed length of one time quantum. Signal edges are expected to happen within this section.
- Time Segment 1: This segment includes the propagation segment and the phase segment 1 of the CAN standard. It can be programmed by setting CTRL1[PROPSEG] and CTRL1[PSEG1] so that their sum (plus 2) is in the range of 2 to 16 time quanta. When CBT[BTF] is asserted, FlexCAN uses EPROPSEG and EPSEG1 fields from CBT register so that their sum (plus 2) is in the range of 2 to 96 time quanta. For messages in CAN FD format with the BRS bit set, FlexCAN uses FDPROPSEG and FDPSEG1 from FDCBT instead, so that their sum (plus 1) is in the range of 2 to 39 time quanta.
- Time Segment 2: This segment represents the phase segment 2 of the CAN standard. It can be programmed by setting CTRL1[PSEG2] (plus 1) to be 2 to 8 time quanta long. When CBT[BTF] is asserted, FlexCAN uses EPSEG2 fields of CBT register so that its value (plus 1) is in the range of 2 to 32 time quanta. For messages in CAN FD format with the BRS bit set, FlexCAN uses FDPSEG2 from FDCBT instead, so that its value (plus 1) is in the range of 2 to 8 time quanta. The time segment 2 cannot be smaller than the Information Processing Time (IPT), which value is 2 time quanta in FlexCAN.

NOTE

The bit time defined by the above time segments must not be smaller than five time quanta. For bit time calculations, use an Information Processing Time (IPT) of two, which is the value implemented in the FlexCAN module.

1. For further explanation of the underlying concepts, see ISO 11898-1. See also the CAN 2.0A/B protocol specification for bit timing.

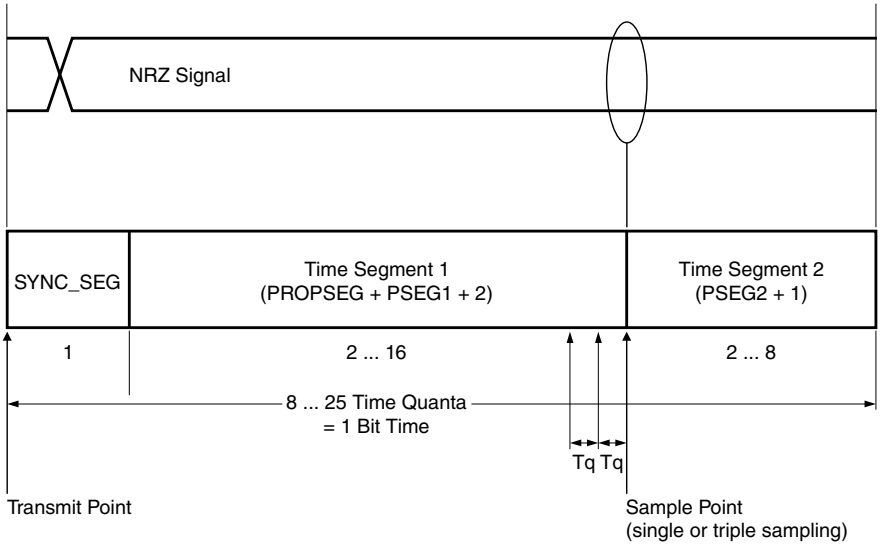


Figure 45-9. Segments within the bit time (example using CTRL1 bit timing variables for Classical CAN format)

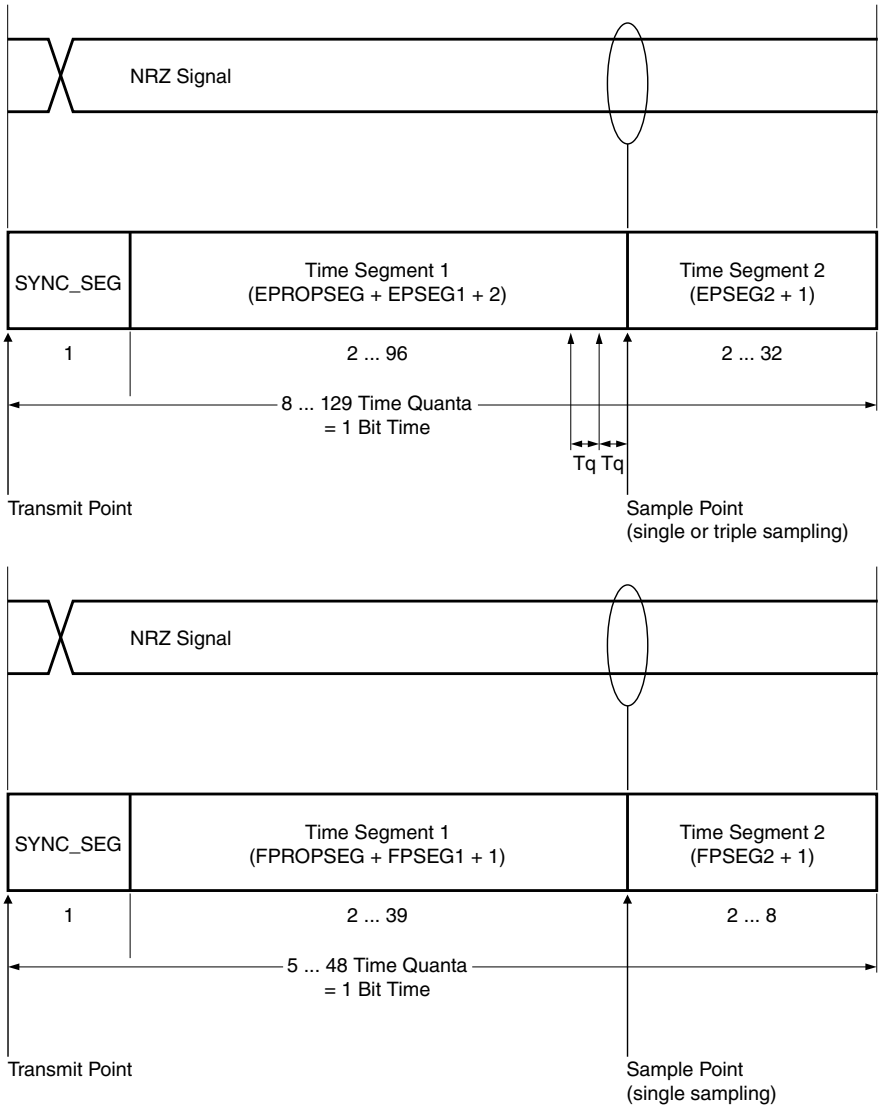


Figure 45-10. Segments within the bit time (example using CBT and FDCBT bit timing variables for CAN FD format)

Table 45-10. Time segment syntax

Syntax	Description
SYNC_SEG	System expects transitions to occur on the bus during this period.
TSEG1	Corresponds to the sum of PROPSEG and PSEG1.
TSEG2	Corresponds to the PSEG2 value.
Transmit point	A node in transmit mode transfers a new value to the CAN bus at this point.
Sample point	A node samples the bus at this point. If the three samples per bit option is selected, then this point marks the position of the third sample.

The following table gives some examples of the CAN compliant segment settings for Classical CAN format (Bosch CAN 2.0B) (non-FD) messages.

Table 45-11. Bosch CAN 2.0B standard compliant bit time segment settings

Time segment 1	Time segment 2	Re-synchronization jump width
5 .. 10	2	1 .. 2
4 .. 11	3	1 .. 3
5 .. 12	4	1 .. 4
6 .. 13	5	1 .. 4
7 .. 14	6	1 .. 4
8 .. 15	7	1 .. 4
9 .. 16	8	1 .. 4

Note

The user must ensure the bit time settings are in compliance with the CAN Protocol standard (ISO 11898-1).

Whenever CAN bit is used as a measure of time duration (for example, estimating the occurrence of a CAN bit event in a message), the number of peripheral clocks in one CAN bit (NumClkBit) can be calculated as:

$$\text{NumClkBit} = \frac{f_{SYS}}{f_{CANCLK}} \times (\text{PRESDIV} + 1) \times (\text{PROPSEG} + \text{PSEG1} + \text{PSEG2} + 4)$$

Figure 45-11. Number of peripheral clocks per CAN bit when CTRL2[BTE] = 0

where:

- NumClkBit is the number of peripheral clocks in one CAN bit.
- f_{CANCLK} is the Protocol Engine (PE) Clock (see Figure 45-6), in Hz.
- f_{SYS} is the frequency of operation of the system (CHI) clock, in Hz.
- PSEG1 is the value in CTRL1[PSEG1] field.
- PSEG2 is the value in CTRL1[PSEG2] field.
- PROPSEG is the value in CTRL1[PROPSEG] field.
- PRESDIV is the value in CTRL1[PRESDIV] field.

The formula above is also applicable to the alternative CAN bit timing variables described in the CAN Bit Timing register (CBT) and also to the CAN FD Bit Timing register (FDCBT).

For example, 180 CAN bits = (180 x NumClkBit) peripheral clock periods.

45.3.9.8 Arbitration and matching timing

During normal reception and transmission, the matching, arbitration, move-in, and move-out processes are executed during certain time windows inside the CAN frame, as shown in the following figures.

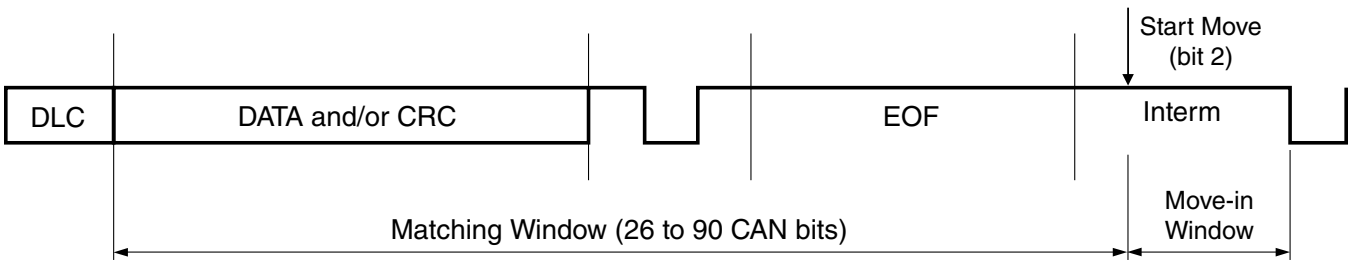


Figure 45-12. Matching and move-in time windows

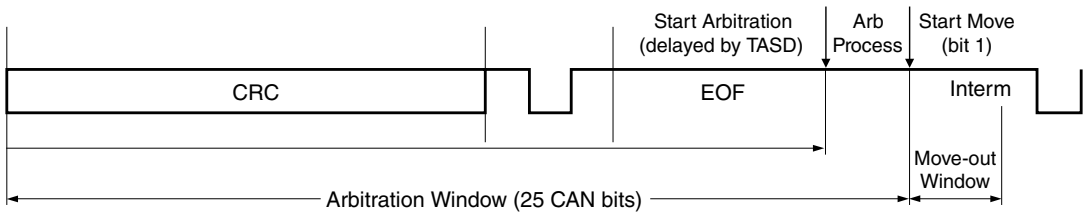


Figure 45-13. Arbitration and move-out time windows

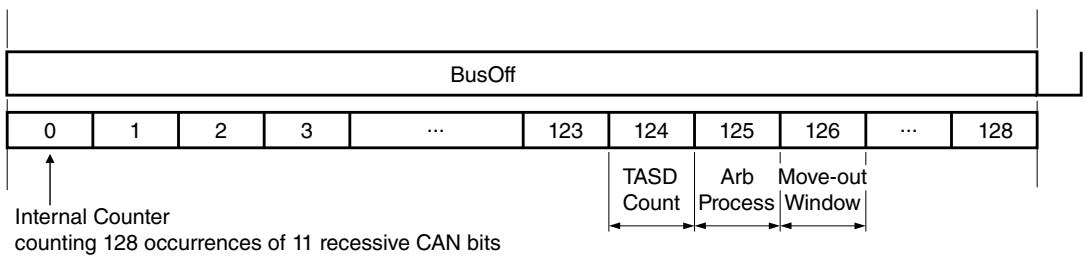


Figure 45-14. Arbitration at the end of bus off and move-out time windows

NOTE

In the preceding figures, the matching and arbitration timing does not take into account the delay caused by the concurrent memory access due to the CPU or other internal FlexCAN subblocks.

45.3.9.9 Tx arbitration start delay

CTRL2[TASD] (Tx Arbitration Start Delay) is a variable that indicates the number of CAN bits used by FlexCAN to delay the Tx arbitration process start point from the first bit of CRC field of the current frame. This variable can be written only in Freeze mode because it is blocked by hardware in other modes.

The transmission performance is impacted by the ability of the CPU to reconfigure message buffers (MBs) for transmission after the end of the internal arbitration process, where FlexCAN finds the winner MB for transmission (see Arbitration process). If the arbitration ends too early before the first bit of Intermission field, then there is a chance that the CPU reconfigures some Tx MBs and the winner MB is no longer the best candidate to be transmitted.

TASD is useful to optimize the transmission performance by defining the arbitration start point, as shown in the next figure, based on factors such as:

- Peripheral-to-oscillator clock ratio
- CAN bit timing variables that determine the CAN bit rate
- Number of message buffers (MBs) in use by the matching and arbitration processes

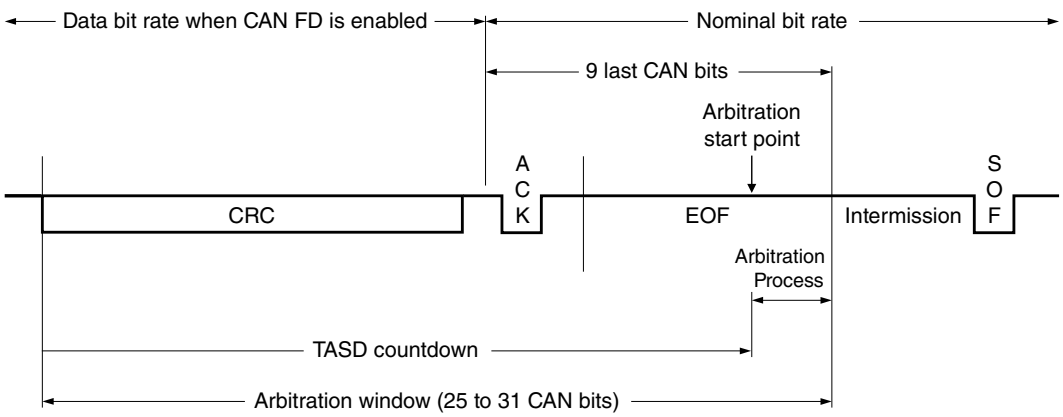


Figure 45-15. Optimal Tx Arbitration start point

The duration of an arbitration process, in terms of CAN bits, is directly proportional to the number of available MBs and to the CAN bit rate, and inversely proportional to the peripheral clock frequency.

The optimal arbitration timing is that in which the last MB is scanned just before the first bit of the Intermission field of a CAN frame. For instance, if there are few MBs and the peripheral/oscillator clock ratio is high and the CAN baud rate is low, then the arbitration can be placed closer to the frame's end, adding more delay to its start point, and vice-versa.

If TASD is set to 0 then the arbitration start is not delayed and more time is reserved for arbitration. On the other hand, if TASD is close to 24 then the CPU can configure a Tx MB later and less time is reserved for arbitration. If too little time is reserved for arbitration the FlexCAN may be not be able to find a winner MB in time to be transmitted with the best chance to win the bus arbitration against external nodes on the CAN bus.

The optimal TASD value can be calculated as follows:

For CAN FD frames and $(MAXMB + 1) \leq NMB_{END}$

$$TASD = 31 - \frac{2 * (MAXMB + 1) + 4}{CPCB_N}$$

For CAN FD frames and $(MAXMB + 1) > NMB_{END}$

$$TASD = 22 - \frac{2 * (MAXMB + 1) - NMB_{END}}{CPCB_F}$$

For non-FD frames

$$TASD = 25 - \frac{2 * (MAXMB + 1) + 4}{CPCB}$$

Figure 45-16. Optimal value for TASD

where:

$$NMB_{END} = \frac{(9 * CPCB_N) - 4}{2}$$

$$BITRATE_N = \left(\frac{f_{CANCLK}}{[1 + (EPSEG1 + 1) + (EPSEG2 + 1) + (EPROPSEG + 1)] \times (EPRESDIV + 1)} \right)$$

$$BITRATE_F = \left(\frac{f_{CANCLK}}{[1 + (FPSEG1 + 1) + (FPSEG2 + 1) + FPROPSEG] \times (FPRESDIV + 1)} \right)$$

$$CPCB_N = \frac{f_{SYS}}{BITRATE_N}$$

$$CPCB_F = \frac{f_{SYS}}{BITRATE_F}$$

$$CPCB = CPCB_N$$

Figure 45-17. Variables used in TASD calculation

- MAXMB is the value in CTRL1[MAXMB].
- NMB_{END} is the number of message buffers that can be scanned by the arbitration process during the 9 last CAN bits at the end of a frame (see the figure above).
- $BITRATE_N$ is the CAN bit rate in bits per second calculated by the nominal CAN bit time variables.
- $BITRATE_F$ is the CAN bit rate in bits per second calculated by the data CAN bit time variables.

- $CPCB_N$ is the number of peripheral clocks per CAN bit in nominal bit rate for CAN FD frames.
- $CPCB_F$ is the number of peripheral clocks per CAN bit in data bit rate for CAN FD frames.
- CPCB is the number of peripheral clocks per CAN bit for non-FD frames.
- f_{CANCLK} is the oscillator clock, in Hz.
- f_{SYS} is the peripheral clock, in Hz.
- EPSEG1 is the value in CBT[EPSEG1] (CTRL1[PSEG1] can also be used).
- EPSEG2 is the value in CBT[EPSEG2] (CTRL1[PSEG2] can also be used).
- EPROPSEG is the value in CBT[EPROPSEG] (CTRL1[PROPSEG] can also be used).
- EPRESDIV is the value in CBT[EPRESDIV] (CTRL1[PRESDIV] can also be used).
- FPSEG1 is the value in FDCBT[FPSEG1].
- FPSEG2 is the value in FDCBT[FPSEG2].
- FPROPSEG is the value in FDCBT[FPROPSEG].
- FPRESDIV is the value in FDCBT[FPRESDIV].

See also Protocol timing for more details.

The following tables give the TASD value calculated for some configuration cases.

Case 1:

- Clock ratio = 2:1 (example: peripheral clock 80 MHz and oscillator clock 40 MHz)
- Bit rate in arbitration phase = 1 Mbaud

Table 45-12. TASD values:

Number of message buffers	TASD value	Maximum bit rate in data phase (MBd)
16	24	Invalid
32	24	8.0
64	23	8.0

Case 2:

- Clock ratio = 1:1 (example: peripheral clock 40 MHz and oscillator clock 40 MHz)
- Bit rate in arbitration phase = 1 Mbaud

Table 45-13. TASD values:

Number of message buffers	TASD value	Maximum bit rate in data phase (MBd)
16	24	Invalid
32	23	6.67

Table continues on the next page...

Table 45-13. TASD values: (continued)

Number of message buffers	TASD value	Maximum bit rate in data phase (MBd)
54	22	5.0
64	21	3.33

Case 3:

- Clock ratio = 2:1 (example: peripheral clock 40 MHz and oscillator clock 20 MHz)
- Bit rate in arbitration phase = 1 Mbaud

Table 45-14. TASD values:

Number of message buffers	TASD value	Maximum bit rate in data phase (MBd)
16	24	Invalid
32	23	4.0
54	22	4.0
64	21	3.33

45.3.10 Clocks

The following table describes the clock sources for FlexCAN. Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 45-15. FlexCAN clocks

Clock name	Description
ipg_clk	Peripheral clock
ipg_clk_chi	Control Host Interface (CHI) clock
ipg_clk_pe	Protocol Engine (PE) clock
ipg_clk_pe_nogate	Protocol Engine clock (no gating)
ipg_clk_s	Peripheral access clock

45.3.10.1 Clock domains and restrictions

The FlexCAN module has two clock domains asynchronous to each other:

- The bus domain feeds the Control Host Interface (CHI) submodule.
- The oscillator domain feeds the CAN Protocol Engine (PE) submodule.

When the two domains are connected to clocks with different frequencies and/or phases, there are restrictions on the frequency relationship between the two clock domains. In the case of asynchronous operation, the bus domain clock frequency must always be greater than the oscillator domain clock frequency.

NOTE

Asynchronous operation with a 1:1 ratio between peripheral and oscillator clocks is not allowed.

When doing matching and arbitration, FlexCAN needs to scan the whole message buffer memory during the time slot of one CAN frame, comprised of a number of CAN bits. In order to have sufficient time to do that, the following requirements must be observed:

- The peripheral clock frequency cannot be smaller than the oscillator clock frequency.
- There must be a minimum number of peripheral clocks per CAN bit, as specified in the table shown below.

Table 45-16. Minimum number of peripheral clocks per CAN bit for Classical CAN format

Number of mailboxes	Value of MCR[RFEN]	Minimum number of peripheral clocks per CAN bit
16	0	16
32	0	16
64	0	25
16	1	16
32	1	17
64	1	30

For classical frame format, the minimum number of peripheral clocks per CAN bit specified in the preceding table determines the minimum peripheral clock frequency for a given number of mailboxes and for an expected CAN bit rate. The CAN bit rate depends on the number of time quanta in a CAN bit, which can be defined by adjusting one or more of the bit timing values contained in either the Control 1 register (CTRL1) or CAN Bit Time register (CBT). The time quantum (Tq) is defined in Protocol timing. The minimum number of time quanta per CAN bit must be 8; therefore, the oscillator clock frequency should be at least 8 times the CAN bit rate.

For CAN FD frame format, there are some constraints that need to be satisfied. The number of peripheral clocks per CAN bit in nominal bit rate (NumClkNomBit) can be calculated by the equation below.

$$\text{NumClkNomBit} = \frac{f_{SYS}}{f_{CANCLK}} \times (\text{PRESDIV} + 1) \times (\text{PROPSEG} + \text{PSEG1} + \text{PSEG2} + 4)$$

$$= \frac{f_{SYS}}{\text{NomBitRate}}$$

Figure 45-18. Number of peripheral clocks per nominal CAN bit

where PRESDIV, PSEG1, and PSEG2 are CAN bit time values in CTRL1 register. Alternatively, EPRESDIV, EPSEG1, and EPSEG2 values in CBT register can be used instead. NumClkNomBit can also be calculated as a function of the expected nominal bit rate used in the arbitration phase (NomBitRate), as shown in the equation above.

The number of CAN bits in the data phase of an FD frame with the BRS bit set (fast CAN bits, in short) depends on the number of data bytes in the payload. The number of fast CAN bits (NumOfFastBits) can be determined in the table below. The fewer the number of data bytes, the fewer the number of fast CAN bits, and less time is available for FlexCAN to scan the whole message buffer memory during the internal matching and arbitration processes.

Table 45-17. Number of fast CAN bits in a CAN FD frame

Minimum number of data bytes	DLC field	NumOfFastBits
0	0h	21
1	1h	29
2	2h	37
3	3h	45
4	4h	53
5	5h	61
6	6h	69
7	7h	77
8	8h	85
12	9h	117
16	Ah	149
20	Bh	186
24	Ch	218
32	Dh	282
48	Eh	410
64	Fh	538

The critical part of a CAN FD frame is during the data phase, where the CAN bit rate is faster than in the arbitration phase. The minimum number of peripheral clocks per fast CAN bit (MinNumClkFastBit) can be calculated to guarantee that enough time is available for FlexCAN to scan the message buffer memory during reception and transmission. The equation below calculates this constraint.

$$\text{MinNumClkFastBit}_A = \frac{(\ 8.5 \times \text{MaxNumOfMb}\ ) + 64 - (\ 9 \times \text{NumClkNomBit}\ )}{\text{NumOfFastBits}}$$

Figure 45-19. Minimum number of peripheral clocks per fast CAN bit for FlexCAN scan process

where MaxNumOfMb is the maximum number of available mailboxes defined in MCR[MAXMB].

The clock domain crossing circuit between the CHI and PE subblocks also imposes a minimum number of peripheral clocks per fast CAN bit for the handshake mechanism to work properly without losing status information through the interface, as shown in the equation below.

$$\text{MinNumClkFastBit}_B = 3 \times \left(1 + \frac{f_{SYS}}{f_{CANCLK}}\right)$$

Figure 45-20. Minimum number of peripheral clocks per fast CAN bit for FlexCAN clock domain interface

Therefore, the minimum number of peripheral clocks per fast CAN bit (MinNumClkFastBit) is determined by the larger of the two values calculated above.

$$\text{MinNumClkFastBit} = \text{Maximum}\ (\ \text{MinNumClkFastBit}_A, \text{MinNumClkFastBit}_B\ )$$

Figure 45-21. Minimum number of peripheral clocks per fast CAN bit

Then, the maximum CAN bit rate in the data phase of CAN FD frames (DataBitRateMAX) can be calculated as below.

$$\text{DataBitrate}_{MAX} = \frac{f_{CANCLK}}{\text{ROUNDUP}\left(\frac{\text{MinNumClkFastBit} \times f_{CANCLK}}{f_{SYS}}\right)}$$

Figure 45-22. Maximum achievable baud rate for data phase

The peripheral and oscillator clock frequencies, the maximum number of mailboxes, and the expected nominal bit rate affect the maximum data bit rate attainable by FlexCAN in CAN FD mode. Also, the data bit rate depends on the minimum payload size of FD frames used in a given application.

To illustrate how the CAN FD bit rate is affected by the configuration of FlexCAN variables, an application example with the peripheral and oscillator clock frequencies set to 50 MHz and 40 MHz, respectively, is considered.

1. Considering the nominal bit rate as 1 Mbps, the number of peripheral clocks per CAN bit in nominal bit rate is calculated as below.

$$\text{NumClkNomBit} = \frac{50 \times 10^6}{1 \times 10^6} = 50$$

Figure 45-23. Calculation example for number of peripheral clocks per nominal CAN bit

2. The number of fast CAN bits (NumOfFastBits) is determined in the table presented above. For example, if the minimum payload in FD frames is 8 bytes, then there are 85 CAN bits in the data phase.
3. Assuming the maximum number of mailboxes is 96, the minimum number of peripheral clocks per fast CAN bit (MinNumClkFastBit) can be calculated.

$$\text{MinNumClkFastBit}_A = \frac{(8.5 \times 96) + 64 - (9 \times 50)}{85} = 5.06$$

Figure 45-24. Calculation example for number of peripheral clocks per fast CAN bit for FlexCAN scan process

$$\text{MinNumClkFastBit}_B = 3 \times \left(1 + \frac{50}{40}\right) = 6.75$$

Figure 45-25. Calculation example for number of peripheral clocks per fast CAN bit for FlexCAN clock domain interface

$$\text{MinNumClkFastBit} = \text{Maximum}\ (5.06, 6.75) = 6.75$$

Figure 45-26. Calculation example for number of peripheral clocks per fast CAN bit

4. The maximum CAN bit rate in the data phase can finally be found.

$$\text{DataBitRate}_{MAX} = \frac{40 \times 10^6}{\text{ROUNDUP}\left(\frac{6.75 \times 40 \times 10^6}{50 \times 10^6}\right)} = 6.667\ \text{Mbps}$$

Figure 45-27. Calculation example for maximum achievable baud rate

As demonstrated in this example, even though the oscillator clock frequency (40 MHz) is adequate to generate a data rate of 8 Mbps in CAN FD mode, the specific FlexCAN configuration limits this rate to 6.667 Mbps. This limitation is mainly due to the low peripheral clock frequency that imposes the MinNumClkFastBitB bound.

The table below shows the maximum data rate for CAN FD according to clock frequencies, payload size, and number of available mailboxes. See in this table that, for some cases, if the number of available mailboxes is reduced, the FlexCAN can then achieve a data rate up to 8 Mbps.

Table 45-18. Maximum CAN bit rate in data phase on CAN FD frames

Peripheral clock frequency (MHz)	Payload size	Number of available mailboxes	Maximum data rate (Mbps)
40	8	94	6.667

Table continues on the next page...

Table 45-18. Maximum CAN bit rate in data phase on CAN FD frames (continued)

Peripheral clock frequency (MHz)	Payload size	Number of available mailboxes	Maximum data rate (Mbps)
40	8	114	5.0
40	12	117	6.667
40	12	128	5.714
50	12 to 64	128	6.667
60	8	126	8.0
60	12	128	8.0
67	6	128	8.0
80	3	128	8.0
100	0	128	8.0

45.3.11 Reset

You can reset FlexCAN in the following ways:

1. Chip level hard reset, which resets all memory-mapped registers asynchronously.
2. MCR[SOFTRST], which resets some of the memory-mapped registers synchronously. See Table 45-21 to see what registers are affected by soft reset.
3. Chip level soft reset, which has the same effect as MCR[SOFTRST].

Soft reset is synchronous and has to follow an internal request/acknowledge procedure across clock domains. Therefore, it may take some time to fully propagate its effects. MCR[SOFTRST] remains asserted while soft reset is pending, so software can poll this bit to know when the reset has completed. Also, soft reset cannot be applied while clocks are shut down in a low-power mode. The low-power mode should be exited and the clocks resumed before applying soft reset.

When the module is enabled (MCR[MDIS] negated), FlexCAN automatically enters Freeze mode. In Freeze mode, FlexCAN is un-synchronized to the CAN bus, MCR[HALT] and MCR[FRZ] are set, the internal state machines are disabled, then MCR[FRZACK] and MCR[NOTRDY] are set. The Tx pin is in recessive state and FlexCAN does not initiate any transmission or reception of CAN frames. Note that the message buffers and the Rx Individual Mask registers are not affected by reset, so they are not automatically initialized.

45.3.12 Interrupts

The module has many interrupt sources: interrupts due to message buffers and interrupts due to the ORed interrupts from MBs, Bus Off, Bus Off Done, Error, Error Fast (errors detected in the data phase of CAN FD format messages with the BRS bit set), Wake Up, Tx Warning, and Rx Warning.

Each one of the message buffers can be an interrupt source, if its corresponding IMASK bit is set. There is no distinction between Tx and Rx interrupts for a particular buffer, under the assumption that the buffer is initialized for either transmission or reception. Each of the buffers has an assigned flag bit in the IFLAG registers. The bit is set when the corresponding buffer completes a successful transfer and is cleared when the CPU writes one to it (unless another interrupt is generated at the same time).

Note

It must be guaranteed that the CPU clears only the bit causing the current interrupt. For this reason, bit manipulation instructions (BSET) must not be used to clear interrupt flags. These instructions may cause accidental clearing of interrupt flags which are set after entering the current interrupt service routine.

If the Rx FIFO is enabled (MCR[RFEN] = 1) and DMA is disabled (MCR[DMA] = 0), the interrupts corresponding to MBs 0 to 7 have different meanings. Bit 7 of the IFLAG1 register becomes the "FIFO Overflow" flag; bit 6 becomes the "FIFO Warning" flag, bit 5 becomes the "Frames Available in FIFO" flag and bits 4-0 are unused. See the description of Interrupt Flags 1 Register (IFLAG1) for more information.

If both Rx FIFO and DMA are enabled (MCR[RFEN] and MCR[DMA] = 1) the FlexCAN does not generate any FIFO interrupt. Bit 5 of the IFLAG1 register still indicates "Frames Available in FIFO" and generates a DMA request. Bits 7, 6, 4-0 are unused.

CAUTION

FIFO cannot be enabled when CAN FD feature is enabled.

For a combined interrupt where multiple MB interrupt sources are OR'd together, the interrupt is generated when any of the associated MBs (or FIFO, if applicable) generates an interrupt. In this case, the CPU must read the IFLAG registers to determine which MB or FIFO source caused the interrupt.

The following interrupt sources generate interrupts like the MB interrupt sources, and can be read from ESR1 register

- Bus Off
- Bus Off Done

- Error
- Error Fast
- Wake Up
- Tx Warning
- Rx Warning

The Bus Off, Error, Tx Warning, and Rx Warning interrupt mask bits are located in the CTRL1 register; the Wake-Up interrupt mask bit is located in MCR.

45.3.13 Bus interface

CPU access to FlexCAN registers is subject to the following rules:

- Unrestricted read and write access to supervisor registers (registers identified with S/U in Table 45-21 in Supervisor Mode or with S only) results in an access error.
- Read and write access to implemented reserved address space results in an access error.
- Write access to positions whose bits are all currently read-only results in an access error. If at least one of the bits is not read-only then no access error is issued. Write permission to positions or some of their bits can change depending on the mode of operation or transitory state. See register and field descriptions for details.
- Read and write access to unimplemented address space results in access error.
- Read and write access to RAM located positions during Low Power mode results in an access error.
- It is possible for the RXIMR memory region to be considered as general purpose memory and available for access. There are two ways of doing this:
  a. If MCR[IRMQ] is cleared, the individual masks (RXIMR) are disabled. In this case the RXIMR memory region is considered as general purpose memory.
  b. If MCR[MAXMB] is programmed with a value smaller than the available number of MBs, then the unused memory space can be used as general purpose RAM space. Note that reserved words within RAM cannot be used. As an example, suppose FlexCAN's RAM can support up to 16 MBs, CTRL2[RFFN] is 0h, and MCR[MAXMB] is programmed with zero. The maximum number of MBs in this case becomes one. The RAM starts at 0080h, and the space 0080h–008Fh is used by the one MB. The memory space 0090h–017Fh is available. The space 0180h–087Fh is reserved. The space 0880h–0883h is used by the one individual mask and the available memory in the mask registers space would be 0884h–08BFh. From 08C0h–09DFh there are reserved words for internal use which cannot be used as general purpose RAM. As a general rule, free memory space for general purpose depends only on MAXMB.

c. If MCR[FDEN] is enabled, general purpose memory can be used out of Freeze mode only.

Table 45-19. Access permissions

Supervisor access	0				1		
Supervisor mode - MCR[SUPV] = 1	0			1	any		
Modes of operation	Normal	Freeze	Low power	*	Normal	Freeze	Low power
MCR	bus error	bus error	bus error	bus error	read/write	read/write	read/write
CTRL1	read/write	read write	read/write	bus error	read/write	read/write	read/write
TIMER	read/write	read/write	read/write	bus error	read/write	read/write	read/write
TCR	bus error	bus error	bus error	bus error	bus error	bus error	bus error
RXMGMASK[1]	bus error for write operation	read/write	bus error for write operation	bus error	bus error for write operation	read/write	bus error for write operation
RX14MASK[1]	bus error for write operation	read/write	bus error for write operation	bus error	bus error for write operation	read/write	bus error for write operation
RX15MASK[1]	bus error for write operation	read/write	bus error for write operation	bus error	bus error for write operation	read/write	bus error for write operation
ECR	bus error for write operation	read/write	bus error for write operation	bus error	bus error for write operation	read/write	bus error for write operation
ESR1	read/write	read/write	read/write	bus error	read/write	read/write	read/write
IMASK2	read/write	read/write	read/write	bus error	read/write	read/write	read/write
IMASK1	read/write	read/write	read/write	bus error	read/write	read/write	read/write
IFLAG2	read/write	read/write	read/write	bus error	read/write	read/write	read/write
IFLAG1	read/write	read/write	read/write	bus error	read/write	read/write	read/write
CTRL2	read/write	read/write	read/write	bus error	read/write	read/write	read/write
ESR2	read/write	read/write	read/write	bus error	read/write	read/write	read/write
CRCR	bus error for write operation	bus error for write operation	bus error for write operation	bus error	bus error for write operation	bus error for write operation	bus error for write operation
RXFGMASK[1]	bus error for write operation	read/write	bus error for write operation	bus error	bus error for write operation	read/write	bus error for write operation
RXFIR[1]	bus error for write operation	bus error for write operation	bus error for write operation	bus error	bus error for write operation	bus error for write operation	bus error for write operation
CBT	bus error for write operation	read/write	bus error for write operation	bus error	bus error for write operation	read/write	bus error for write operation
DBG1	bus error for write operation	bus error for write operation	bus error for write operation	bus error for write operation	bus error for write operation	bus error for write operation	bus error for write operation

Table continues on the next page...

Table 45-19. Access permissions (continued)

Supervisor access	0			1			
Supervisor mode - MCR[SUPV] = 1	0			1	any		
Modes of operation	Normal	Freeze	Low power	*	Normal	Freeze	Low power
DBG2	bus error for write operation	bus error for write operation	bus error for write operation	bus error for write operation	bus error for write operation	bus error for write operation	bus error for write operation
MB[1, 2]	read/write	read/write	read/write	bus error	read/write	read/write	read/write
FIFO header[1]	read/write	read/write	read/write	bus error	read/write	read/write	read/write
FIFO reserved space[1]	bus error	bus error	bus error	bus error	bus error	bus error	bus error
FIFO filters[1]	read/write	read/write	read/write	bus error	read/write	read/write	read/write
RXIMR[1]	bus error	read/write	bus error	bus error	bus error	read/write	bus error
FDCTRL	read/write	read/write	read/write	bus error	read/write	read/write	read/write
FDCBT	read/write [3]	read/write	read/write [3]	bus error	read/write [3]	read/write	read/write [3]
FDCRC	bus error for write operation	bus error for write operation	bus error for write operation	bus error	bus error for write operation	bus error for write operation	bus error for write operation
General purpose RAM[1]	read/write	read/write	read/write	bus error	read/write	read/write	read/write
Reserved space (used)[1]	bus error	bus error	bus error	bus error	bus error	bus error	bus error
Reserved space (empty)[1]	bus error	bus error	bus error	bus error	bus error	bus error	bus error

1. Access in low power is only possible if RAM_clk and ipg_clk are enabled.
2. If MCR[RFEN] = 1, see FIFO access rules below.
3. Write operation has no effect.

45.4 FlexCAN signal descriptions

The FlexCAN module has two I/O signals connected to the external chip pins. These signals are summarized in the following table and described in more detail in the next subsections.

Table 45-20. FlexCAN signal descriptions

Signal	Description	I/O
CAN Rx	CAN receive pin	Input
CAN Tx	CAN transmit pin	Output

45.4.1 CAN Rx

This pin is the receive pin from the CAN bus transceiver. Dominant state is represented by logic level 0. Recessive state is represented by logic level 1.

45.4.2 CAN Tx

This pin is the transmit pin to the CAN bus transceiver. Dominant state is represented by logic level 0. Recessive state is represented by logic level 1.

45.5 Initialization/application information

This section provides instructions for initializing the FlexCAN module.

45.5.1 FlexCAN initialization sequence

For any configuration change/initialization it is required that FlexCAN be put into Freeze mode (see Freeze mode). Note that the module needs to be initialized after every reset.

The following is a generic initialization sequence applicable to the FlexCAN module:

1. Initialize the Module Configuration register (MCR).
   a. Enable the individual filtering per MB and reception queue features by setting IRMQ.
   b. Enable the warning interrupts by setting WRNEN.
   c. If required, disable frame self reception by setting SRXDIS.
   d. Enable the Rx FIFO by setting RFEN.
   e. If Rx FIFO is enabled and DMA is required, set DMA.
   f. Enable the abort mechanism by setting AEN.
   g. Enable the local priority feature by setting LPRIOEN.
2. Initialize the Control 1 register (CTRL1) and optionally the CAN Bit Timing register (CBT). Initialize also the CAN FD CAN Bit Timing register (FDCBT).
   a. Determine the bit timing parameters: PROPSEG, PSEG1, PSEG2, and RJW.
   b. Optionally determine the bit timing parameters: EPROPSEG, EPSEG1, EPSEG2, and ERJW.
   c. Determine the CAN FD bit timing parameters: FPROPSEG, FPSEG1, FPSEG2, and FRJW.
   d. Determine the bit rate by programming the PRESDIV field and optionally the EPRESDIV field.
   e. Determine the CAN FD bit rate by programming the FPRESDIV field.

f. Determine the internal arbitration mode (LBUF).

3. Initialize the message buffers.
   a. The control and status word of all message buffers must be initialized.
   b. If Rx FIFO was enabled, the ID filter table must be initialized.
   c. Other entries in each message buffer should be initialized as required.
4. Initialize the Rx Individual Mask registers (RXIMRn).
5. Set required interrupt mask bits in
   - IMASK registers (for all MB interrupts)
   - MCR register (for Wake-Up interrupt)
   - CTRL1 / CTRL2 registers (for Bus Off and Error interrupts)
6. Negate MCR[HALT].

After the last step listed above, FlexCAN attempts to synchronize to the CAN bus.

45.6 Memory map/register definition

This section describes the registers and data structures in the FlexCAN module. The base address of the module depends on the particular memory map of the chip.

45.6.1 FlexCAN memory mapping

The address space occupied by FlexCAN has 128 bytes for registers starting at the module base address, followed by embedded RAM starting at address offset 0080h.

Each individual register is identified by its complete name and the corresponding mnemonic . The access type can be Supervisor (S) or Unrestricted (U). Most of the registers can be configured to have either Supervisor or Unrestricted access by programming MCR[SUPV]. These registers are identified as S/U in the Access column of Table 45-21.

NOTE

An invalid register access will result in a bus error. This includes reading a write-only register, writing a read-only register, or accessing an invalid address.

Table 45-21. Register access and reset information

Register	Access type	Affected by hard reset	Affected by soft reset
Module Configuration Register (MCR)	S	Yes	Yes
Control 1 Register (CTRL1)	S/U	Yes	No

Table continues on the next page...

Table 45-21. Register access and reset information (continued)

Register	Access type	Affected by hard reset	Affected by soft reset
Free Running Timer Register (TIMER)	S/U	Yes	Yes
Rx Mailboxes Global Mask Register (RXMGMASK)	S/U	No	No
Rx Buffer 14 Mask Register (RX14MASK)	S/U	No	No
Rx Buffer 15 Mask Register (RX15MASK)	S/U	No	No
Error Counter Register (ECR)	S/U	Yes	Yes
Error and Status 1 Register (ESR1)	S/U	Yes	Yes
Interrupt Masks 2 Register (IMASK2)	S/U	Yes	Yes
Interrupt Masks 1 Register (IMASK1)	S/U	Yes	Yes
Interrupt Flags 2 Register (IFLAG2)	S/U	Yes	Yes
Interrupt Flags 1 Register (IFLAG1)	S/U	Yes	Yes
Control 2 Register (CTRL2)	S/U	Yes	No
Error and Status 2 Register (ESR2)	S/U	Yes	Yes
CRC Register (CRCR)	S/U	Yes	Yes
Rx FIFO Global Mask Register (RXFGMASK)	S/U	No	No
Rx FIFO Information Register (RXFIR)	S/U	No	No
CAN Bit Timing Register (CBT)	S/U	Yes	No
Message buffers	S/U	No	No
Rx Individual Mask Registers	S/U	No	No
CAN FD Control Register (FDCTRL)	S/U	Yes	No
CAN FD Bit Timing Register (FDCBT)	S/U	Yes	No
CAN FD CRC Register (FDCRC)	S/U	Yes	Yes

The FlexCAN module can store CAN messages for transmission and reception using mailboxes and Rx FIFO structures.

45.6.2 CAN register descriptions

The table below shows the FlexCAN memory map.

The address range from offset 80h–47Fh allocates the sixty-four 128-bit message buffers (MBs).

The memory maps for the message buffers are in FlexCAN message buffer memory map.

45.6.2.1 CAN memory map

FLEXCAN3 base address: 401D_8000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Module Configuration Register (MCR)	32	RW	5980_000Fh
4h	Control 1 Register (CTRL1)	32	RW	0000_0000h
8h	Free Running Timer (TIMER)	32	RW	0000_0000h
10h	Rx Mailboxes Global Mask Register (RXMGMASK)	32	RW	Table 45-21
14h	Rx 14 Mask Register (RX14MASK)	32	RW	Table 45-21
18h	Rx 15 Mask Register (RX15MASK)	32	RW	Table 45-21
1Ch	Error Counter (ECR)	32	RW	0000_0000h
20h	Error and Status 1 Register (ESR1)	32	R2C	0000_0000h
24h	Interrupt Masks 2 Register (IMASK2)	32	RW	0000_0000h
28h	Interrupt Masks 1 Register (IMASK1)	32	RW	0000_0000h
2Ch	Interrupt Flags 2 Register (IFLAG2)	32	W1C	0000_0000h
30h	Interrupt Flags 1 Register (IFLAG1)	32	W1C	0000_0000h
34h	Control 2 Register (CTRL2)	32	RW	0080_0000h
38h	Error and Status 2 Register (ESR2)	32	RO	0000_0000h
44h	CRC Register (CRCR)	32	RO	0000_0000h
48h	Rx FIFO Global Mask Register (RXFGMASK)	32	RW	Table 45-24
4Ch	Rx FIFO Information Register (RXFIR)	32	RO	Table 45-24
50h	CAN Bit Timing Register (CBT)	32	RW	0000_0000h
880h - 97Ch	Rx Individual Mask Registers (RXIMR0 - RXIMR63)	32	RW	Table 45-24
C00h	CAN FD Control Register (FDCTRL)	32	RW	8000_0100h
C04h	CAN FD Bit Timing Register (FDCBT)	32	RW	0000_0000h
C08h	CAN FD CRC Register (FDCRC)	32	RO	0000_0000h

45.6.2.2 Module Configuration Register (MCR)

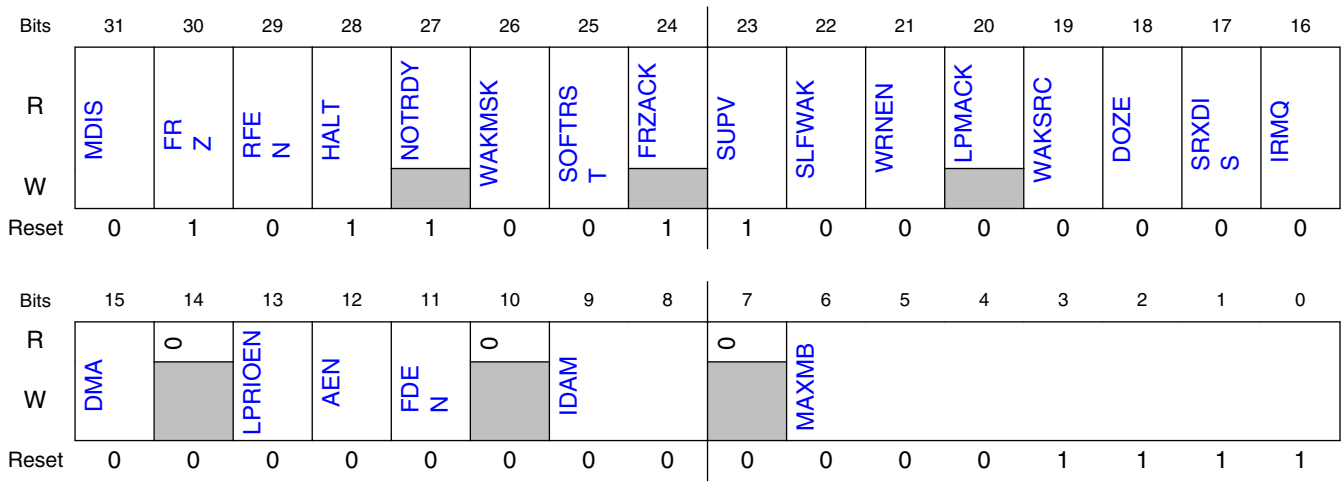
45.6.2.2.1 Offset

Register	Offset
MCR	0h

45.6.2.2.2 Function

This register defines global system configurations, such as the module operation modes and the maximum message buffer configuration.

45.6.2.2.3 Diagram



45.6.2.2.4 Fields

Field	Function
31 MDIS	Module Disable This bit controls whether FlexCAN is enabled or not. When disabled, FlexCAN disables the clocks to the CAN Protocol Engine and Controller Host Interface submodules. This bit is not affected by soft reset. 0b - Enable the FlexCAN module. 1b - Disable the FlexCAN module.
30 FRZ	Freeze Enable The FRZ bit specifies the FlexCAN behavior when MCR[HALT] is set or when Debug mode is requested at chip level. When FRZ is asserted, FlexCAN is enabled to enter Freeze mode. Negation of this bit field causes FlexCAN to exit from Freeze mode. 0b - Not enabled to enter Freeze mode. 1b - Enabled to enter Freeze mode.
29 RFEN	Rx FIFO Enable This bit controls whether the Rx FIFO feature is enabled or not. When RFEN is set, MBs 0 to 5 cannot be used for normal reception and transmission because the corresponding memory region (80h–DCh) is used by the FIFO engine as well as additional MBs (up to 32, depending on CTRL2[RFFN] setting) which are used as Rx FIFO ID filter table elements. RFEN also impacts the definition of the minimum number of peripheral clocks per CAN bit as described in Table 45-16. This bit can be written in Freeze mode only, because it is blocked by hardware in other modes. **NOTE:** This bit cannot be set when CAN FD operation is enabled (see FDEN bit). 0b - Rx FIFO not enabled. 1b - Rx FIFO enabled.
28	Halt FlexCAN

Table continues on the next page...

Field	Function
HALT	Assertion of this bit puts the FlexCAN module into Freeze mode. The CPU should clear it after initializing the message buffers and the Control registers CTRL1 and CTRL2. No reception or transmission is performed by FlexCAN before this bit is cleared. Freeze mode cannot be entered when FlexCAN is in a low power mode. 0b - No Freeze mode request. 1b - Enters Freeze mode if the FRZ bit is asserted.
27 NOTRDY	FlexCAN Not Ready This read-only bit indicates that FlexCAN is either in Disable mode, Doze mode, Stop mode or Freeze mode. It is negated when FlexCAN has exited these modes. This bit is not affected by soft reset. 0b - FlexCAN module is either in Normal mode, Listen-Only mode, or Loop-Back mode. 1b - FlexCAN module is either in Disable mode, Doze mode, Stop mode, or Freeze mode.
26 WAKMSK	Wake Up Interrupt Mask This bit enables the Wake Up interrupt generation under Self Wake Up mechanism. 0b - Wake Up interrupt is disabled. 1b - Wake Up interrupt is enabled.
25 SOFTRST	Soft Reset When SOFTRST is asserted, FlexCAN resets its internal state machines and some of the memory-mapped registers. SOFTRST can be asserted directly by the CPU when it writes to the MCR register, but it is also asserted when global soft reset is requested at chip level . Because soft reset is synchronous and has to follow a request/acknowledge procedure across clock domains, it may take some time to fully propagate its effect. The SOFTRST bit remains asserted when reset is pending, and is automatically negated when reset completes. Therefore, software can poll this bit to know when the soft reset has completed. Soft reset cannot be applied when clocks are shut down in a low power mode. The module should be first removed from low power mode, and then soft reset can be applied. This bit is not affected by soft reset. 0b - No reset request. 1b - Resets the registers affected by soft reset.
24 FRZACK	Freeze Mode Acknowledge This read-only bit indicates that FlexCAN is in Freeze mode and its prescaler is stopped. The Freeze mode request cannot be granted until current transmission or reception processes have finished. Therefore the software can poll the FRZACK bit to know when FlexCAN has actually entered Freeze mode. If Freeze mode request is negated, then this bit is negated after the FlexCAN prescaler is running again. If Freeze mode is requested when FlexCAN is in a low power mode, then the FRZACK bit will be set only when the low-power mode is exited. See Freeze mode. This bit is not affected by soft reset. **NOTE:** FRZACK will be asserted within 178 CAN bits from the Freeze mode request by the CPU, and negated within 2 CAN bits after the Freeze mode request removal (see Protocol timing). 0b - FlexCAN not in Freeze mode, prescaler running. 1b - FlexCAN in Freeze mode, prescaler stopped.
23 SUPV	Supervisor Mode This bit configures the FlexCAN to be either in Supervisor or User mode. The registers affected by this bit are marked as S/U in the "Access type" column of Table 45-21. Reset value of this bit is 1, so the affected registers start with Supervisor access allowance only. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. 0b - FlexCAN is in User mode. Affected registers allow both Supervisor and Unrestricted accesses. 1b - FlexCAN is in Supervisor mode. Affected registers allow only Supervisor access. Unrestricted access behaves as though the access was done to an unimplemented register location.
22 SLFWAK	Self Wake Up This bit enables the Self Wake Up feature when FlexCAN is in a low-power mode other than Disable mode. When this feature is enabled, the FlexCAN module monitors the bus for wakeup event, that is, a recessive-to-dominant transition.

Table continues on the next page...

Field	Function
	If a wakeup event is detected during Doze mode, FlexCAN requests to resume its clocks and, if enabled to do so, generates a Wake Up interrupt to the CPU. If a wakeup event is detected during Stop mode, then FlexCAN generates, if enabled to do so, a Wake Up interrupt to the CPU so that it can exit Stop mode globally and FlexCAN can request to resume the clocks. When FlexCAN is in a low-power mode other than Disable mode, this bit cannot be written as it is blocked by hardware. 0b - FlexCAN Self Wake Up feature is disabled. 1b - FlexCAN Self Wake Up feature is enabled.
21 WRNEN	Warning Interrupt Enable When asserted, this bit enables the generation of the TWRNINT and RWRNINT flags in the Error and Status Register 1 (ESR1). If WRNEN is negated, the TWRNINT and RWRNINT flags will always be zero, independent of the values of the error counters, and no warning interrupt will ever be generated. This bit can be written in Freeze mode only, because it is blocked by hardware in other modes. 0b - TWRNINT and RWRNINT bits are zero, independent of the values in the error counters. 1b - TWRNINT and RWRNINT bits are set when the respective error counter transitions from less than 96 to greater than or equal to 96.
20 LPMACK	Low-Power Mode Acknowledge This read-only bit indicates that FlexCAN is in a low-power mode (Disable mode, Doze mode, Stop mode). A low-power mode cannot be entered until all current transmission or reception processes have finished, so the CPU can poll the LPMACK bit to know when FlexCAN has actually entered low power mode. This bit is not affected by soft reset. **NOTE:** LPMACK will be asserted within 180 CAN bits from the low-power mode request by the CPU, and negated within 2 CAN bits after the low-power mode request removal (see Protocol timing). 0b - FlexCAN is not in a low-power mode. 1b - FlexCAN is in a low-power mode.
19 WAKSRC	Wake Up Source This bit defines whether the integrated low-pass filter is applied to protect the Rx CAN input from spurious wakeup. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. 0b - FlexCAN uses the unfiltered Rx input to detect recessive to dominant edges on the CAN bus. 1b - FlexCAN uses the filtered Rx input to detect recessive to dominant edges on the CAN bus.
18 DOZE	Doze Mode Enable This bit defines whether FlexCAN is allowed to enter low-power mode when Doze mode is requested at chip level. This bit is automatically reset when FlexCAN wakes up from Doze mode upon detecting activity on the CAN bus (self wakeup enabled). 0b - FlexCAN is not enabled to enter low-power mode when Doze mode is requested. 1b - FlexCAN is enabled to enter low-power mode when Doze mode is requested.
17 SRXDIS	Self Reception Disable This bit defines whether FlexCAN is allowed to receive frames transmitted by itself. If this bit is asserted, frames transmitted by the module will not be stored in any MB, regardless if the MB is programmed with an ID that matches the transmitted frame, and no interrupt flag or interrupt signal will be generated due to the frame reception. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. 0b - Self-reception enabled. 1b - Self-reception disabled.
16 IRMQ	Individual Rx Masking And Queue Enable This bit indicates whether Rx matching process will be based either on individual masking and queue or on masking scheme with RXMGMASK, RX14MASK, RX15MASK, and RXFGMASK. This bit can be written in Freeze mode only because it is blocked by hardware in other modes.

Table continues on the next page...

Field	Function
	0b - Individual Rx masking and queue feature are disabled. For backward compatibility with legacy applications, the reading of C/S word locks the MB even if it is EMPTY. 1b - Individual Rx masking and queue feature are enabled.
15 DMA	DMA Enable DMA controls whether the DMA feature is enabled or not. The DMA feature can only be used in Rx FIFO, so consequently MCR[RFEN] must be asserted. When DMA and RFEN are set, IFLAG1[BUF5I] generates the DMA request and no RX FIFO interrupt is generated. This bit can be written in Freeze mode only as it is blocked by hardware in other modes. 0b - DMA feature for RX FIFO disabled. 1b - DMA feature for RX FIFO enabled.
14 —	Reserved
13 LPRIOEN	Local Priority Enable This bit is provided for backwards compatibility with legacy applications. It controls whether the local priority feature is enabled or not. It is used to expand the ID used during the arbitration process. With this expanded ID concept, the arbitration process is done based on the full 32-bit word, but the actual transmitted ID still has 11-bit for standard frames and 29-bit for extended frames. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. 0b - Local Priority disabled. 1b - Local Priority enabled.
12 AEN	Abort Enable When asserted, this bit enables the Tx abort mechanism. This mechanism guarantees a safe procedure for aborting a pending transmission, so that no frame is sent in the CAN bus without notification. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. **NOTE:** When MCR[AEN] is asserted, only the abort mechanism (see Transmission abort mechanism) must be used for updating mailboxes configured for transmission. **CAUTION:** Writing the Abort code into Rx mailboxes can cause unpredictable results when MCR[AEN] is asserted. 0b - Abort disabled. 1b - Abort enabled.
11 FDEN	CAN FD operation enable This bit enables the CAN with Flexible Data rate (CAN FD) operation. This bit can be written in Freeze mode only. **NOTE:** FlexCAN is able to transmit FD frame format according to ISO 11898-1. **NOTE:** The Rx FIFO Enable (RFEN) bit cannot be set if FDEN is asserted. 0b - CAN FD is disabled. FlexCAN is able to receive and transmit messages in CAN 2.0 format. 1b - CAN FD is enabled. FlexCAN is able to receive and transmit messages in both CAN FD and CAN 2.0 formats.
10 —	Reserved
9-8 IDAM	ID Acceptance Mode This 2-bit field identifies the format of the Rx FIFO ID filter table elements. Note that all elements of the table are configured at the same time by this field (they are all the same format). See Rx FIFO structure. This field can be written only in Freeze mode because it is blocked by hardware in other modes. 00b - Format A: One full ID (standard and extended) per ID filter table element. 01b - Format B: Two full standard IDs or two partial 14-bit (standard and extended) IDs per ID filter table element. 10b - Format C: Four partial 8-bit standard IDs per ID filter table element.

Table continues on the next page...

Field	Function
	11b - Format D: All frames rejected.
7 —	Reserved
6-0 MAXMB	Number Of The Last Message Buffer This 7-bit field defines the number of the last message buffers that will take part in the matching and arbitration processes. The reset value (0Fh) is equivalent to a 16 MB configuration. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Number of the last MB = MAXMB **NOTE:** MAXMB must be programmed with a value smaller than or equal to the number of available message buffers, as described in FlexCAN memory partition for CAN FD. Additionally, the definition of MAXMB value must take into account the region of MBs occupied by Rx FIFO and its ID filters table space defined by CTRL2[RFFN]. MAXMB also impacts the definition of the minimum number of peripheral clocks per CAN bit as described in Table 45-16.

45.6.2.3 Control 1 Register (CTRL1)

45.6.2.3.1 Offset

Register	Offset
CTRL1	4h

45.6.2.3.2 Function

This Register is defined for specific FlexCAN control features related to the CAN bus, such as bit rate, programmable sampling point within an Rx bit, Loop Back mode, Listen-Only mode, Bus Off recovery behavior, and interrupt enabling (Bus-Off, Error, Warning). It also determines the division factor for the clock prescaler.

The CAN bit timing variables (PRESDIV, PROPSEG, PSEG1, PSEG2, and RJW) can also be configured in the CBT register, which extends the range of all these variables. If CBT[BTF] is set, PRESDIV, PROPSEG, PSEG1, PSEG2, and RJW fields of CTRL1 become read only.

NOTE

When the CAN FD feature is enabled, do not use the PRESDIV, RJW, PSEG1, PSEG2, and PROPSEG fields of the CTRL1 register for CAN bit timing. Instead use the CBT register's EPRESDIV, ERJW, EPSEG1, EPSEG2, and EPROPSEG fields.

NOTE

The CAN bit variables in CTRL1 and in CBT are stored in the same register.

The contents of this register are not affected by soft reset.

45.6.2.3.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PRESDIV								RJW		PSEG1			PSEG2		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BOFFMSK	ERRMSK	Reserved	LPB	TWRNMSK	RWRNMSK	Reserved	Reserved	SMP	BOFFREC	TSYN	LBUF	LOM	PROPSEG		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

45.6.2.3.4 Fields

Field	Function
31-24 PRESDIV	Prescaler Division Factor This 8-bit field defines the ratio between the PE clock frequency and the serial clock (Sclock) frequency. The Sclock period defines the time quantum of the CAN protocol. For the reset value, the Sclock frequency is equal to the PE clock frequency. The maximum value of this field is FFh, which gives a minimum Sclock frequency equal to the PE clock frequency divided by 256. See Protocol timing for more information. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Sclock frequency = PE clock frequency / (PRESDIV + 1).
23-22 RJW	Resync Jump Width This 2-bit field defines the maximum number of time quanta that a bit time can be changed by one resynchronization. One time quantum is equal to the Sclock period. The valid programmable values are 0–3. See Protocol timing for more information. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Resync Jump Width = RJW + 1.
21-19 PSEG1	Phase Segment 1 This 3-bit field defines the length of phase segment 1 in the bit time. The valid programmable values are 0–7. See Protocol timing for more information. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Phase Buffer Segment 1 = (PSEG1 + 1) × Time-Quanta.
18-16 PSEG2	Phase Segment 2

Table continues on the next page...

Field	Function
	This 3-bit field defines the length of phase segment 2 in the bit time. The valid programmable values are 1–7. See Protocol timing for more information. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Phase Buffer Segment 2 = (PSEG2 + 1) × Time-Quanta.
15 BOFFMSK	Bus Off Interrupt Mask This bit provides a mask for the Bus Off interrupt ESR1[BOFFINT]. 0b - Bus Off interrupt disabled. 1b - Bus Off interrupt enabled.
14 ERRMSK	Error Interrupt Mask This bit provides a mask for the Error interrupt ESR1[ERRINT]. 0b - Error interrupt disabled. 1b - Error interrupt enabled.
13 —	Reserved **NOTE:** This field is writeable only if the module is disabled. Otherwise the access type is read-only.
12 LPB	Loop Back Mode This bit configures FlexCAN to operate in Loop-Back mode. In this mode, FlexCAN performs an internal loop back that can be used for self-test operation. The bit stream output of the transmitter is fed back internally to the receiver input. The Rx CAN input pin is ignored and the Tx CAN output goes to the recessive state (logic 1). FlexCAN behaves as it normally does when transmitting, and treats its own transmitted message as a message received from a remote node. In this mode, FlexCAN ignores the bit sent during the ACK slot in the CAN frame acknowledge field, generating an internal acknowledge bit to ensure proper reception of its own message. Both transmit and receive interrupts are generated. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. **NOTE:** In this mode, MCR[SRXDIS] cannot be asserted because this will impede the self-reception of a transmitted message. **NOTE:** FDCTRL[TDCEN] must be disabled when LPB is asserted. 0b - Loop Back disabled. 1b - Loop Back enabled.
11 TWRNMSK	Tx Warning Interrupt Mask This bit provides a mask for the Tx Warning interrupt associated with the TWRNINT flag in the Error and Status Register 1 (ESR1). This bit is read as zero when MCR[WRNEN] is negated. This bit can be written only if MCR[WRNEN] is asserted. 0b - Tx Warning interrupt disabled. 1b - Tx Warning interrupt enabled.
10 RWRNMSK	Rx Warning Interrupt Mask This bit provides a mask for the Rx Warning interrupt associated with the RWRNINT flag in the Error and Status Register 1 (ESR1). This bit is read as zero when MCR[WRNEN] bit is negated. This bit can be written only if MCR[WRNEN] bit is asserted. 0b - Rx Warning interrupt disabled. 1b - Rx Warning interrupt enabled.
9 —	Reserved
8 —	Reserved
7 SMP	CAN Bit Sampling This bit defines the sampling mode of CAN bits at the Rx input. It can be written in Freeze mode only, because it is blocked by hardware in other modes.

Table continues on the next page...

Field	Function
	NOTE: For proper operation, to assert SMP it is necessary to guarantee a minimum value of two TQs in CTRL1[PSEG1] (or CBT[EPSEG1]).This bit cannot be asserted when CAN FD is enabled (MCR[FDEN] = 1). 0b - Just one sample is used to determine the bit value. 1b - Three samples are used to determine the value of the received bit: the regular one (sample point) and two preceding samples; a majority rule is used.
6 BOFFREC	Bus Off Recovery This bit defines how FlexCAN recovers from Bus Off state. If this bit is negated, automatic recovering from Bus Off state occurs according to the CAN Specification 2.0B. If the bit is asserted, automatic recovering from Bus Off is disabled and the module remains in Bus Off state until the bit is negated by the user. If the negation occurs before 128 sequences of 11 recessive bits are detected on the CAN bus, then Bus Off recovery happens as if the BOFFREC bit had never been asserted. If the negation occurs after 128 sequences of 11 recessive bits occurred, then FlexCAN will resynchronize to the bus by waiting for 11 recessive bits before joining the bus. After negation, the BOFFREC bit can be reasserted again during Bus Off, but it will be effective only the next time the module enters Bus Off. If BOFFREC was negated when the module entered Bus Off, asserting it during Bus Off will not be effective for the current Bus Off recovery. **NOTE:** See Bus off in the CAN Protocol standard (ISO 11898-1) for details. 0b - Automatic recovering from Bus Off state enabled. 1b - Automatic recovering from Bus Off state disabled.
5 TSYN	Timer Sync This bit enables a mechanism that resets the free running timer each time a message is received in message buffer 0. This feature provides means to synchronize multiple FlexCAN stations with a special “SYNC” message, that is, global network time. If MCR[RFEN] is set (Rx FIFO enabled), the first available mailbox, according to CTRL2[RFFN] setting, is used for timer synchronization instead of MB0. This bit can be written in Freeze mode only because it is blocked by hardware in other modes. 0b - Timer sync feature disabled 1b - Timer sync feature enabled
4 LBUF	Lowest Buffer Transmitted First This bit defines the ordering mechanism for message buffer transmission. When asserted, MCR[LPRIOEN] does not affect the priority arbitration. This bit can be written in Freeze mode only, because it is blocked by hardware in other modes. 0b - Buffer with highest priority is transmitted first. 1b - Lowest number buffer is transmitted first.
3 LOM	Listen-Only Mode This bit configures FlexCAN to operate in Listen-Only mode. In this mode, transmission is disabled, all error counters described in the ECR register are frozen, and the module operates in a CAN Error Passive mode. Only messages acknowledged by another CAN station will be received. If FlexCAN detects a message that has not been acknowledged, it will flag a BIT0 error without changing the receive error counter (ECR[RXERRCNT]), as if it was trying to acknowledge the message. Listen-Only mode is acknowledged by the state of ESR1[FLTCONF] indicating Passive Error. There can be some delay between the Listen-Only mode request and acknowledge. This bit can be written in Freeze mode only because it is blocked by hardware in other modes. 0b - Listen-Only mode is deactivated. 1b - FlexCAN module operates in Listen-Only mode.
2-0 PROPSEG	Propagation Segment This 3-bit field defines the length of the propagation segment in the bit time. The valid programmable values are 0–7. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Propagation segment time = (PROPSEG + 1) × time-quanta.

Field	Function
	Time-quantum = one Sclock period.

45.6.2.4 Free Running Timer (TIMER)

45.6.2.4.1 Offset

Register	Offset
TIMER	8h

45.6.2.4.2 Function

This register represents a 16-bit free-running counter that can be read and written by the CPU. The timer starts from 0h after Reset, counts linearly to FFFFh, and wraps around.

When CTRL2[TIMER_SRC] is asserted, the timer is continuously incremented by an external time tick. The time tick must be synchronous to the peripheral clock, with a minimum pulse width of one clock cycle.

When CTRL2[TIMER_SRC] is negated, the timer is incremented by the CAN bit clock, which defines the baud rate on the CAN bus. During a message transmission/reception, it increments by one for each bit that is received or transmitted. When there is no message on the bus, it counts using the previously programmed baud rate. The timer is not incremented during Disable, Doze, Stop and Freeze modes.

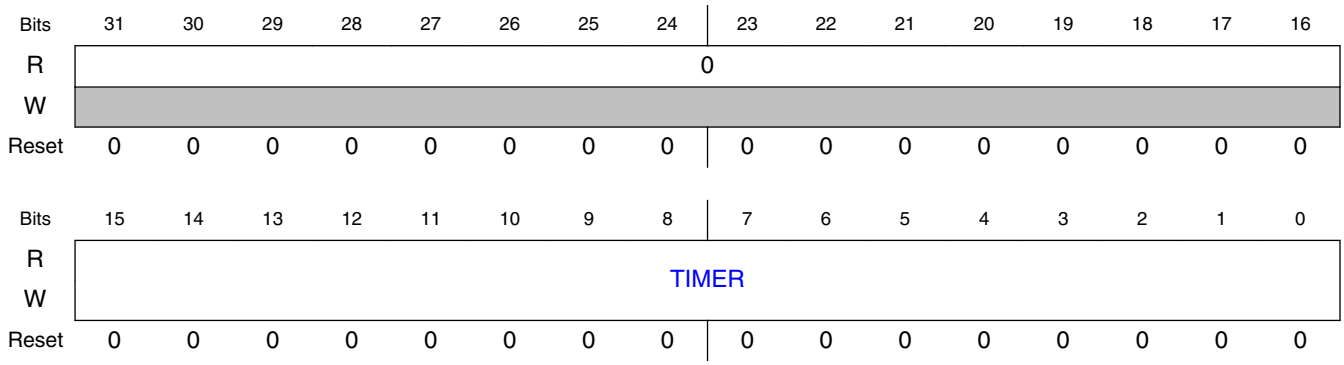
The timer value is captured when the second bit of the identifier field of any frame is on the CAN bus. This captured value is written into the time stamp entry in a message buffer after a successful reception or transmission of a message.

If CTRL1[TSYN] is asserted, the timer is reset whenever a message is received in the first available mailbox, according to CTRL2[RFFN] setting.

The CPU can write to this register anytime. However, if the write occurs at the same time that the timer is being reset by a reception in the first mailbox, then the write value is discarded.

Reading this register affects the mailbox unlocking procedure (see Mailbox lock mechanism).

45.6.2.4.3 Diagram



45.6.2.4.4 Fields

Field	Function
31-16 —	Reserved
15-0 TIMER	Timer Value Contains the free-running counter value.

45.6.2.5 Rx Mailboxes Global Mask Register (RXMGMASK)

45.6.2.5.1 Offset

Register	Offset
RXMGMASK	10h

45.6.2.5.2 Function

This register is located in RAM.

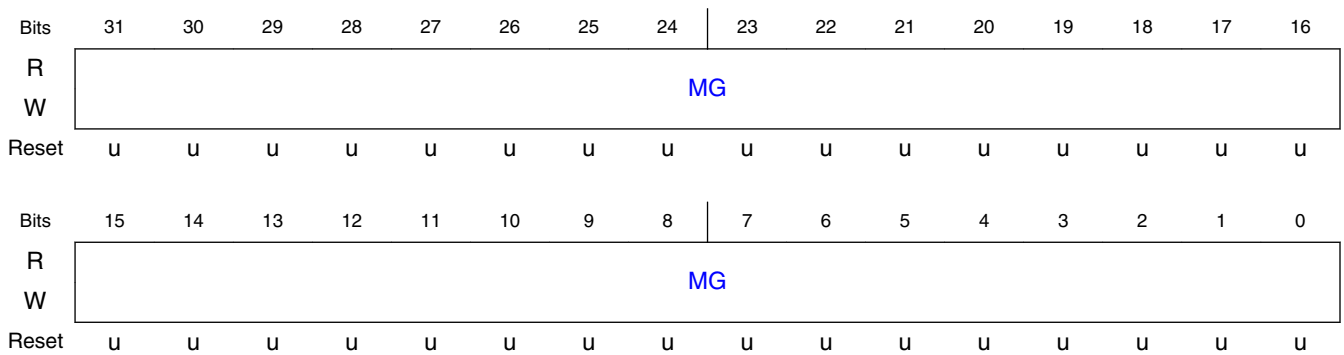
RXMGMASK is provided for legacy application support.

- When MCR[IRMQ] is negated, RXMGMASK is always in effect (the bits in the MG field will mask the mailbox filter bits).
- When MCR[IRMQ] is asserted, RXMGMASK has no effect (the bits in the MG field will not mask the mailbox filter bits).

RXMGMASK is used to mask the filter fields of all Rx MBs, excluding MBs 14-15, which have individual mask registers.

This register can only be written in Freeze mode as it is blocked by hardware in other modes.

45.6.2.5.3 Diagram



45.6.2.5.4 Fields

Field	Function
31-0 MG	Rx Mailboxes Global Mask Bits These bits mask the mailbox filter bits. Note that the alignment with the ID word of the mailbox is not perfect as the two most significant MG bits affect the fields RTR and IDE, which are located in the Control and Status word of the mailbox. The following table shows in detail which MG bits mask each mailbox filter field.

SMB[RTR] [-1]	CTRL2[RRS]	CTRL2[EACEN]	Mailbox filter fields			
			MB[RTR]	MB[IDE]	MB[ID]	Reserved
0	—	0	note [-1]	note [-1]	MG[28:0]	MG[31:29]
0	—	1	MG[31]	MG[30]	MG[28:0]	MG[29]
1	0	—	—	—	—	MG[31:0]
1	1	0	—	—	MG[28:0]	MG[31:29]
1	1	1	MG[31]	MG[30]	MG[28:0]	MG[29]

1. RTR bit of the incoming frame. It is saved into an auxiliary MB called Rx serial message buffer (Rx SMB).
2. If CTRL2[EACEN] is negated, the RTR bit of mailbox is never compared with the RTR bit of the incoming frame.
3. If CTRL2[EACEN] is negated, the IDE bit of mailbox is always compared with the IDE bit of the incoming frame.

0b - The corresponding bit in the filter is "don't care."
1b - The corresponding bit in the filter is checked.

1. RTR bit of the incoming frame. It is saved into an auxiliary MB called Rx serial message buffer (Rx SMB).

2. If CTRL2[EACEN] is negated, the RTR bit of mailbox is never compared with the RTR bit of the incoming frame.
3. If CTRL2[EACEN] is negated, the IDE bit of mailbox is always compared with the IDE bit of the incoming frame.

45.6.2.6 Rx 14 Mask Register (RX14MASK)

45.6.2.6.1 Offset

Register	Offset
RX14MASK	14h

45.6.2.6.2 Function

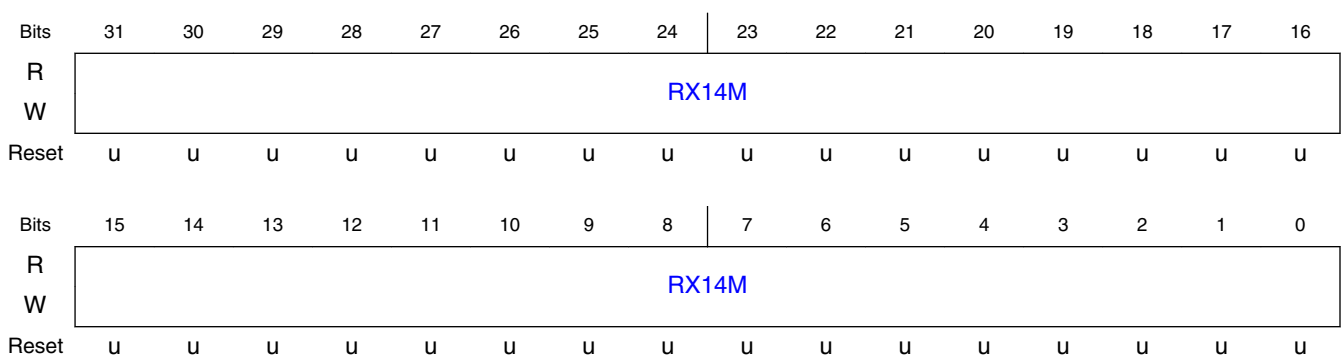
This register is located in RAM.

RX14MASK is provided for legacy application support. When MCR[IRMQ] is asserted, RX14MASK has no effect.

RX14MASK is used to mask the filter fields of message buffer 14.

This register can only be programmed when the module is in Freeze mode as it is blocked by hardware in other modes.

45.6.2.6.3 Diagram



45.6.2.6.4 Fields

Field	Function
31-0 RX14M	Rx Buffer 14 Mask Bits Each mask bit masks the corresponding mailbox 14 filter field in the same way that RXMGMASK masks other mailboxes' filters. See the description of the RXMGMASK register.

Field	Function
	0b - The corresponding bit in the filter is "don't care." 1b - The corresponding bit in the filter is checked.

45.6.2.7 Rx 15 Mask Register (RX15MASK)

45.6.2.7.1 Offset

Register	Offset
RX15MASK	18h

45.6.2.7.2 Function

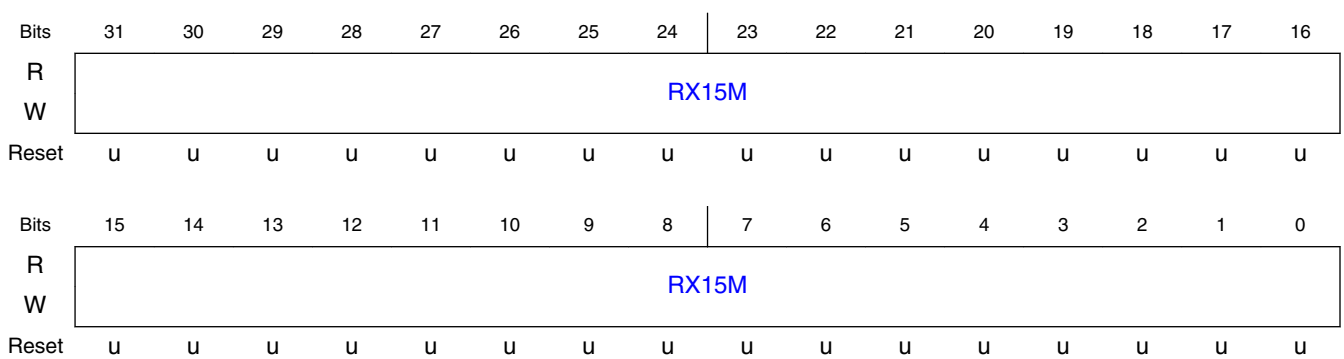
This register is located in RAM.

RX15MASK is provided for legacy application support. When MCR[IRMQ] is asserted, RX15MASK has no effect.

RX15MASK is used to mask the filter fields of message buffer 15.

This register can be programmed only when the module is in Freeze mode because it is blocked by hardware in other modes.

45.6.2.7.3 Diagram



45.6.2.7.4 Fields

Field	Function
31-0	Rx Buffer 15 Mask Bits

Field	Function
RX15M	Each mask bit masks the corresponding mailbox 15 filter field in the same way that RXMGMASK masks other mailboxes' filters. See the description of the RXMGMASK register. 0b - The corresponding bit in the filter is "don't care." 1b - The corresponding bit in the filter is checked.

45.6.2.8 Error Counter (ECR)

45.6.2.8.1 Offset

Register	Offset
ECR	1Ch

45.6.2.8.2 Function

This register has four 8-bit fields reflecting the value of the FlexCAN error counters:

- Transmit Error Counter (TXERRCNT field)
- Receive Error Counter (RXERRCNT field)
- Transmit Error Counter for errors detected in the data phase of CAN FD messages with the BRS bit set (TXERRCNT_FAST field)
- Receive Error Counter for errors detected in the data phase of CAN FD messages with the BRS bit set (RXERRCNT_FAST field)

The TXERRCNT and RXERRCNT counters take into account all errors in both CAN FD and non-FD message formats. TXERRCNT_FAST and RXERRCNT_FAST are dedicated to count only the errors that occur in the data phase of CAN FD frames with the BRS bit set.

The Fault Confinement state (ESR1[FLTCONF]) is updated based on TXERRCNT and RXERRCNT counters only. TXERRCNT and RXERRCNT counters can be written in Freeze mode only. TXERRCNT_FAST and RXERRCNT_FAST counters are read-only except in Freeze mode when the CPU can write value zero. The rules for increasing and decreasing these counters are described in the CAN protocol and are completely implemented in the FlexCAN module.

The following are the basic rules for FlexCAN bus state transitions:

- If the value of TXERRCNT or RXERRCNT becomes greater than or equal to 128, ESR1[FLTCONF] is updated to reflect Error Passive state.

- If the FlexCAN state is Error Passive, and either TXERRCNT or RXERRCNT decrements to a value less than or equal to 127 when the other already satisfies this condition, ESR1[FLTCONF] is updated to reflect Error Active state.
- If the value of TXERRCNT becomes greater than 255, ESR1[FLTCONF] is updated to reflect Bus Off state, and an interrupt may be issued. The value of TXERRCNT is then reset to zero.
- If FlexCAN is in Bus Off state, then TXERRCNT is cascaded together with another internal counter to count the 128th occurrences of 11 consecutive recessive bits on the bus. Hence, TXERRCNT is reset to zero and counts in a manner where the internal counter counts 11 such bits and then wraps around when incrementing the TXERRCNT. When TXERRCNT reaches the value of 128, ESR1[FLTCONF] is updated to be Error Active and both error counters are reset to zero. At any instance of dominant bit following a stream of less than 11 consecutive recessive bits, the internal counter resets itself to zero without affecting the TXERRCNT value. The TXERRCNT_FAST counter is frozen during Bus Off.
- If during system startup only one node is operating, then its TXERRCNT increases in each message it is trying to transmit, as a result of acknowledge errors (indicated by the ACKERR bit in the Error and Status Register). After the transition to Error Passive state, TXERRCNT does not increment anymore by acknowledge errors. Therefore the device never goes to the Bus Off state.
- If RXERRCNT increases to a value greater than 127, it is not incremented further, even if more errors are detected when being a receiver. At the next successful message reception, the counter is set to a value between 119 and 127 to return to Error Active state.
- TXERRCNT_FAST and RXERRCNT_FAST error counter values increment and decrement based on errors detected only in the data phase of CAN FD frames with the BRS bit set, following the same increment and decrement rules as TXERRCNT and RXERRCNT counters. These counters do not wrap around and get stuck at their maximum value (255). They stop counting and keep their values frozen when FlexCAN is in Bus Off state. They are reset when FlexCAN leaves Bus Off state and restart counting after FlexCAN returns to Error Active state.

NOTE

See Fault confinement in the CAN Protocol standard (ISO 11898-1) for details.

45.6.2.8.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	RXERRCNT_FAST								TXERRCNT_FAST							
W	0								0							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RXERRCNT								TXERRCNT							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

45.6.2.8.4 Fields

Field	Function
31-24 RXERRCNT_FAST	Receive Error Counter for fast bits Receive error counter for errors detected in the data phase of received CAN FD messages with the BRS bit set. The RXERRCNT_FAST counter is read-only except in Freeze mode, where the CPU can write an 8-bit zero value only.
23-16 TXERRCNT_FAST	Transmit Error Counter for fast bits Transmit error counter for errors detected in the data phase of transmitted CAN FD messages with the BRS bit set. The TXERRCNT_FAST counter is read-only except in Freeze mode, where the CPU can write an 8-bit zero value only.
15-8 RXERRCNT	Receive Error Counter Receive error counter for all errors detected in received messages. The RXERRCNT counter is read-only except in Freeze mode, where it can be written by the CPU.
7-0 TXERRCNT	Transmit Error Counter Transmit error counter for all errors detected in transmitted messages. The TXERRCNT counter is read-only except in Freeze mode, where it can be written by the CPU.

45.6.2.9 Error and Status 1 Register (ESR1)

45.6.2.9.1 Offset

Register	Offset
ESR1	20h

45.6.2.9.2 Function

This register reports various error conditions detected in the reception and transmission of a CAN frame, some general status of the device, and is the source of some interrupts to the CPU. The reported error conditions are:

NOTE

Reading Host can clear these field.

- Errors detected in CAN frames of any format:
  - BIT1ERR
  - BIT0ERR
  - ACKKERR
  - CRCERR
  - FRMERR
  - STFERR
- Errors detected in the data phase of CAN FD frames with the BRS bit set only:
  - BIT1ERR_FAST
  - BIT0ERR_FAST
  - CRCERR_FAST
  - FRMERR_FAST
  - STFERR_FAST

An error detected in a single CAN frame may be reported by one or more error flags. Also, error reporting is cumulative, in case more error events happen in the next frames when the CPU does not attempt to read this register.

Status bits:
- TXWRN
- RXWRN
- IDLE
- TX
- FLTCONF
- RX
- SYNCH

Interrupt bits:
- BOFFINT
- BOFFDONEINT
- ERRINT
- ERRINT_FAST
- WAKINT
- TWRNINT
- RWRNINT

It is recommended that the CPU use the following procedure when servicing interrupt requests generated by these bits:

1. Read this register to capture all error condition and status bits. This action clears the respective bits that were set since the last read access.
2. Write 1 to clear the interrupt bit that has triggered the interrupt request.
3. Write 1 to clear the ERR_OVR bit, if it is set.

Starting from all error flags cleared, a first error event sets either the ERRINT or the ERRINT_FAST (provided the corresponding mask bit is asserted). If other error events in subsequent frames happen before the CPU serves the interrupt request, the ERR_OVR bit is set to indicate that errors from different frames have accumulated.

Table 45-22. Can bus status

SYNCH	IDLE	TX	RX	FlexCAN state
0	0	0	0	Not synchronized to CAN bus
1	1	x	x	Idle
1	0	1	0	Transmitting
1	0	0	1	Receiving

NOTE

See Fault confinement in the CAN Protocol standard (ISO 11898-1) for details.

45.6.2.9.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BIT1ERR_FAST	BIT0ERR_FAST	0	CRCERR_FAST	FRMERR_FAST	STFERR_FAST	0				ERROVR	ERRINT_FAST	BOFFDONEINT	SYNCH	TWRNINT	RWRNINT
W											W1C	W1C	W1C		W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BIT1ER	BIT0ER	ACKERR	CRCER	FRMER	STFER	TXWRN	RXWRN	IDLE	TX	FLTCONF		RX	BOFFINT	ERRINT	WAKINT
W														W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

45.6.2.9.4 Fields

Field	Function
31 BIT1ERR_FAST	Bit1 Error in the Data Phase of CAN FD frames with the BRS bit set This bit indicates when an inconsistency occurs between the transmitted and the received bit in the data phase of CAN FD frames with the BRS bit set. After a read operation, the field's value clears to 0. 0b - No such occurrence. 1b - At least one bit sent as recessive is received as dominant.
30 BIT0ERR_FAST	Bit0 Error in the Data Phase of CAN FD frames with the BRS bit set This bit indicates when an inconsistency occurs between the transmitted and the received bit in the data phase of CAN FD frames with the BRS bit set. After a read operation, the field's value clears to 0. 0b - No such occurrence. 1b - At least one bit sent as dominant is received as recessive.
29 —	Reserved
28 CRCERR_FAST	Cyclic Redundancy Check Error in the CRC field of CAN FD frames with the BRS bit set This bit indicates that a CRC error has been detected by the receiver node in the CRC field of CAN FD frames with the BRS bit set, that is, the calculated CRC is different from the received. After a read operation, the field's value clears to 0. 0b - No such occurrence. 1b - A CRC error occurred since last read of this register.
27 FRMERR_FAST	Form Error in the Data Phase of CAN FD frames with the BRS bit set This bit indicates that a form error has been detected by the receiver node in the data phase of CAN FD frames with the BRS bit set, that is, a fixed-form bit field contains at least one illegal bit. After a read operation, the field's value clears to 0. 0b - No such occurrence. 1b - A form error occurred since last read of this register.
26 STFERR_FAST	Stuffing Error in the Data Phase of CAN FD frames with the BRS bit set This bit indicates that a stuffing error has been detected in the data phase of CAN FD frames with the BRS bit set. After a read operation, the field's value clears to 0. 0b - No such occurrence. 1b - A stuffing error occurred since last read of this register.
25-22 —	Reserved
21 ERROVR	Error Overrun This bit indicates that an error condition occurred when any error flag is already set. This bit is cleared by writing it to 1. 0b - Overrun has not occurred. 1b - Overrun has occurred.
20	Error interrupt for errors detected in Data Phase of CAN FD frames with BRS bit set

Table continues on the next page...

Field	Function
ERRINT_FAST	This bit indicates that at least one of the error bits detected in the data phase of CAN FD frames with the BRS bit set (BIT1ERR_FAST, BIT0ERR_FAST, CRCERR_FAST, FRMERR_FAST, or STFERR_FAST) is set. If the corresponding mask bit CTRL2[ERRMSK_FAST] is set, an interrupt is generated to the CPU. This bit is cleared by writing it to 1. Writing 0 has no effect. 0b - No such occurrence. 1b - Indicates setting of any error bit detected in the data phase of CAN FD frames with the BRS bit set.
19 BOFFDONEINT	Bus Off Done Interrupt This bit is set when the Tx Error Counter (TXERRCNT) has finished counting 128 occurrences of 11 consecutive recessive bits on the CAN bus and is ready to leave Bus Off. If the corresponding mask bit in the Control 2 Register (BOFFDONEMSK) is set, an interrupt is generated to the CPU. This bit is cleared by writing it to 1. Writing 0 has no effect. 0b - No such occurrence. 1b - FlexCAN module has completed Bus Off process.
18 SYNCH	CAN Synchronization Status This read-only flag indicates whether the FlexCAN is synchronized to the CAN bus and able to participate in the communication process. It is set and cleared by the FlexCAN. See the table in the overall ESR1 register description. 0b - FlexCAN is not synchronized to the CAN bus. 1b - FlexCAN is synchronized to the CAN bus.
17 TWRNINT	Tx Warning Interrupt Flag If MCR[WRNEN] is asserted, TWRNINT is set when the TXWRN flag transitions from 0 to 1, meaning that the Tx error counter reached 96. If the corresponding mask bit in the Control 1 Register (CTRL1[TWRNMSK]) is set, an interrupt is generated to the CPU. This bit is cleared by writing it to 1. When MCR[WRNEN] is negated, this flag is masked. CPU must clear this flag before disabling MCR[WRNEN]. Otherwise it will be set when MCR[WRNEN] is set again. Writing 0 has no effect. This flag is not generated during Bus Off state. This bit is not updated during Freeze mode. 0b - No such occurrence. 1b - The Tx error counter transitioned from less than 96 to greater than or equal to 96.
16 RWRNINT	Rx Warning Interrupt Flag If MCR[WRNEN] is asserted, RWRNINT is set when the RXWRN flag transitions from 0 to 1, meaning that the Rx error counters reached 96. If the corresponding mask bit in the Control 1 Register (CTRL1[RWRNMSK]) is set, an interrupt is generated to the CPU. This bit is cleared by writing it to 1. When WRNEN is negated, this flag is masked. CPU must clear this flag before disabling the bit. Otherwise it will be set when WRNEN is set again. Writing 0 has no effect. This bit is not updated during Freeze mode. 0b - No such occurrence. 1b - The Rx error counter transitioned from less than 96 to greater than or equal to 96.
15 BIT1ERR	Bit1 Error This bit indicates when an inconsistency occurs between the transmitted and the received bit in a non-CAN FD message or in the arbitration or data phase of a CAN FD message. **NOTE:** This bit is not set by a transmitter in case of arbitration field or ACK slot, or in case of a node sending a passive error flag that detects dominant bits. After a read operation, the field's value clears to 0. 0b - No such occurrence. 1b - At least one bit sent as recessive is received as dominant.
14 BIT0ERR	Bit0 Error This bit indicates when an inconsistency occurs between the transmitted and the received bit in a non-CAN FD message or in the arbitration or data phase of a CAN FD message.

Table continues on the next page...

Field	Function
	After a read operation, the field's value clears to 0. 0b - No such occurrence. 1b - At least one bit sent as dominant is received as recessive.
13 ACKERR	Acknowledge Error This bit indicates that an Acknowledge Error has been detected by the transmitter node, that is, a dominant bit has not been detected during the ACK SLOT. After a read operation, the field's value clears to 0. 0b - No such occurrence. 1b - An ACK error occurred since last read of this register.
12 CRCERR	Cyclic Redundancy Check Error This bit indicates that a Cyclic Redundancy Check (CRC) error has been detected by the receiver node either in a non-FD message or in the arbitration or data phase of a frame in CAN FD format, that is, the calculated CRC is different from the received. After a read operation, the field's value clears to 0. 0b - No such occurrence. 1b - A CRC error occurred since last read of this register.
11 FRMERR	Form Error This bit indicates that a form error has been detected in a non-FD message or else in an FD message's arbitration or data phase by the receiver node, that is, a fixed-form bit field contains at least one illegal bit. After a read operation, the field's value clears to 0. 0b - No such occurrence. 1b - A Form Error occurred since last read of this register.
10 STFERR	Stuffing Error This bit indicates that a stuffing error has been detected in a non-FD message or else in an FD message's arbitration or data phase by the receiver node. After a read operation, the field's value clears to 0. 0b - No such occurrence. 1b - A stuffing error occurred since last read of this register.
9 TXWRN	TX Error Warning This bit indicates when repetitive errors are occurring during message transmission and is affected by the value of TXERRCNT in ECR register only. This bit is not updated during Freeze mode. After a read operation, the field's value clears to 0. 0b - No such occurrence. 1b - TXERRCNT is greater than or equal to 96.
8 RXWRN	Rx Error Warning This bit indicates when repetitive errors are occurring during message reception and is affected by the value of RXERRCNT in ECR register only. This bit is not updated during Freeze mode. After a read operation, the field's value clears to 0. 0b - No such occurrence. 1b - RXERRCNT is greater than or equal to 96.
7 IDLE	IDLE This bit indicates when CAN bus is in IDLE state. See the table in the overall ESR1 register description. 0b - No such occurrence. 1b - CAN bus is now IDLE.

Table continues on the next page...

Field	Function
6 TX	FlexCAN In Transmission This bit indicates if FlexCAN is transmitting a message. See the table in the overall ESR1 register description. 0b - FlexCAN is not transmitting a message. 1b - FlexCAN is transmitting a message.
5-4 FLTCONF	Fault Confinement State This 2-bit field indicates the confinement state of the FlexCAN module. If CTRL1[LOM] is asserted, after a delay that depends on the CAN bit timing, ESR1[FLTCONF] will indicate Error Passive. The very same delay affects the way that ESR1[FLTCONF] reflects an update to ECR register by the CPU. It may be necessary to wait up to one CAN bit time to get them coherent again. This bit field is affected by soft reset, but if the LOM bit is asserted, its reset value lasts just one CAN bit. After this time, ESR1[FLTCONF] reports Error Passive. 00b - Error Active 01b - Error Passive 1xb - Bus Off
3 RX	FlexCAN In Reception This bit indicates if FlexCAN is receiving a message. See the table in the overall ESR1 register description. 0b - FlexCAN is not receiving a message. 1b - FlexCAN is receiving a message.
2 BOFFINT	Bus Off Interrupt This bit is set when FlexCAN enters Bus Off state. If the corresponding mask bit in the Control Register 1 (CTRL1[BOFFMSK]) is set, an interrupt is generated to the CPU. This bit is cleared by writing it to 1. Writing 0 has no effect. 0b - No such occurrence. 1b - FlexCAN module entered Bus Off state.
1 ERRINT	Error Interrupt This bit indicates that at least one of the error bits (BIT1ERR, BIT0ERR, ACKERR, CRCERR, FRMERR, or STFERR) is set. If the corresponding mask bit CTRL1[ERRMSK] is set, an interrupt is generated to the CPU. This bit is cleared by writing it to 1. Writing 0 has no effect. 0b - No such occurrence. 1b - Indicates setting of any error bit in the Error and Status register.
0 WAKINT	Wake-Up Interrupt This field applies when FlexCAN is in low-power mode under Self Wake Up mechanism: • Doze mode • Stop mode When a recessive-to-dominant transition is detected on the CAN bus and if MCR[WAKMSK] is set, an interrupt is generated to the CPU. This bit is cleared by writing it to 1. When MCR[SLFWAK] is negated, this flag is masked. The CPU must clear this flag before disabling the bit. Otherwise it will be set when SLFWAK is set again. Writing 0 has no effect. 0b - No such occurrence. 1b - Indicates a recessive to dominant transition was received on the CAN bus.

45.6.2.10 Interrupt Masks 2 Register (IMASK2)

45.6.2.10.1 Offset

Register	Offset
IMASK2	24h

45.6.2.10.2 Function

This register allows any number of a range of the 32 message buffer interrupts to be enabled or disabled for MB63 to MB32. It contains one interrupt mask bit per buffer, enabling the CPU to determine which buffer generates an interrupt after a successful transmission or reception, that is, when the corresponding IFLAG2 bit is set.

45.6.2.10.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BUF63TO32M															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BUF63TO32M															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

45.6.2.10.4 Fields

Field	Function
31-0 BUF63TO32M	Buffer MBi Mask Each bit enables or disables the corresponding FlexCAN message buffer interrupt for MB63 to MB32. **NOTE:** Setting or clearing a bit in the IMASK2 Register can assert or negate an interrupt request, if the corresponding IFLAG2 bit is set. 0b - The corresponding buffer interrupt is disabled. 1b - The corresponding buffer interrupt is enabled.

45.6.2.11 Interrupt Masks 1 Register (IMASK1)

45.6.2.11.1 Offset

Register	Offset
IMASK1	28h

45.6.2.11.2 Function

This register allows any number of a range of the 32 message buffer interrupts to be enabled or disabled for MB31 to MB0. It contains one interrupt mask bit per buffer, enabling the CPU to determine which buffer generates an interrupt after a successful transmission or reception, that is, when the corresponding IFLAG1 bit is set.

45.6.2.11.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BUF31TO0M															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	BUF31TO0M															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

45.6.2.11.4 Fields

Field	Function
31-0 BUF31TO0M	Buffer MBi Mask Each bit enables or disables the corresponding FlexCAN message buffer interrupt for MB31 to MB0. **NOTE:** Setting or clearing a bit in the IMASK1 Register can assert or negate an interrupt request, if the corresponding IFLAG1 bit is set. 0b - The corresponding buffer interrupt is disabled. 1b - The corresponding buffer interrupt is enabled.

45.6.2.12 Interrupt Flags 2 Register (IFLAG2)

45.6.2.12.1 Offset

Register	Offset
IFLAG2	2Ch

45.6.2.12.2 Function

This register defines the flags for the 32 message buffer interrupts for MB63 to MB32. It contains one interrupt flag bit per buffer. Each successful transmission or reception sets the respective IFLAG2 bit. If the corresponding IMASK2 bit is set, an interrupt will be generated. The interrupt flag must be cleared by writing 1 to it. Writing 0 has no effect.

Before updating MCR[MAXMB], CPU must service the IFLAG2 bits whose MB value is greater than the MAXMB to be updated; otherwise, they will remain set and be inconsistent with the number of MBs available.

45.6.2.12.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	BUF63TO32I															
W	W1C															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	BUF63TO32I															
W	W1C															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

45.6.2.12.4 Fields

Field	Function
31-0 BUF63TO32I	Buffer MBi Interrupt Each bit flags the corresponding FlexCAN message buffer interrupt for MB63 to MB32. 0b - The corresponding buffer has no occurrence of successfully completed transmission or reception. 1b - The corresponding buffer has successfully completed transmission or reception.

45.6.2.13 Interrupt Flags 1 Register (IFLAG1)

45.6.2.13.1 Offset

Register	Offset
IFLAG1	30h

45.6.2.13.2 Function

This register defines the flags for the 32 message buffer interrupts for MB31 to MB0. It contains one interrupt flag bit per buffer. Each successful transmission or reception sets the corresponding IFLAG1 bit. If the corresponding IMASK1 bit is set, an interrupt will be generated. The interrupt flag must be cleared by writing 1 to it. Writing 0 has no effect. There is an exception when DMA for Rx FIFO is enabled, as described below.

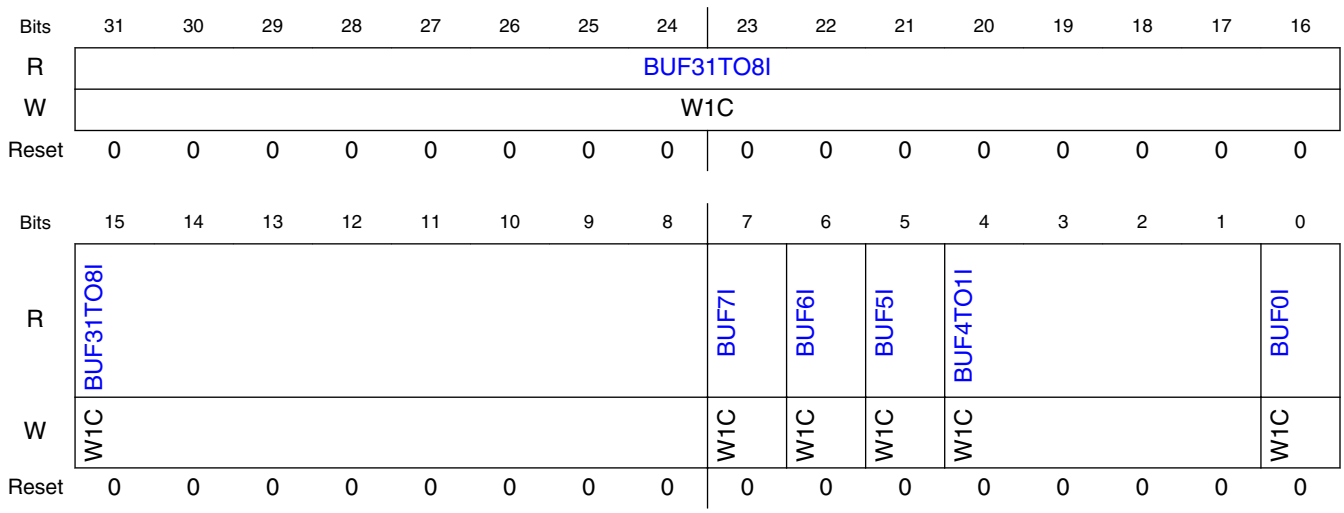
The BUF7I to BUF5I flags are also used to represent FIFO interrupts when the Rx FIFO is enabled. When MCR[RFEN] is set and MCR[DMA] is negated, the function of the eight least significant interrupt flags changes: BUF7I, BUF6I, and BUF5I indicate operating conditions of the FIFO , BUF0I is used to empty FIFO, and BUF4I to BUF1I bits are reserved.

Before enabling MCR[RFEN], the CPU must service the IFLAG bits asserted in the Rx FIFO region; see Rx FIFO. Otherwise, these IFLAG bits will mistakenly show the related MBs now belonging to FIFO as having contents to be serviced. When MCR[RFEN] is negated, the FIFO flags must be cleared. The same care must be taken when a CTRL2[RFFN] value is selected, extending Rx FIFO filters beyond MB7. For example, when RFFN is 8h, the MB0–23 range is occupied by Rx FIFO filters and related IFLAG bits must be cleared.

When both the MCR[RFEN] and MCR[DMA] bits are asserted (DMA feature for Rx FIFO enabled), the function of the eight least significant interrupt flags (BUF7I–BUF0I) are changed to support the DMA operation. BUF7I and BUF6I are not used, as well as BUF4I–BUF1I. BUF5I indicates operating condition of FIFO, and BUF0I is used to empty FIFO. Moreover, BUF5I does not generate a CPU interrupt, but generates a DMA request. IMASK1 bits in Rx FIFO region are not considered when bit MCR[DMA] is enabled. In addition the CPU must not clear the flag BUF5I when DMA is enabled. Before enabling MCR[DMA], the CPU must service the IFLAGs asserted in the Rx FIFO region. When MCR[DMA] is negated, the FIFO must be empty. FIFO must be disabled when MCR[FDEN] is enabled.

Before updating MCR[MAXMB], CPU must service the IFLAG1 bits whose MB value is greater than the MCR[MAXMB] to be updated; otherwise, they will remain set and be inconsistent with the number of MBs available.

45.6.2.13.3 Diagram



45.6.2.13.4 Fields

Field	Function
31-8 BUF31TO8I	Buffer MBi Interrupt Each bit flags the corresponding FlexCAN message buffer interrupt for MB31 to MB8. 0b - The corresponding buffer has no occurrence of successfully completed transmission or reception. 1b - The corresponding buffer has successfully completed transmission or reception.
7 BUF7I	Buffer MB7 Interrupt Or Rx FIFO Overflow When MCR[RFEN] is cleared (Rx FIFO disabled), this bit flags the interrupt for MB7. **NOTE:** This flag is cleared by the FlexCAN whenever MCR[RFEN] is changed by CPU writes. The BUF7I flag represents Rx FIFO overflow when MCR[RFEN] is set. In this case, the flag indicates that a message was lost because the Rx FIFO is full. Note that the flag will not be asserted when the Rx FIFO is full and the message was captured by a mailbox. 0b - No occurrence of MB7 completing transmission/reception when MCR[RFEN]=0, or of Rx FIFO overflow when MCR[RFEN]=1 1b - MB7 completed transmission/reception when MCR[RFEN]=0, or Rx FIFO overflow when MCR[RFEN]=1
6 BUF6I	Buffer MB6 Interrupt Or Rx FIFO Warning When MCR[RFEN] is cleared (Rx FIFO disabled), this bit flags the interrupt for MB6. **NOTE:** This flag is cleared by the FlexCAN whenever MCR[RFEN] is changed by CPU writes.

Table continues on the next page...

Field	Function
	The BUF6I flag represents Rx FIFO warning when MCR[RFEN] is set. In this case, the flag indicates when the number of unread messages within the Rx FIFO is increased to five from four due to the reception of a new one, meaning that the Rx FIFO is almost full. Note that if the flag is cleared when the number of unread messages is greater than four, it does not assert again until the number of unread messages within the Rx FIFO is decreased to be equal to or less than four. 0b - No occurrence of MB6 completing transmission/reception when MCR[RFEN]=0, or of Rx FIFO almost full when MCR[RFEN]=1 1b - MB6 completed transmission/reception when MCR[RFEN]=0, or Rx FIFO almost full when MCR[RFEN]=1
5 BUF5I	Buffer MB5 Interrupt Or Frames available in Rx FIFO When MCR[RFEN] is cleared (Rx FIFO disabled), this field flags the interrupt for MB5. **NOTE:** This flag is cleared by the FlexCAN whenever MCR[RFEN] is changed by CPU writes. When MCR[RFEN] is set (Rx FIFO enabled), the BUF5I flag represents "Frames available in Rx FIFO" and indicates that at least one frame is available to be read from the Rx FIFO. When the MCR[DMA] bit is enabled, this flag generates a DMA request and the CPU must not clear this bit by writing 1 in BUF5I. 0b - No occurrence of MB5 completing transmission/reception when MCR[RFEN]=0, or of frame(s) available in the FIFO, when MCR[RFEN]=1 1b - MB5 completed transmission/reception when MCR[RFEN]=0, or frame(s) available in the Rx FIFO when MCR[RFEN]=1. It generates a DMA request in case of MCR[RFEN] and MCR[DMA] are enabled.
4-1 BUF4TO1I	Buffer MBi Interrupt Or Reserved When MCR[RFEN] is cleared (Rx FIFO disabled), these bits flag the interrupts for MB4 to MB1. **NOTE:** These flags are cleared by the FlexCAN whenever MCR[RFEN] is changed by CPU writes. The BUF4TO1I flags are reserved when MCR[RFEN] is set. 0b - The corresponding buffer has no occurrence of successfully completed transmission or reception when MCR[RFEN]=0. 1b - The corresponding buffer has successfully completed transmission or reception when MCR[RFEN]=0.
0 BUF0I	Buffer MB0 Interrupt Or Clear FIFO bit When MCR[RFEN] is cleared (Rx FIFO disabled), this field flags the interrupt for MB0. If the Rx FIFO is enabled, this bit is used to trigger the clear FIFO operation. This operation empties FIFO contents. Before performing this operation the CPU must service all FIFO related IFLAGs. When MCR[DMA] is enabled this operation also clears the BUF5I flag and consequently aborts the DMA request. The clear FIFO operation occurs when the CPU writes 1 in BUF0I. It is only allowed in Freeze mode and is blocked by hardware in other conditions. 0b - The corresponding buffer has no occurrence of successfully completed transmission or reception when MCR[RFEN]=0. 1b - The corresponding buffer has successfully completed transmission or reception when MCR[RFEN]=0.

45.6.2.14 Control 2 Register (CTRL2)

45.6.2.14.1 Offset

Register	Offset
CTRL2	34h

45.6.2.14.2 Function

This register complements Control1 register, providing control bits for memory write access in Freeze mode, for extending FIFO filter quantity, and to adjust the operation of internal FlexCAN processes like matching and arbitration.

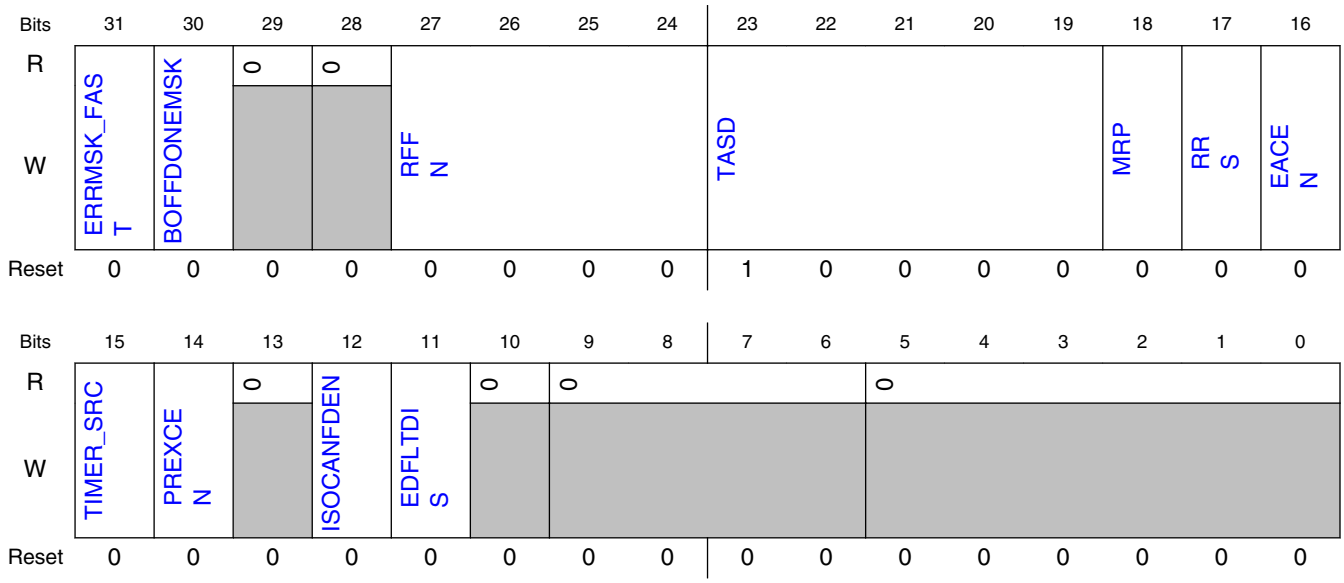
The contents of this register are not affected by soft reset.

This table shows how the Rx FIFO filter structure is determined by the value of CTRL2[RFFN]. See the CTRL2[RFFN] field description for more information.

Table 45-23. Rx FIFO filter: possible structures

RFFN[3:0]	Number of Rx FIFO filter elements	Message buffers occupied by Rx FIFO and ID filter table	Remaining available mailboxes	Rx FIFO ID filter table elements affected by Rx individual masks	Rx FIFO ID filter table elements affected by Rx FIFO global mask
0h	8	MB 0-7	MB 8-63	Elements 0-7	none
1h	16	MB 0-9	MB 10-63	Elements 0-9	Elements 10-15
2h	24	MB 0-11	MB 12-63	Elements 0-11	Elements 12-23
3h	32	MB 0-13	MB 14-63	Elements 0-13	Elements 14-31
4h	40	MB 0-15	MB 16-63	Elements 0-15	Elements 16-39
5h	48	MB 0-17	MB 18-63	Elements 0-17	Elements 18-47
6h	56	MB 0-19	MB 20-63	Elements 0-19	Elements 20-55
7h	64	MB 0-21	MB 22-63	Elements 0-21	Elements 22-63
8h	72	MB 0-23	MB 24-63	Elements 0-23	Elements 24-71
9h	80	MB 0-25	MB 26-63	Elements 0-25	Elements 26-79
Ah	88	MB 0-27	MB 28-63	Elements 0-27	Elements 28-87
Bh	96	MB 0-29	MB 30-63	Elements 0-29	Elements 30-95
Ch	104	MB 0-31	MB 32-63	Elements 0-31	Elements 32-103
Dh	112	MB 0-33	MB 34-63	Elements 0-31	Elements 32-111
Eh	120	MB 0-35	MB 36-63	Elements 0-31	Elements 32-119
Fh	128	MB 0-37	MB 38-63	Elements 0-31	Elements 32-127

45.6.2.14.3 Diagram



45.6.2.14.4 Fields

Field	Function
31 ERRMSK_FAST	Error Interrupt Mask for errors detected in the data phase of fast CAN FD frames This bit provides a mask for the ERRINT_FAST interrupt in ESR1 register. 0b - ERRINT_FAST error interrupt disabled. 1b - ERRINT_FAST error interrupt enabled.
30 BOFFDONEMSK	Bus Off Done Interrupt Mask This bit provides a mask for the bus off done interrupt in ESR1 register. 0b - Bus off done interrupt disabled. 1b - Bus off done interrupt enabled.
29 —	Reserved
28 —	Reserved
27-24 RFFN	Number Of Rx FIFO Filters This 4-bit field defines the number of Rx FIFO filters, as shown in Table 45-23. The maximum selectable number of filters is determined by the chip. This field can only be written in Freeze mode as it is blocked by hardware in other modes. This field must not be programmed with values that cause the number of message buffers occupied by Rx FIFO and ID Filter to exceed the number of mailboxes present, as defined by MCR[MAXMB]. **NOTE:** Each group of eight filters occupies a memory space equivalent to two message buffers, which means that the more filters are implemented the fewer mailboxes will be available. Considering that the Rx FIFO occupies the memory space originally reserved for MB0-5, RFFN should be programmed with a value corresponding to a number of filters not greater than the number of available memory words, which can be calculated as follows: (SETUP_MB - 6) × 4

Table continues on the next page...

Field	Function
	where SETUP_MB is the smaller of the parameter NUMBER_OF_MB and MCR[MAXMB]. The number of remaining mailboxes available will be: (SETUP_MB - 8) - (RFFN × 2) If the number of Rx FIFO filters programmed through RFFN exceeds the SETUP_MB value (memory space available), then the exceeding ones will not be functional. **NOTE:** • The number of the last remaining available mailboxes is defined by the smaller of NUMBER_OF_MB minus 1 and MCR[MAXMB]. • If Rx Individual Mask registers are not enabled then all Rx FIFO filters are affected by the Rx FIFO Global Mask.
23-19 TASD	Tx Arbitration Start Delay This 5-bit field indicates how many CAN bits the Tx arbitration process start point can be delayed from the first bit of CRC field on CAN bus. See Tx arbitration start delay for more details. This field can be written only in Freeze mode because it is blocked by hardware in other modes.
18 MRP	Mailboxes Reception Priority If this bit is set the matching process starts from the mailboxes and if no match occurs the matching continues on the Rx FIFO. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. 0b - Matching starts from Rx FIFO and continues on mailboxes. 1b - Matching starts from mailboxes and continues on Rx FIFO.
17 RRS	Remote Request Storing If this bit is asserted a remote request frame is submitted to a matching process and stored in the corresponding message buffer in the same fashion as a data frame. No automatic remote response frame will be generated. If this bit is negated the remote request frame is submitted to a matching process and an automatic remote response frame is generated if a message buffer with CODE = 1010b is found with the same ID. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. 0b - Remote response frame is generated. 1b - Remote request frame is stored.
16 EACEN	Entire Frame Arbitration Field Comparison Enable For Rx Mailboxes This bit controls the comparison of IDE and RTR bits within Rx mailbox filters with their corresponding bits in the incoming frame by the matching process. This bit does not affect matching for Rx FIFO. This bit can be written only in Freeze mode because it is blocked by hardware in other modes. 0b - Rx mailbox filter's IDE bit is always compared and RTR is never compared despite mask bits. 1b - Enables the comparison of both Rx mailbox filter's IDE and RTR bit with their corresponding bits within the incoming frame. Mask bits do apply.
15 TIMER_SRC	Timer Source Selects the time tick source used for incrementing the free running timer counter. This bit can be written in Freeze mode only. 0b - The free running timer is clocked by the CAN bit clock, which defines the baud rate on the CAN bus. 1b - The free running timer is clocked by an external time tick. The period can be either adjusted to be equal to the baud rate on the CAN bus, or a different value as required. See the device-specific section for details about the external time tick.
14 PREXCEN	Protocol Exception Enable This bit enables the protocol exception feature. This field is writable only in Freeze mode. **NOTE:** See Protocol exception event in the CAN Protocol standard (ISO 11898-1) for details.

Table continues on the next page...

Field	Function
	0b - Protocol exception is disabled. 1b - Protocol exception is enabled.
13 —	Reserved
12 ISOCANFDEN	ISO CAN FD Enable This field enables the CAN FD protocol according to ISO specification (ISO 11898-1) (see CAN FD ISO compliance). This field is writable only in Freeze mode. **NOTE:** FlexCAN is able to transmit FD frame format according to CAN Protocol standard (ISO 11898-1). 0b - FlexCAN operates using the non-ISO CAN FD protocol. 1b - FlexCAN operates using the ISO CAN FD protocol (ISO 11898-1).
11 EDFLTDIS	Edge Filter Disable This bit disables the edge filter used during the bus integration state. When the Edge Filter is enabled, two consecutive nominal time quanta with dominant bus state are required to detect an edge that causes synchronization. When synchronization occurs, the counting of the sequence of eleven consecutive recessive bits is restarted. The edge filter prevents the dominant pulses that are shorter than a nominal bit time (present during the data phase of an FD frame) from being mistaken for an idle condition. This field is writable only in Freeze mode. **NOTE:** See Bus Integration state in the CAN Protocol standard (ISO 11898-1) for details. 0b - Edge filter is enabled 1b - Edge filter is disabled
10 —	Reserved
9-6 —	Reserved
5-0 —	Reserved

45.6.2.15 Error and Status 2 Register (ESR2)

45.6.2.15.1 Offset

Register	Offset
ESR2	38h

45.6.2.15.2 Function

This register reports some general status information.

45.6.2.15.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0									LPTM						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	VPS	IMB	0												
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

45.6.2.15.4 Fields

Field	Function
31-23 —	Reserved
22-16 LPTM	Lowest Priority Tx Mailbox If ESR2[VPS] is asserted, this field indicates the lowest number inactive mailbox (see the ESR2[IMB] bit description). If there is no inactive mailbox then the mailbox indicated depends on the value of CTRL1[LBUF]. If CTRL1[LBUF] is negated, then the mailbox indicated is the one that has the greatest arbitration value (see Highest-priority mailbox first). If CTRL1[LBUF] is asserted then the mailbox indicated is the highest number active Tx mailbox. If a Tx mailbox is being transmitted it is not considered in LPTM calculation. If ESR2[IMB] is not asserted and a frame is transmitted successfully, LPTM is updated with its mailbox number.
15 —	Reserved
14 VPS	Valid Priority Status This bit indicates whether ESR2[IMB] and ESR2[LPTM] contents are currently valid or not. It is asserted upon every complete Tx arbitration process unless the CPU writes to Control and Status word of a mailbox that has already been scanned, that is, it is behind Tx Arbitration Pointer, during the Tx arbitration process. If there is no inactive mailbox and only one Tx mailbox that is being transmitted then VPS is not asserted. This bit is negated upon the start of every Tx arbitration process or upon a write to Control and Status word of any mailbox. **NOTE:** ESR2[VPS] is not affected by any CPU write into Control Status (C/S) of an MB that is blocked by abort mechanism. When MCR[AEN] is asserted, the abort code write in C/S of an MB that is being transmitted (pending abort), or any write attempt into a Tx MB with IFLAG set is blocked. 0b - Contents of IMB and LPTM are invalid. 1b - Contents of IMB and LPTM are valid.
13 IMB	Inactive Mailbox If ESR2[VPS] is asserted, this bit indicates whether there is any inactive mailbox (CODE field is either 1000b or 0b). This bit is asserted in the following cases: • During arbitration, if an ESR2[LPTM] is found and it is inactive. • If ESR2[IMB] is not asserted and a frame is transmitted successfully. This bit is always cleared in start of arbitration (see Arbitration process).

Table continues on the next page...

Field	Function
	NOTE: ESR2[LPTM] mechanism has the following behavior: if an MB is successfully transmitted and ESR2[IMB]=0 (no inactive mailbox), then ESR2[VPS] and ESR2[IMB] are asserted and the index related to the MB just transmitted is loaded into ESR2[LPTM]. 0b - If ESR2[VPS] is asserted, the ESR2[LPTM] is not an inactive mailbox. 1b - If ESR2[VPS] is asserted, there is at least one inactive mailbox. LPTM content is the number of the first one.
12-0 —	Reserved

45.6.2.16 CRC Register (CRCR)

45.6.2.16.1 Offset

Register	Offset
CRCR	44h

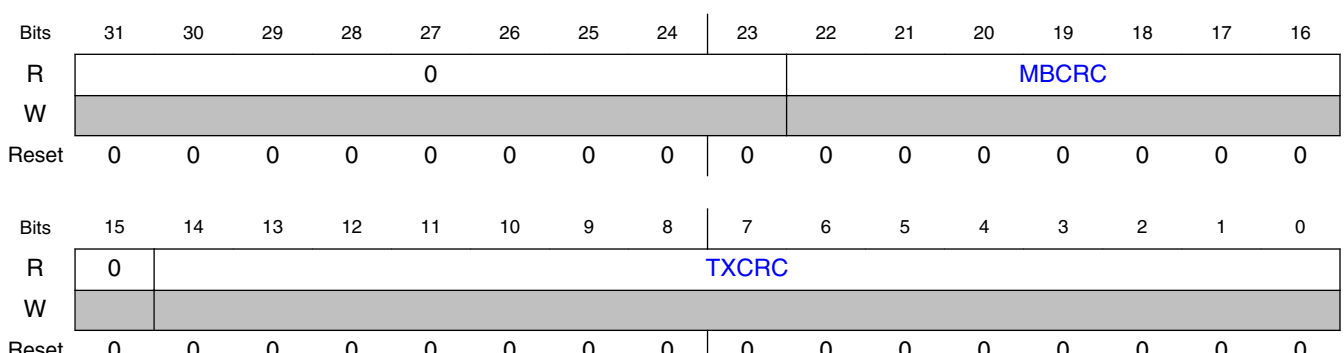
45.6.2.16.2 Function

This register provides information about the CRC of transmitted messages for non-FD messages. This register only reports the 15 low order bits of CRC calculations for messages in CAN FD format that require either 17 or 21 bits. For CAN FD format frames, the FDCRC register must be used. This register is updated at the same time the Tx interrupt flag is asserted.

NOTE

See CRC sequence calculation in the CAN Protocol standard (ISO 11898-1) for details.

45.6.2.16.3 Diagram



45.6.2.16.4 Fields

Field	Function
31-23 —	Reserved
22-16 MBCRC	CRC Mailbox This field indicates the number of the mailbox corresponding to the value in CRCR[TXCRC] field.
15 —	Reserved
14-0 TXCRC	Transmitted CRC value This field indicates the CRC value of the last transmitted message for non-FD frames. For FD frames, CRC value is reported in FDCRC register.

45.6.2.17 Rx FIFO Global Mask Register (RXFGMASK)

45.6.2.17.1 Offset

Register	Offset
RXFGMASK	48h

45.6.2.17.2 Function

This register is located in RAM.

If Rx FIFO is enabled, RXFGMASK is used to mask the Rx FIFO ID filter table elements that do not have a corresponding RXIMR according to CTRL2[RFFN] field setting.

This register can only be written in Freeze mode as it is blocked by hardware in other modes.

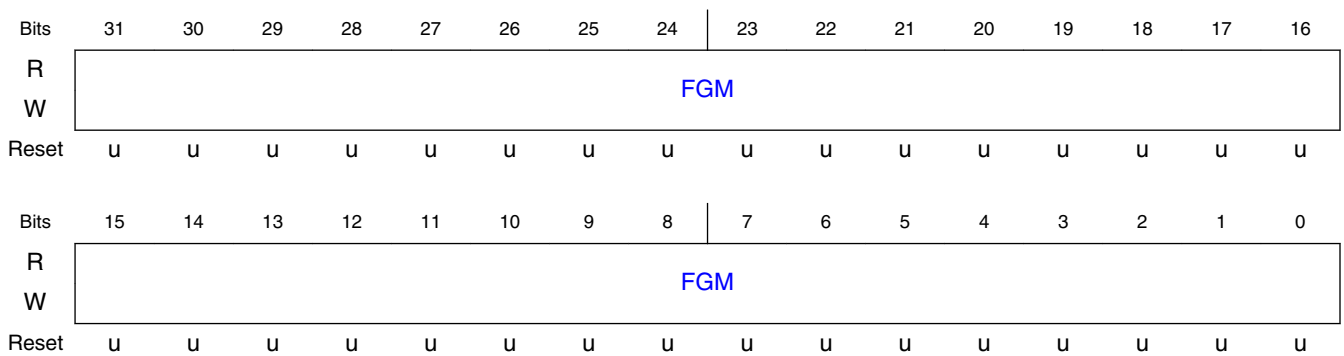
The following table shows how the FGM bits correspond to each IDAF field.

Table 45-24. Correspondence of Rx FIFO global mask bits to IDF fields

Rx FIFO ID filter table elements format (MCR[IDAM])	Identifier acceptance filter fields					
	RTR	IDE	RXIDA	RXIDB [1]	RXIDC [2]	Reserved
A	FGM[31]	FGM[30]	FGM[29:1]	—	—	FGM[0]
B	FGM[31], FGM[15]	FGM[30], FGM[14]	—	FGM[29:16], FGM[13:0]		—
C	—	—		—	FGM[31:24], FGM[23:16], FGM[15:8], FGM[7:0]	

1. If MCR[IDAM] is equivalent to format B, only the fourteen most significant bits of the identifier of the incoming frame are compared with the Rx FIFO filter.
2. If MCR[IDAM] is equivalent to format C, only the eight most significant bits of the identifier of the incoming frame are compared with the Rx FIFO filter.

45.6.2.17.3 Diagram



45.6.2.17.4 Fields

Field	Function
31-0 FGM	Rx FIFO Global Mask Bits These bits mask the ID filter table elements bits in a perfect alignment. 0b - The corresponding bit in the filter is "don't care." 1b - The corresponding bit in the filter is checked.

45.6.2.18 Rx FIFO Information Register (RXFIR)

45.6.2.18.1 Offset

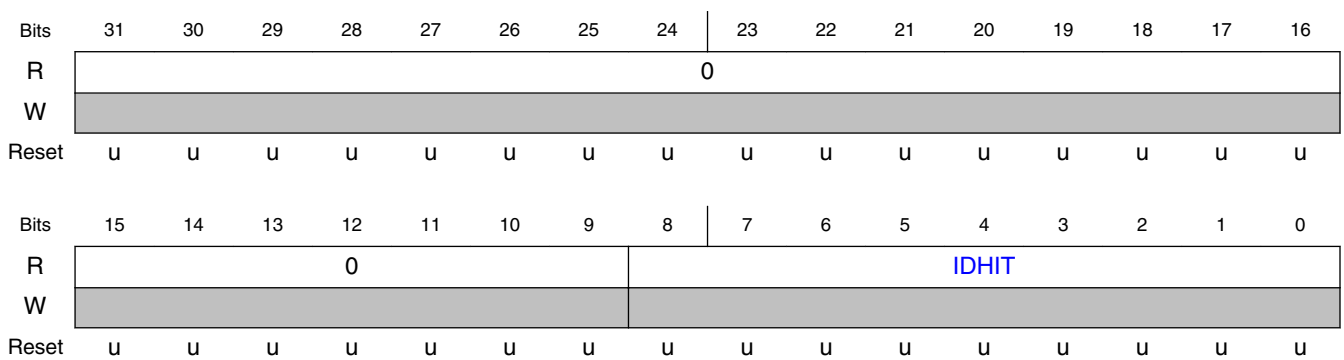
Register	Offset
RXFIR	4Ch

45.6.2.18.2 Function

RXFIR provides information on Rx FIFO.

This register is the port through which the CPU accesses the output of the RXFIR FIFO located in RAM. The RXFIR FIFO is written by the FlexCAN whenever a new message is moved into the Rx FIFO. Also, its output is updated whenever the output of the Rx FIFO is updated with the next message. See Rx FIFO for instructions on reading this register.

45.6.2.18.3 Diagram



45.6.2.18.4 Fields

Field	Function
31-9 —	Reserved
8-0 IDHIT	Identifier Acceptance Filter Hit Indicator This field indicates which Identifier Acceptance filter was hit by the received message that is in the output of the Rx FIFO. If multiple filters match the incoming message ID then the first matching IDAF found (lowest number) by the matching process is indicated. This field is valid only when IFLAG1[BUF5I] is asserted.

45.6.2.19 CAN Bit Timing Register (CBT)

45.6.2.19.1 Offset

Register	Offset
CBT	50h

45.6.2.19.2 Function

This register is an alternative way to store the CAN bit timing variables described in CTRL1 register. EPRESDIV, EPROPSEG, EPSEG1, EPSEG2, and ERJW are extended versions of PRESDIV, PROPSEG, PSEG1, PSEG2, and RJW fields respectively.

The BTF bit selects the use of the timing variables defined in this register.

The contents of this register are not affected by soft reset.

NOTE

The CAN bit variables in CTRL1 and in CBT are stored in the same register.

NOTE

When the CAN FD feature is enabled (MCR[FDEN] is set), always set CBT[BTF].

NOTE

The user must ensure bit time settings and protocol engine tolerance are in compliance with the CAN Protocol standard (ISO 11898-1).

45.6.2.19.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	BTF	EPRESDIV										ERJW				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	EPROPSEG						EPSEG1					EPSEG2				
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

45.6.2.19.4 Fields

Field	Function
31 BTF	Bit Timing Format Enable Enables the use of extended CAN bit timing fields EPRESDIV, EPROPSEG, EPSEG1, EPSEG2. and ERJW replacing the CAN bit timing variables defined in CTRL1 register. This field can be written in Freeze mode only. 0b - Extended bit time definitions disabled. 1b - Extended bit time definitions enabled.
30-21 EPRESDIV	Extended Prescaler Division Factor This 10-bit field defines the ratio between the PE clock frequency and the serial clock (Sclock) frequency when CBT[BTF] is asserted, otherwise it has no effect. It extends the CTRL1[PRESDIV] value range. The Sclock period defines the time quantum of the CAN protocol. For the reset value, the Sclock frequency is equal to the PE clock frequency (see Protocol timing). This field can be written only in Freeze mode because it is blocked by hardware in other modes. Sclock frequency = PE clock frequency / (EPRESDIV + 1)
20-16 ERJW	Extended Resync Jump Width This 5-bit field defines the maximum number of time quanta that a bit time can be changed by one re-synchronization when CBT[BTF] bit is asserted, otherwise it has no effect. It extends the CTRL1[RJW] value range. One time quantum is equal to the Sclock period. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Resync Jump Width = ERJW + 1.
15-10 EPROPSEG	Extended Propagation Segment This 6-bit field defines the length of the propagation segment in the bit time when CBT[BTF] bit is asserted, otherwise it has no effect. It extends the CTRL1[PROPSEG] value range. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Propagation Segment Time = (EPROPSEG + 1) × Time-Quanta. Time-Quantum = one Sclock period.
9-5 EPSEG1	Extended Phase Segment 1 This 5-bit field defines the length of phase segment 1 in the bit time when CBT[BTF] bit is asserted, otherwise it has no effect. It extends the CTRL1[PSEG1] value range. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Phase Buffer Segment 1 = (EPSEG1 + 1) × Time-Quanta. Time-Quantum = one Sclock period.
4-0 EPSEG2	Extended Phase Segment 2 This 5-bit field defines the length of phase segment 2 in the bit time when CBT[BTF] bit is asserted, otherwise it has no effect. It extends the CTRL1[PSEG2] value range. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Phase Buffer Segment 1 = (EPSEG2 + 1) × Time-Quanta. Time-Quantum = one Sclock period.

45.6.2.20 Rx Individual Mask Registers (RXIMR0 - RXIMR63)

45.6.2.20.1 Offset

For n = 0 to 63:

Register	Offset
RXIMRn	880h + (n × 4h)

45.6.2.20.2 Function

The RX Individual Mask registers are used to store the acceptance masks for ID filtering in Rx MBs and the Rx FIFO.

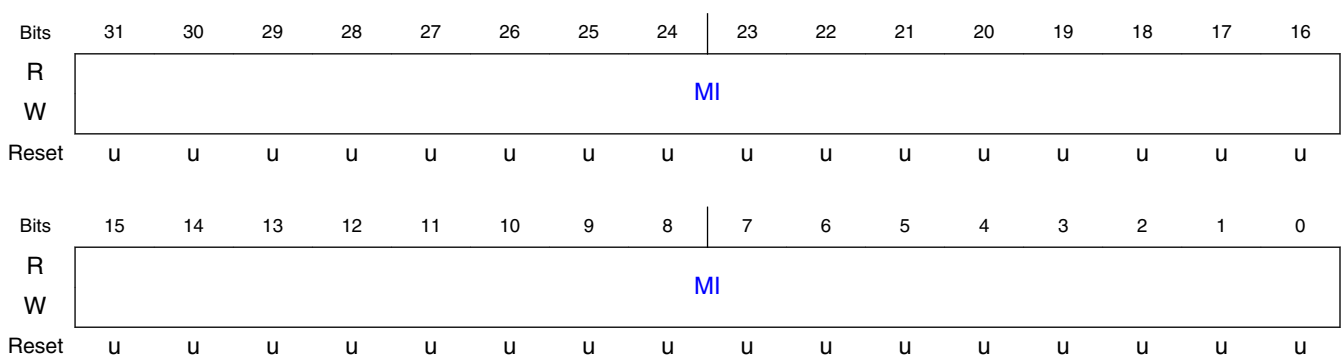
When the Rx FIFO is disabled (MCR[RFEN] bit is negated), an individual mask is provided for each available Rx mailbox on a one-to-one correspondence. When the Rx FIFO is enabled (MCR[RFEN] bit is asserted), an individual mask is provided for each Rx FIFO ID filter table element on a one-to-one correspondence depending on the setting of CTRL2[RFFN] (see Rx FIFO).

RXIMR0 stores the individual mask associated with either MB0 or ID filter table element 0, RXIMR1 stores the individual mask associated with either MB1 or ID filter table element 1, and so on.

RXIMR registers can only be accessed by the CPU when the module is in Freeze mode; otherwise, they are blocked by hardware. These registers are not affected by reset. They are located in RAM and must be explicitly initialized prior to any reception.

It is possible for the RXIMR memory region to be accessed as general purpose memory. See Bus interface for more information.

45.6.2.20.3 Diagram



45.6.2.20.4 Fields

Field	Function
31-0 MI	Individual Mask Bits Each individual mask bit masks the corresponding bit in both the mailbox filter and Rx FIFO ID filter table element in distinct ways. For mailbox filters, see the RXMGMASK register description. For Rx FIFO ID filter table elements, see the RXFGMASK register description. 0b - The corresponding bit in the filter is "don't care." 1b - The corresponding bit in the filter is checked.

45.6.2.21 CAN FD Control Register (FDCTRL)

45.6.2.21.1 Offset

Register	Offset
FDCTRL	C00h

45.6.2.21.2 Function

This register contains control bits for the CAN FD operation. It also defines the data size of message buffers allocated in different partitions of RAM (memory blocks) as described in the table below.

When an 8-byte payload is selected:

- Block R0 allocates MB0 to MB31.
- Block R1 allocates MB32 to MB63.

When a payload larger than 8 bytes is selected, the maximum number of MBs in a block is limited as described below:

Table 45-25. Number of message buffers

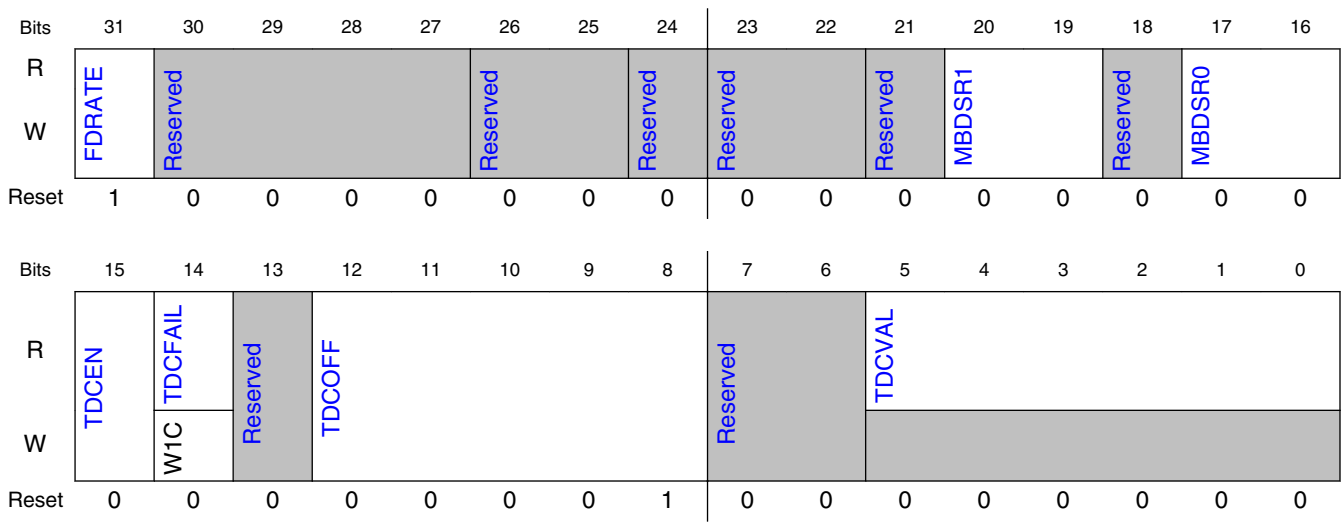
Payload size	Maximum number of message buffers per RAM block
8 bytes	32
16 bytes	21
32 bytes	12
64 bytes	7

NOTE

One memory block fits exactly 32 MBs with an 8-byte payload. For other possible payload sizes, empty memory may exist between the last MB in a block and the beginning of the next block. This empty memory corresponds to less than one MB, and must not be used.

The contents of this register are not affected by soft reset.

45.6.2.21.3 Diagram



45.6.2.21.4 Fields

Field	Function
31 FDRATE	Bit Rate Switch Enable This bit enables the effect of the Bit Rate Switch (BRS bit) during the data phase of Tx messages. The CPU can write this bit any time. However, its effect turns active only when the CAN bus is in Wait for Bus Idle, Bus Idle, or Bus Off state, or when the current frame under reception or transmission reaches the interframe space. By negating FDCTRL[FDRATE], the CPU can force all bits in CAN FD messages to be transmitted in nominal bit rate, no matter what the value is in the BRS bit of the Tx MBs. 0b - Transmit a frame in nominal rate. The BRS bit in the Tx MB has no effect. 1b - Transmit a frame with bit rate switching if the BRS bit in the Tx MB is recessive.
30-27 —	Reserved
26-25 —	Reserved

Table continues on the next page...

Field	Function
24 —	Reserved
23-22 —	Reserved
21 —	Reserved
20-19 MBDSR1	Message Buffer Data Size for Region 1 This two-bit field selects the data size (8, 16, 32, or 64 bytes) for region R1 of message buffers allocated in RAM. It can be written in Freeze mode only. 00b - Selects 8 bytes per message buffer. 01b - Selects 16 bytes per message buffer. 10b - Selects 32 bytes per message buffer. 11b - Selects 64 bytes per message buffer.
18 —	Reserved
17-16 MBDSR0	Message Buffer Data Size for Region 0 This two-bit field selects the data size (8, 16, 32, or 64 bytes) for region R0 of message buffers allocated in RAM. It can be written in Freeze mode only. 00b - Selects 8 bytes per message buffer. 01b - Selects 16 bytes per message buffer. 10b - Selects 32 bytes per message buffer. 11b - Selects 64 bytes per message buffer.
15 TDCEN	Transceiver Delay Compensation Enable This bit can be used to enable and disable the TDC feature. It can be written in Freeze mode only. **NOTE:** See Transmitter delay compensation in the CAN Protocol standard (ISO 11898-1) for details. **NOTE:** TDC must be disabled when Loop Back mode is enabled (see CTRL1[LPB]). 0b - TDC is disabled 1b - TDC is enabled
14 TDCFAIL	Transceiver Delay Compensation Fail This bit indicates when the Transceiver Delay Compensation (TDC) mechanism is out of range, unable to compensate the transceiver's loop delay and successfully compare the delayed received bits to the transmitted ones (see Transceiver delay compensation). TDCFAIL sets in the first time FlexCAN detects the out of range condition. The CPU needs to write one to clear it. 0b - Measured loop delay is in range. 1b - Measured loop delay is out of range.
13 —	Reserved
12-8 TDCOFF	Transceiver Delay Compensation Offset This bit field contains the offset value to be added to the measured transceiver's loop delay in order to define the position of the delayed comparison point when bit rate switching is active. See Transceiver delay compensation for more details on how the loop delay measurement is performed.

Table continues on the next page...

Field	Function
	TDCOFF can be written in Freeze mode only. Its value can be defined in Protocol Engine (PE) Clock periods (CANCLK, see Protocol timing for more details), and must be selected to be smaller than the CAN bit duration in the data bit rate for proper operation. **NOTE:** It is not recommended to have TDCOFF equal to zero.
7-6 —	Reserved
5-0 TDCVAL	Transceiver Delay Compensation Value This register contains the value of the transceiver loop delay measured from the transmitted EDL to R0 transition edge to the respective received one added to the TDCOFF value specified in the FDCTRL register. This value is an integer multiple of the Protocol Engine (PE) Clock period (CANCLK). See Protocol timing for more details on how the loop delay measurement is performed.

45.6.2.22 CAN FD Bit Timing Register (FDCBT)

45.6.2.22.1 Offset

Register	Offset
FDCBT	C04h

45.6.2.22.2 Function

This register stores the CAN bit timing variables used in the data phase of CAN FD messages when the FDCTRL[FDRATE] is set, compatible with CAN FD specification. FPRESDIV, FPROPSEG, FPSEG1, FPSEG2, and FRJW are used to define the time quantum duration, the number of time quanta per CAN bit, and the sample point position for the data bit rate portion of a CAN FD message with the BRS bit set.

The contents of this register are not affected by soft reset.

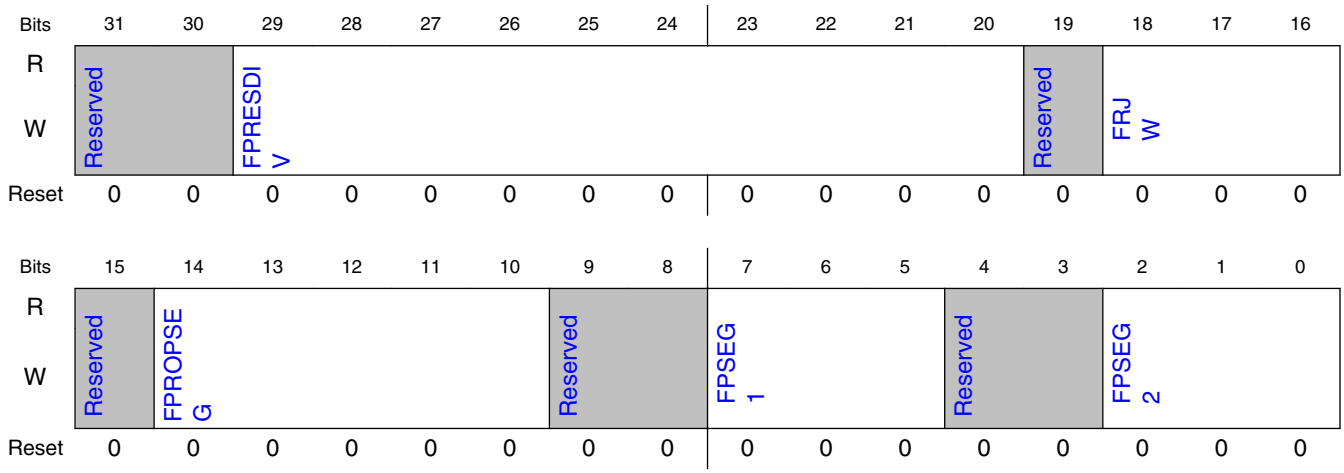
NOTE

The sum of the Fast Propagation Segment (FPROPSEG) and Fast Phase Segment 1 (FPSEG1) must be at least two time quanta.

NOTE

The user must ensure bit time settings and protocol engine tolerance are in compliance with the CAN Protocol standard (ISO 11898-1).

45.6.2.22.3 Diagram



45.6.2.22.4 Fields

Field	Function
31-30 —	Reserved
29-20 FPRESDIV	Fast Prescaler Division Factor This 10-bit field defines the ratio between the PE clock frequency and the serial clock (Sclock) frequency in the data bit rate portion of a CAN FD message with the BRS bit set. The Sclock period defines the time quantum of the CAN FD protocol for the data bit rate. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Sclock frequency = PE clock frequency / (FPRESDIV + 1). **NOTE:** To minimize errors when processing FD frames, use the same value for FPRESDIV and PRESDIV (in CBT or CTRL1). For more details see the first NOTE in section CAN FD frames.
19 —	Reserved
18-16 FRJW	Fast Resync Jump Width This 3-bit field defines the maximum number of time quanta that a bit time can be changed by one resynchronization in the data bit rate portion of a CAN FD message with the BRS bit set. One time quantum is equal to the Sclock period. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Resync Jump Width = FSJW + 1.
15 —	Reserved
14-10 FPROPSEG	Fast Propagation Segment This 5-bit field defines the length of the propagation segment in the bit time in the data bit rate portion of a CAN FD message with the BRS bit set. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Propagation Segment Time = FPROPSEG × Time-Quanta.

Table continues on the next page...

Field	Function
	Time-Quantum = one Sclock period.
9-8 —	Reserved
7-5 FPSEG1	Fast Phase Segment 1 This 3-bit field defines the length of phase segment 1 in the bit time in the data bit rate portion of a CAN FD message with the BRS bit set. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Phase Segment 1 = (FPSEG1 + 1) × Time-Quanta. Time-Quantum = one Sclock period.
4-3 —	Reserved
2-0 FPSEG2	Fast Phase Segment 2 This 3-bit field defines the length of phase segment 2 in the data bit rate portion of a CAN FD message with the BRS bit set. This field can be written only in Freeze mode because it is blocked by hardware in other modes. Phase Segment 2 = (FPSEG2 + 1) × Time-Quanta. Time-Quantum = one Sclock period.

45.6.2.23 CAN FD CRC Register (FDCRC)

45.6.2.23.1 Offset

Register	Offset
FDCRC	C08h

45.6.2.23.2 Function

This register provides information about the CRC of transmitted messages.

FlexCAN uses different CRC polynomials for different frame formats, as shown below.

The CRC_15 polynomial is used for all frames in CAN format. The CRC_17 polynomial is used for frames in CAN FD format with a DATA FIELD up to sixteen bytes. The CRC_21 polynomial is used for frames in CAN FD format with a DATA FIELD longer than sixteen bytes. Each polynomial shown below results in a Hamming distance of 6. This register is updated at the same time the Tx Interrupt flag is asserted.

$CRC_15 = C599h: (x^{15} + x^{14} + x^{10} + x^8 + x^7 + x^4 + x^3 + 1)$

$CRC_17 = 3685Bh: (x^{17} + x^{16} + x^{14} + x^{13} + x^{11} + x^6 + x^4 + x^3 + x^1 + 1)$

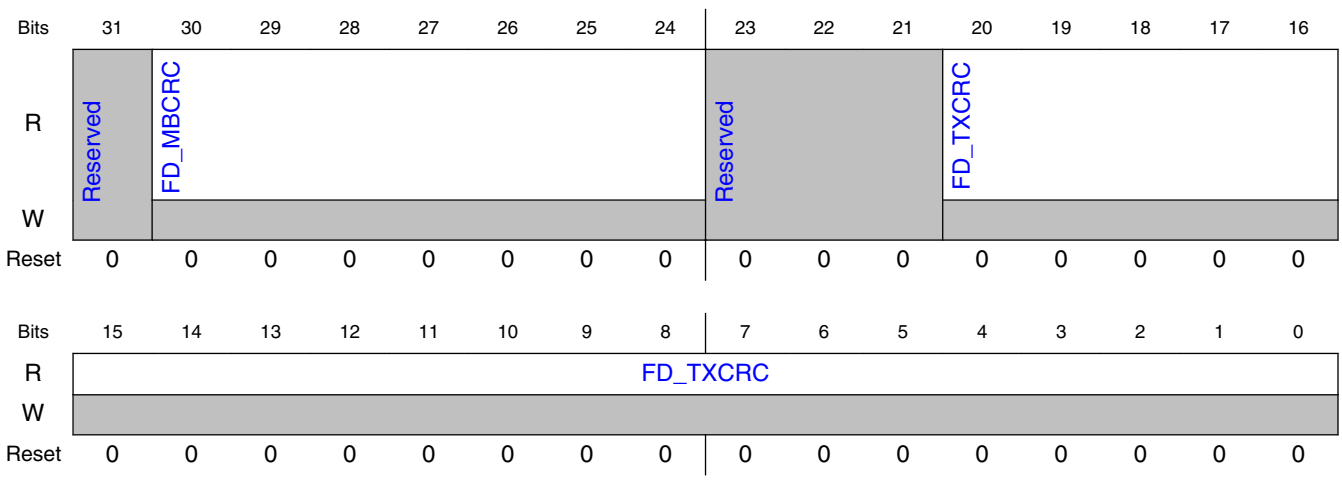
$CRC_21 = 302899h: (x^{21} + x^{20} + x^{13} + x^{11} + x^7 + x^4 + x^3 + 1)$

Figure 45-28. CRC polynomial used on CAN frame

NOTE

See CRC sequence calculation in the CAN Protocol standard (ISO 11898-1) for details.

45.6.2.23.3 Diagram



45.6.2.23.4 Fields

Field	Function
31 —	Reserved
30-24 FD_MBCRC	CRC Mailbox Number for FD_TXCRC This field indicates the number of the mailbox corresponding to the value in the FD_TXCRC field, for both FD and non-FD frames. It reports the same information as in CRCR[MBCRC].
23-21 —	Reserved
20-0 FD_TXCRC	Extended Transmitted CRC value This 21-bit field contains the CRC value calculated over the most recent transmitted message. Different CRC polynomials are used for different frame formats. A 15-bit polynomial, CRC_15, is used for all frames in CAN format. The second 17-bit polynomial, CRC_17, is used for frames in CAN FD format with a data field up to sixteen bytes long. The third 21-bit polynomial, CRC_21, is used for frames in CAN FD format with a data field longer than sixteen bytes.

Field	Function
	For CRC_15 and CRC_17, the 6 most significant bits and the 4 most significant bits are reported as zeros, respectively. For CRC_15, this register has the same content as CRC register.

45.6.3 Message buffer structure

The message buffer structure used by the FlexCAN module is represented in the following figure. Both extended (29-bit identifier) and standard (11-bit identifier) frames used in the CAN specification (Version 2.0 Part B) are represented. Each individual MB is formed by 16, 24, 40, or 72 bytes, depending on the quantity of data bytes allocated for the message payload: 8, 16, 32, or 64 data bytes, respectively.

The memory area 80h–47Fh is used by the mailboxes. When CAN FD is enabled, the exact address for each MB depends on the size of its payload. See FlexCAN memory partition for CAN FD for more detailed information.

Table 45-26. Message buffer structure — example with 64-byte payload

	31	30	29	28	27	24	23	22	21	20	19	18	17	16	15	8	7	0
0h	EDL	BRS	ESI		CODE			SRR	IDE	RTR	DLC				TIME STAMP			
4h	PRIO			ID (standard/extended)											ID (extended)			
8h	Data byte 0							Data byte 1							Data byte 2		Data byte 3	
Ch	Data byte 4							Data byte 5							Data byte 6		Data byte 7	
10h	Data byte 8							Data byte 9							Data byte 10		Data byte 11	
14h	Data byte 12							Data byte 13							Data byte 14		Data byte 15	
18h	Data byte 16							Data byte 17							Data byte 18		Data byte 19	
1Ch	Data byte 20							Data byte 21							Data byte 22		Data byte 23	
20h	Data byte 24							Data byte 25							Data byte 26		Data byte 27	
24h	Data byte 28							Data byte 29							Data byte 30		Data byte 31	
28h	Data byte 32							Data byte 33							Data byte 34		Data byte 35	
2Ch	Data byte 36							Data byte 37							Data byte 38		Data byte 39	
30h	Data byte 40							Data byte 41							Data byte 42		Data byte 43	
34h	Data byte 44							Data byte 45							Data byte 46		Data byte 47	
38h	Data byte 48							Data byte 49							Data byte 50		Data byte 51	
3Ch	Data byte 52							Data byte 53							Data byte 54		Data byte 55	
40h	Data byte 56							Data byte 57							Data byte 58		Data byte 59	
44h	Data byte 60							Data byte 61							Data byte 62		Data byte 63	
				= Unimplemented or reserved														

EDL — Extended Data Length

This bit distinguishes between CAN format and CAN FD format frames. The EDL bit must not be set for message buffers configured to RANSWER with code field 1010b (see table below).

BRS — Bit Rate Switch

This bit defines whether the bit rate is switched inside a CAN FD format frame.

ESI — Error State Indicator

This bit indicates if the transmitting node is error active or error passive.

CODE — Message Buffer Code

This 4-bit field can be accessed (read or write) by the CPU and by the FlexCAN module itself, as part of the message buffer matching and arbitration process. The encoding is shown in Table 45-27 and Table 45-28. See Functional description for additional information.

Table 45-27. Message buffer code for Rx buffers

CODE description	Rx code BEFORE receive new frame	SRV[1]	Rx code AFTER successful reception[2]	RRS[3]	Comment
0000b: INACTIVE — MB is not active.	INACTIVE	—	—	—	MB does not participate in the matching process.
0100b: EMPTY — MB is active and empty.	EMPTY	—	FULL	—	When a frame is received successfully (after the Move-in process), the CODE field is automatically updated to FULL.
0010b: FULL — MB is full.	FULL	Yes	FULL	—	The act of reading the C/S word followed by unlocking the MB (SRV) does not make the code return to EMPTY. It remains FULL. If a new frame is moved to the MB after the MB was serviced, the code still remains FULL. See Matching process for matching details related to FULL code.

Table continues on the next page...

Table 45-27. Message buffer code for Rx buffers (continued)

CODE description	Rx code BEFORE receive new frame	SRV[1]	Rx code AFTER successful reception[2]	RRS[3]	Comment
		No	OVERRUN	—	If the MB is FULL and a new frame is moved to this MB before the CPU services it, the CODE field is automatically updated to OVERRUN. See Matching process for details about overrun behavior.
0110b: OVERRUN — MB is being overwritten into a full buffer.	OVERRUN	Yes	FULL	—	If the CODE field indicates OVERRUN and CPU has serviced the MB, when a new frame is moved to the MB then the code returns to FULL.
		No	OVERRUN	—	If the CODE field already indicates OVERRUN, and another new frame must be moved, the MB will be overwritten again, and the code will remain OVERRUN. See Matching process for details about overrun behavior.
1010b: RANSWER[4] — A frame was configured to recognize a Remote Request frame and transmit a Response frame in return.[5]	RANSWER	—	TANSWER(1110b)	0	A Remote Answer was configured to recognize a remote request frame received. After that an MB is set to transmit a response frame. The code is automatically changed to TANSWER (1110b). See Matching process for details. If CTRL2[RRS] is negated, transmit a response frame whenever a remote

Table continues on the next page...

Table 45-27. Message buffer code for Rx buffers (continued)

CODE description	Rx code BEFORE receive new frame	SRV[1]	Rx code AFTER successful reception[2]	RRS[3]	Comment
					request frame with the same ID is received.
		—	—	1	This code is ignored during matching and arbitration process. See Matching process for details.
CODE[0]=1: BUSY — FlexCAN is updating the contents of the MB. The CPU must not access the MB.	BUSY[6]	—	FULL	—	Indicates that the MB is being updated. It will be negated automatically and does not interfere with the next CODE.
		—	OVERRUN	—	

1. SRV: Serviced MB. MB was read and unlocked by reading TIMER or other MB.
2. A frame is considered a successful reception after the frame to be moved to MB (move-in process). See Move-in for details.
3. Remote Request Stored bit, see "Control 2 register (CTRL2)" for details.
4. Code 1010b is not considered Tx and an MB with this code should not be aborted.
5. Code 1010b must be used in message buffers configured in CAN FD format, having the EDL bit set.
6. Note that for Tx MBs, the BUSY bit should be ignored upon read, except when AEN bit is set in the MCR register. If this bit is asserted, the corresponding MB does not participate in the matching process.

Table 45-28. Message buffer code for Tx buffers

CODE Description	Tx Code BEFORE tx frame	MB RTR	Tx Code AFTER successful transmission	Comment
1000b: INACTIVE — MB is not active	INACTIVE	—	—	MB does not participate in arbitration process.
1001b: ABORT — MB is aborted	ABORT	—	—	MB does not participate in arbitration process.
1100b: DATA — MB is a Tx data frame (MB RTR must be 0)	DATA	0	INACTIVE	Transmit data frame unconditionally once. After transmission, the MB automatically returns to the INACTIVE state.
1100b: REMOTE — MB is a Tx Remote Request frame (MB RTR must be 1)	REMOTE	1	EMPTY	Transmit remote request frame unconditionally once. After transmission, the MB automatically becomes an Rx Empty MB with the same ID.

Table continues on the next page...

Table 45-28. Message buffer code for Tx buffers (continued)

CODE Description	Tx Code BEFORE tx frame	MB RTR	Tx Code AFTER successful transmission	Comment
1110b: TANSWER — MB is a Tx Response frame from an incoming Remote Request frame	TANSWER	—	RANSWER	This is an intermediate code that is automatically written to the MB by the CHI as a result of a match to a remote request frame. The remote response frame will be transmitted unconditionally once, and then the code will automatically return to RANSWER (1010b). The CPU can also write this code with the same effect. The remote response frame can be either a data frame or another remote request frame depending on the RTR bit value. See Matching process and Arbitration process for details.

SRR — Substitute Remote Request

Fixed recessive bit, used only in extended format. It must be set to one by the user for transmission (Tx Buffers) and will be stored with the value received on the CAN bus for Rx receiving buffers. It can be received as either recessive or dominant. If FlexCAN receives this bit as dominant, then it is interpreted as an arbitration loss.

1 = Recessive value is compulsory for transmission in extended format frames

0 = Dominant is not a valid value for transmission in extended format frames

IDE — ID Extended Bit

This field identifies whether the frame format is standard or extended.

1 = Frame format is extended

0 = Frame format is standard

RTR — Remote Transmission Request

This bit affects the behavior of remote frames and is part of the reception filter. See Table 45-27, Table 45-28, and the description of the RRS field in Control 2 register (CTRL2) for additional details.

If FlexCAN transmits this bit as '1' (recessive) and receives it as '0' (dominant), it is interpreted as an arbitration loss. If this bit is transmitted as '0' (dominant), then if it is received as '1' (recessive), the FlexCAN module treats it as a bit error. If the value received matches the value transmitted, it is considered a successful bit transmission.

1 = Indicates the current MB may have a remote request frame to be transmitted if MB is Tx. If the MB is Rx then incoming remote request frames may be stored.

0 = Indicates the current MB has a data frame to be transmitted. In Rx MB it may be considered in matching processes.

NOTE

When configuring CAN FD frames, the RTR bit must be negated.

DLC — Length of Data in Bytes

This 4-bit field is the length (in bytes) of the Rx or Tx data, which is located in offset 8h–Fh of the MB space (see Table 45-26). In reception, this field is written by the FlexCAN module, copied from the DLC (Data Length Code) field of the received frame. In transmission, this field is written by the CPU and corresponds to the DLC field value of the frame to be transmitted. When RTR = 1, the frame to be transmitted is a remote frame and does not include the data field, regardless of the DLC field (see Table 45-29).

TIME STAMP — Free-Running Counter Time Stamp

This 16-bit field is a copy of the Free-Running Timer, captured for Tx and Rx frames at the time when the beginning of the Identifier field appears on the CAN bus.

PRIO — Local priority

This 3-bit field is used only when MCR[LPRIO_EN] is set, and it only makes sense for Tx mailboxes. These bits are not transmitted. They are appended to the regular ID to define the transmission priority. See Arbitration process.

ID — Frame Identifier

In standard frame format, only the 11 most significant bits (28 to 18) are used for frame identification in both receive and transmit cases. The 18 least significant bits are ignored. In extended frame format, all bits are used for frame identification in both receive and transmit cases.

DATA BYTE 0 to 63 — Data Field

Up to sixty four bytes can be used for a data frame, depending on the size of payload selected for the message buffers.

For Rx frames, the data is stored as it is received from the CAN bus. DATA BYTE (*n*) is valid only if *n* is less than DLC as shown in the table below.

Table 45-29. DATA BYTEs validity

DLC	Valid DATA BYTEs
0	none
1	DATA BYTE 0
2	DATA BYTE 0 to 1
3	DATA BYTE 0 to 2
4	DATA BYTE 0 to 3
5	DATA BYTE 0 to 4
6	DATA BYTE 0 to 5
7	DATA BYTE 0 to 6
8	DATA BYTE 0 to 7
9	DATA BYTE 0 to 11
10	DATA BYTE 0 to 15
11	DATA BYTE 0 to 19
12	DATA BYTE 0 to 23
13	DATA BYTE 0 to 31
14	DATA BYTE 0 to 47
15	DATA BYTE 0 to 63

45.6.4 FlexCAN memory partition for CAN FD

When CAN FD is enabled, the FlexCAN RAM can be partitioned in blocks of 512 bytes. Each block can accommodate a number of message buffers which depends on the configuration provided by FDCTRL[MBDSRn] bit fields as shown in table below.

Table 45-30. RAM partition

RAM block	Number of MBs with 8 bytes (default range)	Size control bit field in FDCTRL register	Number of MBs of different sizes, per block
0	0 to 31	MBDSR0	MBDSR0=00, 32 MBs with 8 bytes payload MBDSR0=01, 21 MBs with 16 bytes payload MBDSR0=10, 12 MBs with 32 bytes payload MBDSR0=11, 7 MBs with 64 bytes payload
1	32 to 63	MBDSR1	MBDSR1=00, 32 MBs with 8 bytes payload

Table 45-30. RAM partition

RAM block	Number of MBs with 8 bytes (default range)	Size control bit field in FDCTRL register	Number of MBs of different sizes, per block
			MBDSR1=01, 21 MBs with 16 bytes payload MBDSR1=10, 12 MBs with 32 bytes payload MBDSR1=11, 7 MBs with 64 bytes payload

When payload sizes of 16, 32, or 64 bytes are configured in some or all RAM blocks, the total number of MBs and its respective number order may differ from the default configuration of 8 bytes. For example, suppose Block0 is configured to 8 bytes payload, Block1 to 16 bytes, Then the following table indicates how the message buffers will be arranged in RAM.

Table 45-31. RAM partition example

RAM block	Payload size	Number of MBs in the RAM block	Message buffer range
0	FDCTRL[MBDSR0]=00, 8 bytes payload	32	0 to 31
1	FDCTRL[MBDSR1]=01, 16 bytes payload	21	32 to 52

45.6.5 FlexCAN message buffer memory map

The FlexCAN memory buffers are allocated in memory according to the tables below.

Table 45-32. 8-byte message buffers

Address offset (hex)	MBDSR=b00 8-byte payload
0080	MB0
0090	MB1
00A0	MB2
00B0	MB3
00C0	MB4
00D0	MB5
00E0	MB6
00F0	MB7
0100	MB8

Table continues on the next page...

Table 45-32. 8-byte message buffers (continued)

Address offset (hex)	MBDSR=b00 8-byte payload
0110	MB9
0120	MB10
0130	MB11
0140	MB12
0150	MB13
0160	MB14
0170	MB15
0180	MB16
0190	MB17
01A0	MB18
01B0	MB19
01C0	MB20
01D0	MB21
01E0	MB22
01F0	MB23
0200	MB24
0210	MB25
0220	MB26
0230	MB27
0240	MB28
0250	MB29
0260	MB30
0270	MB31
0280	MB32
0290	MB33
02A0	MB34
02B0	MB35
02C0	MB36
02D0	MB37
02E0	MB38
02F0	MB39
0300	MB40
0310	MB41
0320	MB42
0330	MB43
0340	MB44
0350	MB45
0360	MB46

Table continues on the next page...

Table 45-32. 8-byte message buffers (continued)

Address offset (hex)	MBDSR=b00 8-byte payload
0370	MB47
0380	MB48
0390	MB49
03A0	MB50
03B0	MB51
03C0	MB52
03D0	MB53
03E0	MB54
03F0	MB55
0400	MB56
0410	MB57
0420	MB58
0430	MB59
0440	MB60
0450	MB61
0460	MB62
0470	MB63

Table 45-33. 16-byte message buffers

Address offset (hex)	MBDSR=b01 16-byte payload
0080	MB0
0098	MB1
00B0	MB2
00C8	MB3
00E0	MB4
00F8	MB5
0110	MB6
0128	MB7
0140	MB8
0158	MB9
0170	MB10
0188	MB11
01A0	MB12
01B8	MB13
01D0	MB14
01E8	MB15

Table continues on the next page...

Table 45-33. 16-byte message buffers (continued)

Address offset (hex)	MBDSR=b01 16-byte payload
0200	MB16
0218	MB17
0230	MB18
0248	MB19
0260	MB20
0280	MB21
0298	MB22
02B0	MB23
02C8	MB24
02E0	MB25
02F8	MB26
0310	MB27
0328	MB28
0340	MB29
0358	MB30
0370	MB31
0388	MB32
03A0	MB33
03B8	MB34
03D0	MB35
03E8	MB36
0400	MB37
0418	MB38
0430	MB39
0448	MB40
0460	MB41

Table 45-34. 32-byte message buffers

Address offset (hex)	MBDSR=b10 32-byte payload
0080	MB0
00A8	MB1
00D0	MB2
00F8	MB3
0120	MB4
0148	MB5
0170	MB6

Table continues on the next page...

Table 45-34. 32-byte message buffers (continued)

Address offset (hex)	MBDSR=b10 32-byte payload
0198	MB7
01C0	MB8
01E8	MB9
0210	MB10
0238	MB11
0280	MB12
02A8	MB13
02D0	MB14
02F8	MB15
0320	MB16
0348	MB17
0370	MB18
0398	MB19
03C0	MB20
03E8	MB21
0410	MB22
0438	MB23

Table 45-35. 64-byte message buffers

Address offset (hex)	MBDSR=b11 64-byte payload
0080	MB0
00C8	MB1
0110	MB2
0158	MB3
01A0	MB4
01E8	MB5
0230	MB6
0280	MB7
02C8	MB8
0310	MB9
0358	MB10
03A0	MB11
03E8	MB12
0430	MB13

45.6.6 Rx FIFO structure

When MCR[RFEN] is set, the memory area 80h–DCh (which is normally occupied by MBs 0–5) is used by the reception FIFO engine.

The region 80h–8Ch contains the output of the FIFO which must be read by the CPU as a message buffer. This output contains the oldest message that has been received but not yet read. The region 90h–DCh is reserved for internal use of the FIFO engine.

An additional memory area, which starts at E0h and may extend up to 2DCh (normally occupied by MBs 6–37) depending on the CTRL2[RFFN] field setting, contains the ID filter table (configurable from 8 to 128 table elements) that specifies filtering criteria for accepting frames into the FIFO.

Out of reset, the ID filter table flexible memory area defaults to E0h and extends only to FCh, which corresponds to MBs 6 to 7 for RFFN = 0, for backward compatibility with previous versions of FlexCAN.

The following shows the Rx FIFO data structure.

Table 45-36. Rx FIFO structure

	31 28 24	23	22	21	20	19 18 17 16	15 8	7 0
80h	IDHIT		SRR	IDE	RTR	DLC	TIME STAMP	
84h		ID standard				ID extended		
88h	Data byte 0		Data byte 1				Data byte 2	Data byte 3
8Ch	Data byte 4		Data byte 5				Data byte 6	Data byte 7
90h–DCh	Reserved							
E0h	ID filter table element 0							
E4h	ID filter table element 1							
E8h–2D4h	ID filter table elements 2 to 125							
2D8h	ID filter table element 126							
2DCh	ID filter table element 127							
	= Unimplemented or reserved							

Each ID filter table element occupies an entire 32-bit word and can be compounded by one, two, or four Identifier Acceptance Filters (IDAF) depending on the MCR[IDAM] field setting. The following figures show the IDAF indexation.

The following table shows the three different formats of the ID table elements. Note that all elements of the table must have the same format. See Rx FIFO for more information.

Table 45-37. ID table structure

Format	31	30	29 24	23 16	15	14	13 8	7 1	0
A	RTR	IDE	RXIDA (standard = 29–19, extended = 29–1)						
B	RTR	IDE	RXIDB_0 (standard = 29–19, extended = 29–16)		RTR	IDE	RXIDB_1 (standard = 13–3, extended = 13–0)		
C	RXIDC_0 (std/ext = 31–24)		RXIDC_1 (std/ext = 23–16)		RXIDC_2 (std/ext = 15–8)		RXIDC_3 (std/ext = 7–0)		
	= Unimplemented or Reserved								

RTR — Remote frame

This bit specifies if remote frames are accepted into the FIFO if they match the target ID.

1 = Remote frames can be accepted and data frames are rejected

0 = Remote frames are rejected and data frames can be accepted

IDE — Extended frame

Specifies whether extended or standard frames are accepted into the FIFO if they match the target ID.

1 = Extended frames can be accepted and standard frames are rejected

0 = Extended frames are rejected and standard frames can be accepted

RXIDA — Rx frame identifier (Format A)

Specifies an ID to be used as acceptance criteria for the FIFO. In the standard frame format, only the 11 most significant bits (29 to 19) are used for frame identification. In the extended frame format, all bits are used.

RXIDB_0, RXIDB_1 — Rx frame identifier (Format B)

Specifies an ID to be used as acceptance criteria for the FIFO. In the standard frame format, the 11 most significant bits (a full standard ID) (29 to 19 and 13 to 3) are used for frame identification. In the extended frame format, all 14 bits of the field are compared to the 14 most significant bits of the received ID.

RXIDC_0, RXIDC_1, RXIDC_2, RXIDC_3 — Rx frame identifier (Format C)

Specifies an ID to be used as acceptance criteria for the FIFO. In both standard and extended frame formats, all 8 bits of the field are compared to the 8 most significant bits of the received ID.

IDHIT — Identifier acceptance filter hit indicator

This 9-bit field indicates which identifier acceptance filter was hit by the received message that is in the output of the Rx FIFO. See Rx FIFO for more information.

Chapter 46
Keypad Port (KPP)

46.1 Chip-specific KPP information

Table 46-1. Reference links to related information

Topic	Related module(s)	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
Interrupts, DMA Events and XBAR Assignments	-	Interrupts, DMA Events and XBAR Assignments

46.2 Overview

The Keypad Port (KPP) is a 16-bit peripheral that can be used as a keypad matrix interface or as general purpose input/output (I/O). The figure below shows the KPP block diagram. The KPP provides interface for the keypad matrix with 2-point contact or 3-point contact keys. The KPP is designed to simplify the software task of scanning a keypad matrix. With appropriate software support, the KPP is capable of detecting, debouncing, and decoding one or multiple keys pressed simultaneously on the keypad.

46.2.1 Block diagram

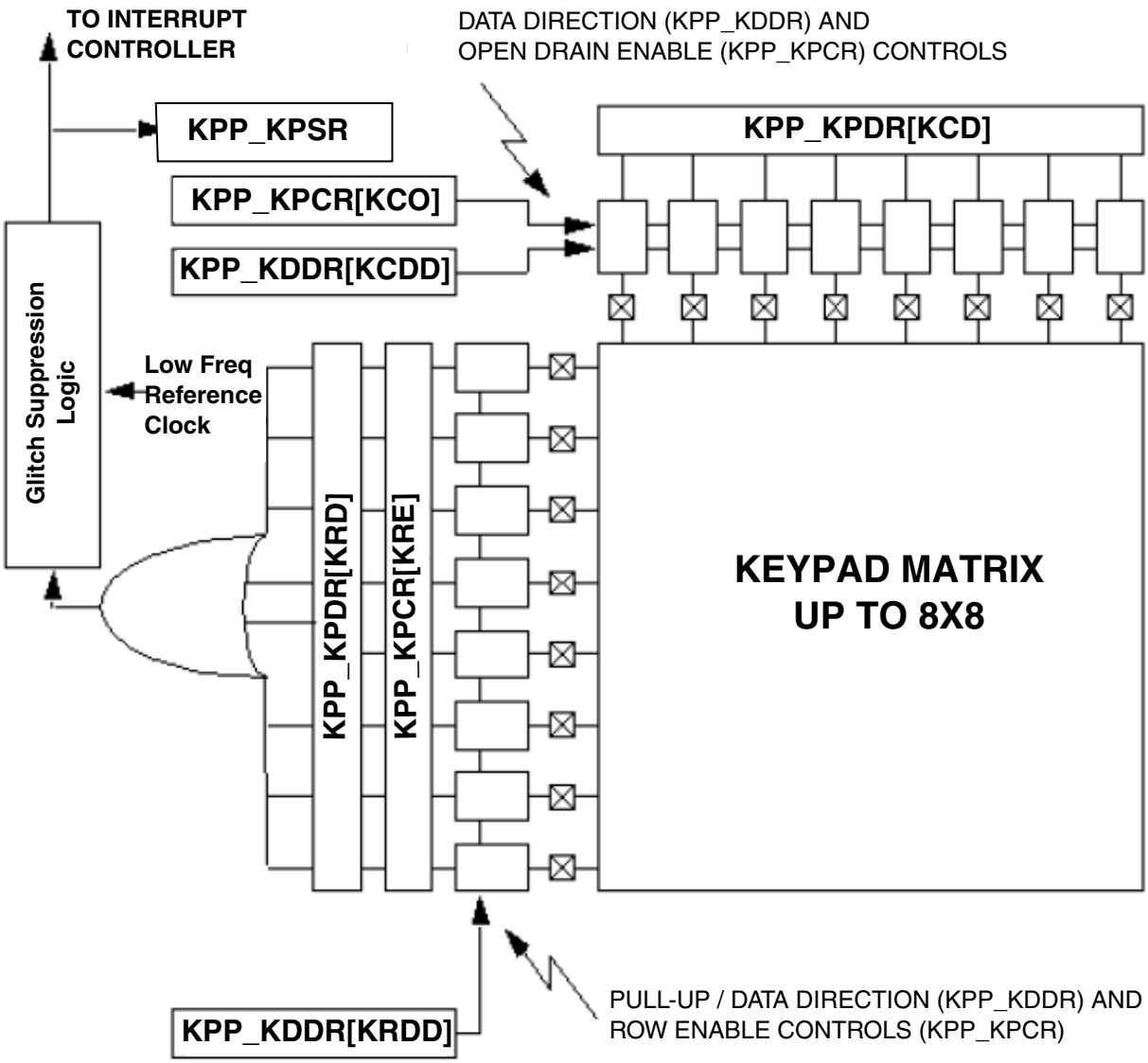


Figure 46-1. KPP Peripheral Block Diagram

46.2.2 Features

The KPP includes these distinctive features:

- Supports up to an 8 x 8 external key pad matrix
- Port pins can be used as general purpose I/O
- Open drain design
- Glitch suppression circuit design
- Multiple-key detection

- Long key-press detection
- Standby key-press detection
- Synchronizer chain clear
- Supports a 2-point and 3-point contact key matrix

46.3 Functional Description

The Keypad Port (KPP) is designed to simplify the software task of scanning a keypad matrix. With appropriate software support and matrix organization, the KPP is capable of detecting, debouncing, and decoding one or more keys pressed simultaneously on the keypad.

Logic in the KPP is capable of detecting a key press even while the processor is in one of the low power standby modes provided that a low frequency reference clock (ipg_clk_32k) is on. The KPP may generate an ARM platform interrupt any time a key press or key release is detected. This interrupt is capable of forcing the processor out of a low power mode.

46.3.1 Modes and Operations

This block supports the following modes:

- Run Mode-This is the normal functional mode in which the KPP can detect any key press event.
- Low Power Mode-The keypad can detect any key press even in low power modes (when there is no MCU clock).

46.3.2 Keypad Matrix Construction

The KPP is designed to interface to a keypad matrix, which shorts the intersecting row and column lines together whenever a key is depressed. The interface is not optimized for any other switch configuration.

46.3.3 Keypad Port Configuration

The software must initialize the KPP for the size of the keypad matrix. Pins connected to the keypad columns should be configured as open-drain outputs. Pins connected to the keypad rows should be configured as inputs. On-chip, pull-up resistors should be implemented for active keypad rows.

In addition to enabled row inputs in the Keypad Control register, corresponding interrupt (depress or/and release) must also be enabled to generate an interrupt.

Discrete switches that are not part of the matrix may be connected to any unused row inputs. The second terminal of the discrete switch is connected to ground. The hardware detects closures of these switches without the need for software polling.

46.3.4 Keypad Matrix Scanning

Keypad scanning is performed by a software loop that walks a zero across each of the keypad columns, reading the value on the rows at each step. The process is repeated several times in succession, with the results of each pass optionally compared to those from the previous pass. When several consecutive scans yield the same key closures, a valid key press has been detected. Software then can decode exactly which switch was depressed and pass the value up to the next software layer.

The KPP may be used to control the debounce period, but it must be defined in software by the user's application. The basic period is the period between the scan of two consecutive columns, so the debouncing time between two consecutive scans of the whole matrix shall be the number of columns multiplied by the basic period.

46.3.5 Keypad Standby

There is no need for the ARM platform to continually scan the keypad. Between key presses, the keypad can be left in a state that requires no software intervention until the next key press is detected. To place the keypad in a standby state, software should write all column outputs low. Row inputs are left enabled. At this point, the ARM platform can attend to other tasks or revert to a low power standby mode. The KPP can interrupt the ARM platform if any key is pressed.

Upon receiving a keypad interrupt, the ARM platform should set all the column strobes high, and begin a normal keypad scanning routine to determine which key was pressed. It is important that open-drain drivers be used when scanning to prevent a possible DC path between power and ground through two or more switches.

46.3.6 Glitch Suppression on Keypad Inputs

A glitch suppression circuit qualifies the keypad inputs to prevent noise from inadvertently interrupting the ARM platform. The circuit is a 4-state synchronizer clocked from a low frequency reference clock (ipg_clk_32k) source. This clock must continue to run in any low power mode where the keypad is a wake-up source, as the ARM platform interrupt is generated from the synchronized input. An interrupt is not generated until all four synchronizer stages have latched a valid key assertion. This guarantees the filtering out of any noise less than three clock periods in duration of a low frequency reference clock. Noise filtering of the duration between three to four clock periods cannot be guaranteed. The interrupt output is latched in an S-R latch and remains asserted until cleared by the software. The Set input of the latch is rising-edge clocked. See the figure below.

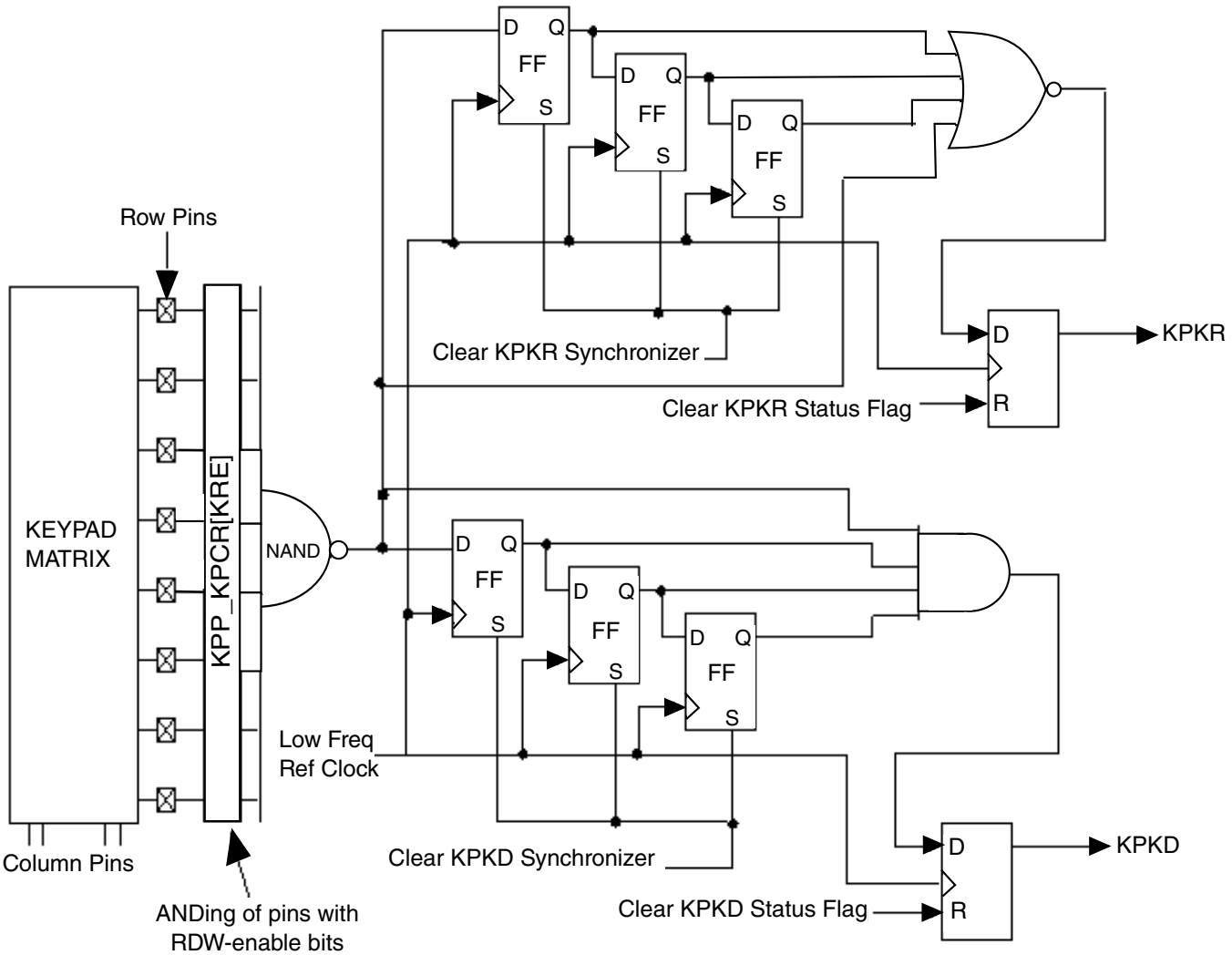


Figure 46-2. Keypad Synchronizer Functional Diagram

46.3.7 Multiple Key Closures

Using the key press and key release interrupts, the software can detect multiple keys or achieve n key rollover. The key scanning routine can be programmed accordingly (See Initialization/Application Information for more information).

The following figures illustrate the interface of a 2-contact keypad matrix with the KPP controller. With proper enabling of row lines and the performing scan-routine, multiple key presses can be detected. When keys present on the same row are pressed, corresponding row lines (multiple lines) become low when the column is driven low during a scan-routine. By reading the data-register, pressed keys can be detected.

Similarly, when keys present on same column line are pressed, the corresponding row line (only one line) becomes low when logic "0" is driven on the column line during a scan-routine.

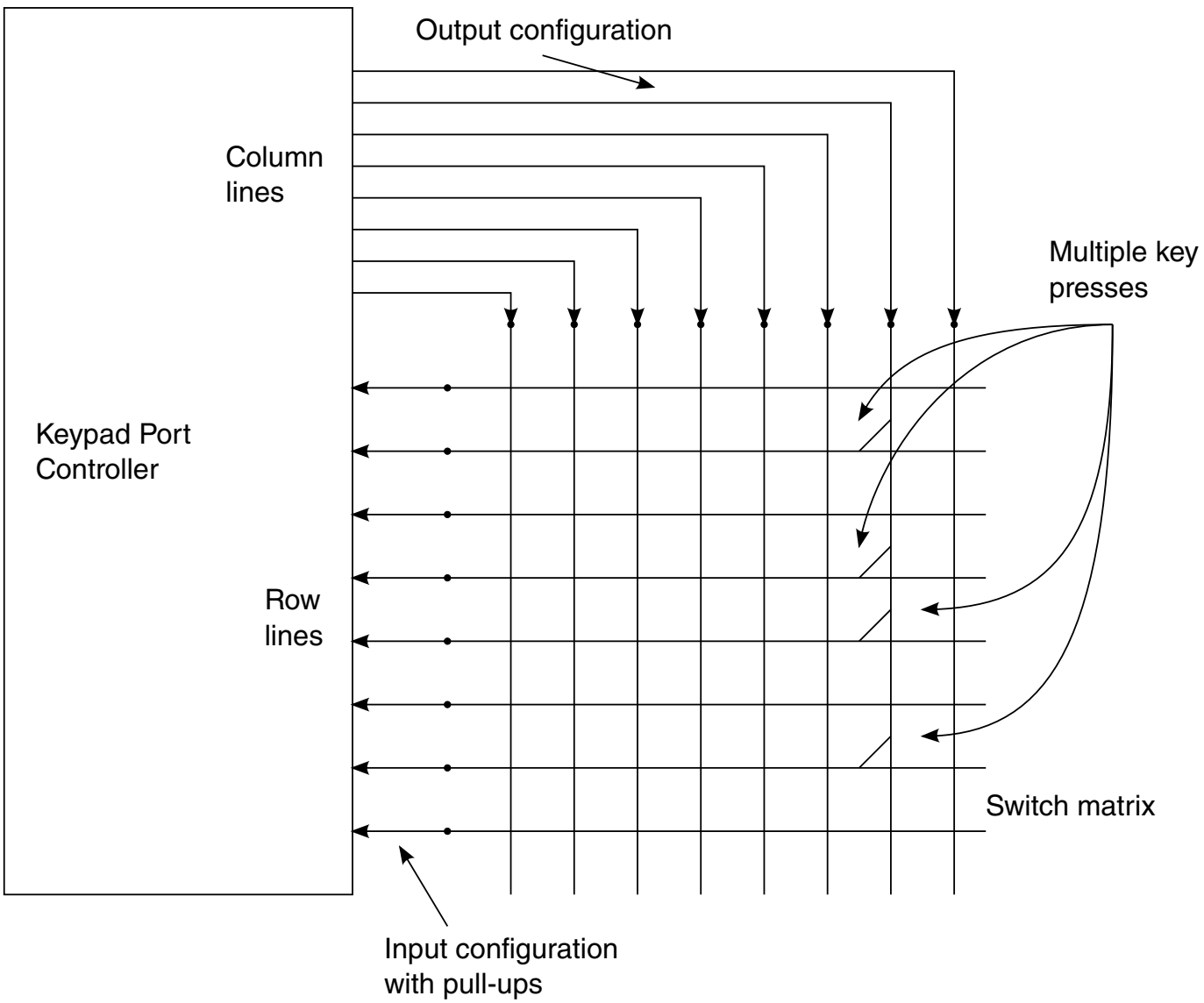


Figure 46-3. Multiple Key Presses on Same Column Line (Simplified View)

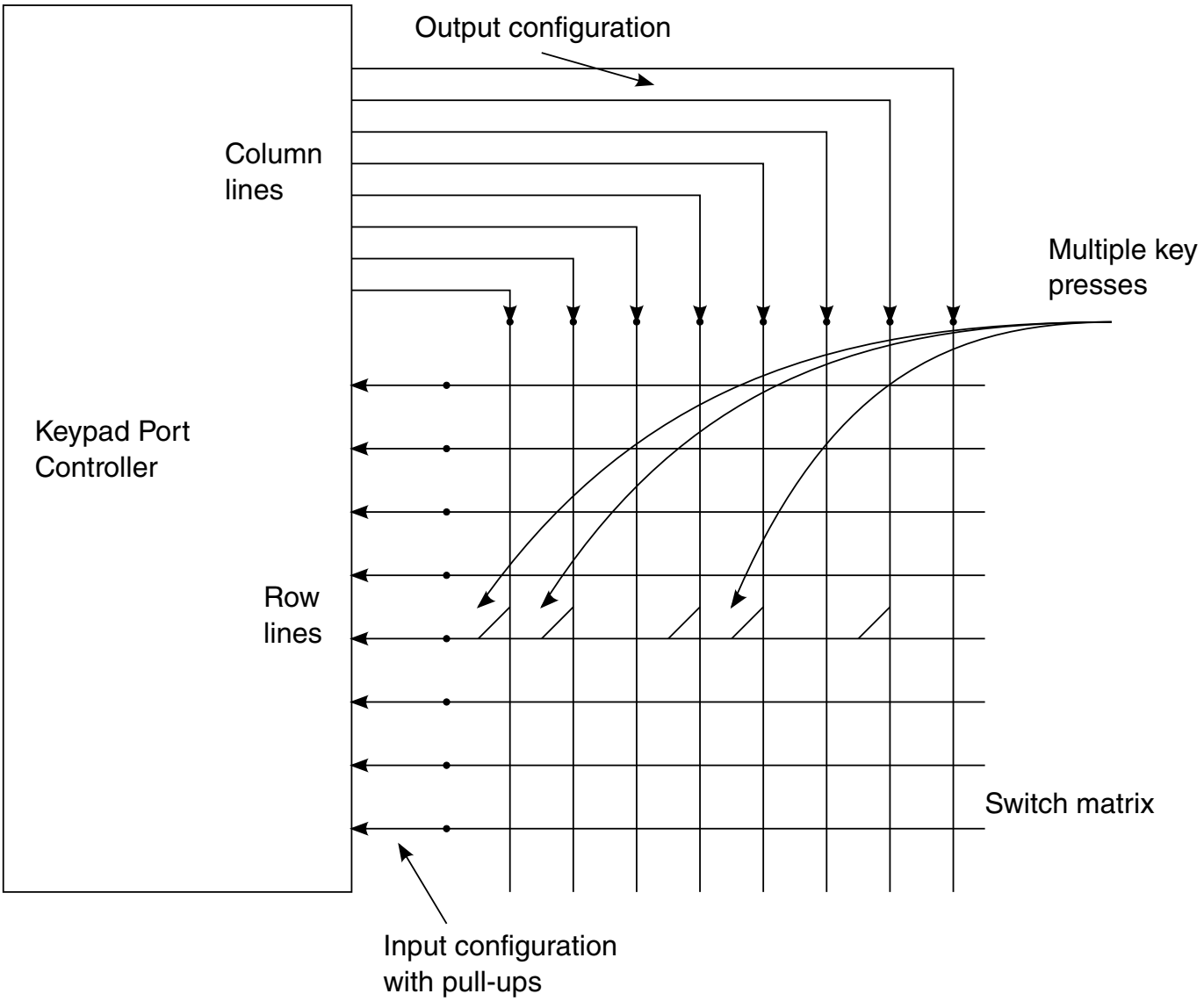


Figure 46-4. Multiple Key Presses on Same Row Line (Simplified View)

NOTE

An n key rollover is a technique with which the system can recognize the order in which keys are pressed.

46.3.7.1 Ghost Key Problem and Correction

The KPP detects if one or multiple keys are pressed or released. In the case where a simple keypad matrix with two-contact switches is used, there is a chance of "ghost" key detection when three or more keys are pressed. This is a limitation imposed by such a keypad matrix.

As seen in Figure 46-5, three keys pressed simultaneously can cause a short between the column currently "scanned" by the software and another column. Depending on the location of the third key pressed, a "ghost" key press may be detected.

However, this can be corrected by using a keypad matrix that provides "ghost" key protection. Such a matrix implements a one-way "diode" at all keypad points between rows and columns. This way, the multiple pressing of three keys will not cause a short at a fourth key (see Figure 46-6).

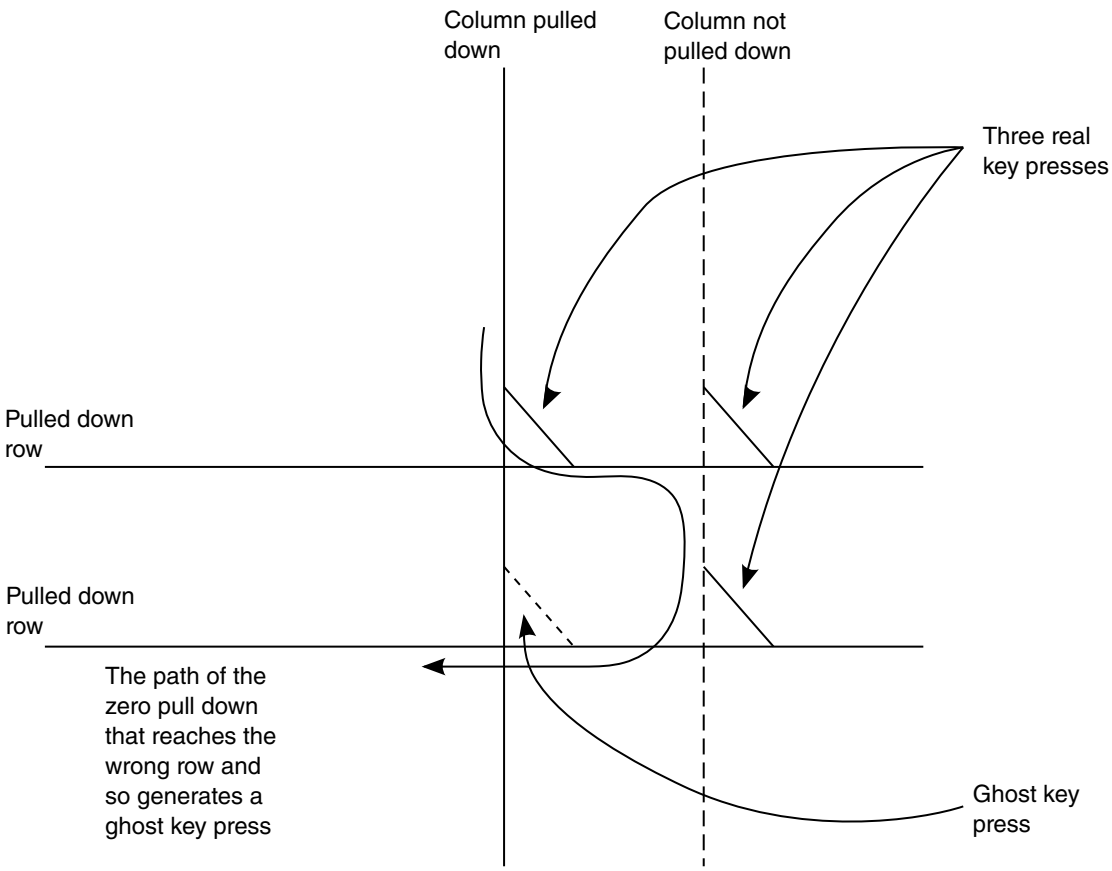


Figure 46-5. Decoding Wrong Three- Key-Presses

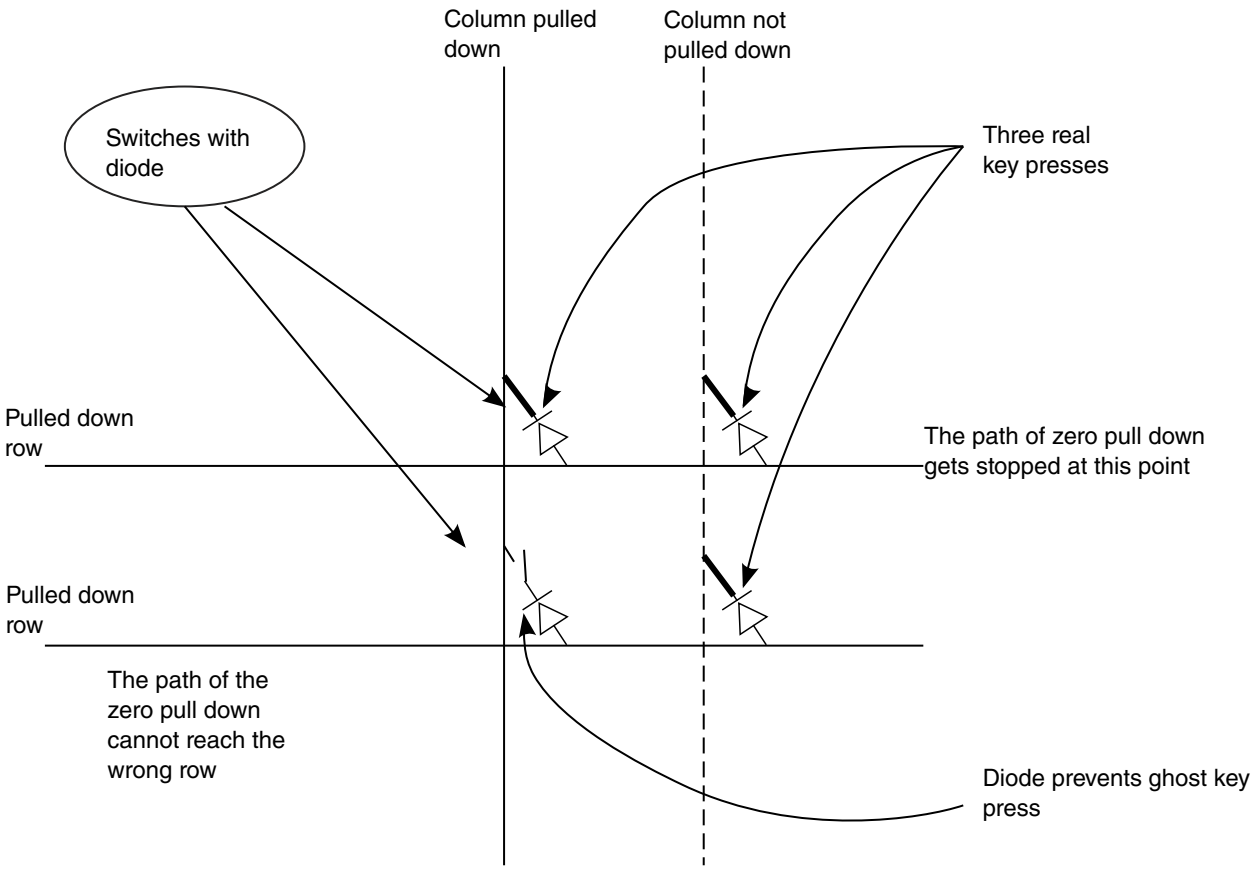


Figure 46-6. Matrix with "Ghost" Key Protections

46.3.8 3-Point Contact Keys Support

The KPP supports interfacing to a matrix consisting of 3-point contact keys. As shown in Figure 46-7, two points of such a key are connected to keypad lines, while a third point is connected to ground (low logic).

The keypad lines should be configured as input and a pull-up should to be present on these lines. When such a key is pressed, corresponding keypad lines go low and an interrupt is generated. There is no need to perform a scanning routine for identification of pressed key as it can be done by reading the keypad data-register. A limitation with such a matrix is that for every key at least one keypad row line should be used.

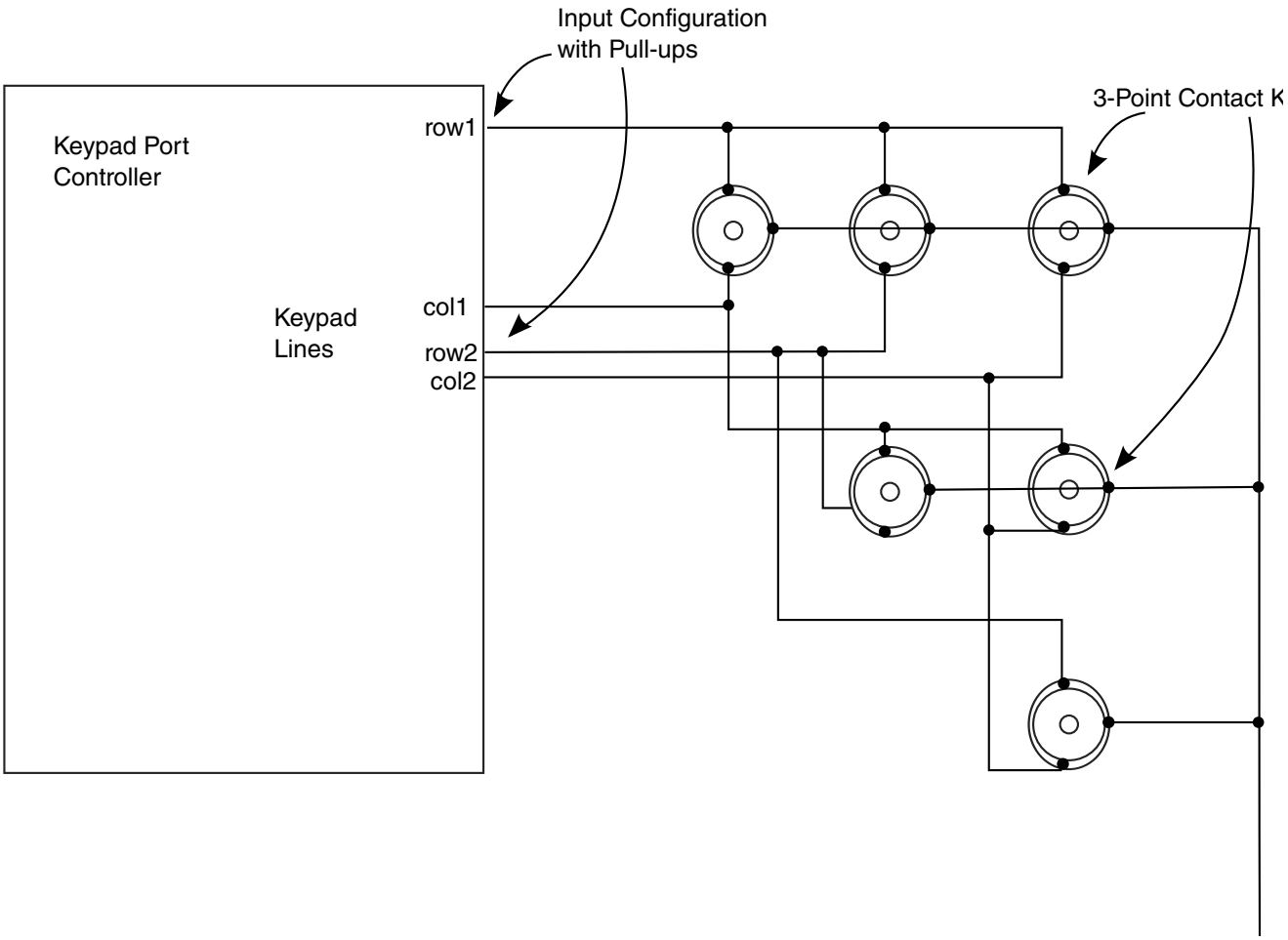


Figure 46-7. KPP Interface with 3-point Contact Key Matrix (Simplified View)

46.3.9 Clocks

The table found here describes the clock sources for KPP. Please see clock control block for clock setting, configuration and gating information.

Table 46-2. KPP Clocks

Clock name	Description
ipg_clk_32k	Low-frequency reference clock (32 kHz)
ipg_clk_s	Peripheral access clock

46.4 External Signals

There are several pins dedicated to the KPP. Keypads of any configuration up to eight rows and eight columns are supported through the software configuration of the peripheral pins. Any pins not used for the keypad are available as general purpose I/O. The registers are configured such that the pins can be treated as an I/O port up to 16 bits wide. Please see IOMUX Controller for KPP pin mux assignments.

See the table below for the list of external signals.

Table 46-3. KPP External Signals

Name	Direction	Function	Reset State	Pull-Up
KEY_COL[7:0]	IO	Column input or output pin, from chip	0	Active[1]
KEY_ROW[7:0]	IO	Row input or output pin, from chip	1	Active[1]

1. The corresponding pads are required to be pull-up enabled.

46.4.1 Input Pins

Any of the 16 pins associated with the KPP can be configured as inputs by writing a "0" to the appropriate bits in the KPP_KDDR. Additionally, the least significant 8 bits (ROW inputs) corresponding to KPP_KDDR[KRDD] have internal pull-ups, which are enabled when the pin is used as an input.

46.4.2 Output Pins

Any of the 16 pins associated with the KPP can be configured as outputs by writing the appropriate bits in the KPP_KDDR to a "1". Additionally, the 8 most significant bits (15-8) can be designated as open drain outputs by writing a "1" to the appropriate bits in the KPP_KPCR. The lower 8 bits (7-0) are always in "totem pole" style, driven when configured as outputs.

See the table below.

Table 46-4. Keypad Port Column Modes

KPP_KDDR (15:8)	KPP_KPCR (15:8)	Pin Function
0	x	Input
1	0	Totem-Pole Output
1	1	Open-Drain Output

NOTE

Totem pole configuration helps for a faster discharge of keypad capacitance when all columns need to be quickly brought to a "1" during the scan routine. With this configuration, delay between the scanning of two subsequent columns is reduced. Totem pole mode is to be used temporarily on the column pins to discharge the lines only, but should be switched to open-drain mode for normal operation.

46.4.3 Generation of Transfer Error Signal on Peripheral Bus

If there is an access to an address which is not implemented, then the KPP asserts a transfer error signal on Peripheral Bus.

46.5 Initialization/Application Information

46.5.1 Typical Keypad Configuration and Scanning Sequence

Perform the following steps to configure the keypad:

1. Enable the number of rows in the keypad (KPP_KPCR[KRE]).
2. Write 0s to KPP_KPDR[KCD].
3. Configure the keypad columns as open-drain (KPP_KPCR[KCO]).
4. Configure columns as output (KPP_KDDR[KCDD]) and rows as input (KPP_KDDR[KRDD]).
5. Clear the KPKD Status Flag and Synchronizer chain.
6. Set the KDIE control bit, and clear the KRIE control bit (avoid false release events).
7. (The system is now in standby mode, and awaiting a key press.)

46.5.2 Key Press Interrupt Scanning Sequence

Perform the following steps to perform a keypad scanning routine:

1. Disable both (depress and release) keypad interrupts.
2. Write 1s to KPP_KPDR[KCD], setting column data to 1s.
3. Configure columns as totem pole outputs (for quick discharging of keypad capacitance).

4. Configure columns as open-drain.
5. Write a single column to 0, and other columns to 1.
6. Sample row inputs and save data. Multiple key presses can be detected on a single column.
7. Repeat Steps 2-6 for remaining columns.
8. Return all columns to 0 in preparation for standby mode.
9. Clear KPKD and KPKR status bit(s) by writing to a "1"; set the KPKR synchronizer chain by writing a "1" to the KPP_KRSS register; and clear the KPKD synchronizer chain by writing a "1" to the KDSC register.
10. Re-enable the appropriate keypad interrupt(s) so that the KDIE detects a key hold condition, or the KRIE detects a key-release event.

46.5.3 Additional Comments

The order of key press detection can be done in software only. Therefore, the software may need to run the scan routines at very short intervals of time per the application's demands. The reason that such functionality cannot be put in the KPP is that the block is limited by the number of external pins.

For the keys that require a very precise order (such as game keys), individual GPIO pins may be more useful.

46.6 Memory Map and Register Definition

46.6.1 KPP register descriptions

The KPP contains four registers.

46.6.1.1 KPP memory map

KPP base address: 401F_C000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Keypad Control Register (KPCR)	16	RW	0000h
2h	Keypad Status Register (KPSR)	16	RW	0400h
4h	Keypad Data Direction Register (KDDR)	16	RW	0000h
6h	Keypad Data Register (KPDR)	16	RW	0000h

46.6.1.2 Keypad Control Register (KPCR)

46.6.1.2.1 Offset

Register	Offset
KPCR	0h

46.6.1.2.2 Function

The Keypad Control Register determines which of the eight possible column strobes are to be open drain when configured as outputs, and which of the eight row sense lines are considered in generating an interrupt to the core.

It is up to the programmer to ensure that pins being used for functions other than the keypad are properly disabled. The KPP_KPCR register is byte- or half-word-addressable.

46.6.1.2.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	KCO								KRE							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

46.6.1.2.4 Fields

Field	Function
15-8 KCO	KCO Keypad Column Strobe Open-Drain Enable. Setting a column open-drain enable bit (KCO7-KCO0) disables the pull-up driver on that pin. Clearing the bit allows the pin to drive to the high state. This bit has no effect when the pin is configured as an input.

Table continues on the next page...

Field	Function
	NOTE: Configuration of external port control logic (for example, IOMUX) should be done properly so that the KPP controls an open-drain enable of the pin. 00000000b - Column strobe output is totem pole drive. 00000001b - Column strobe output is open drain.
7-0 KRE	KRE Keypad Row Enable. Setting a row enable control bit in this register enables the corresponding row line to participate in interrupt generation. Likewise, clearing a bit disables that row from being used to generate an interrupt. This register is cleared by a reset, disabling all rows. The row-enable logic is independent of the programmed direction of the pin. Writing a "0" to the data register of the pins configured as outputs will cause a keypad interrupt to be generated if the row enable associated with that bit is set. 00000000b - Row is not included in the keypad key press detect. 00000001b - Row is included in the keypad key press detect.

46.6.1.3 Keypad Status Register (KPSR)

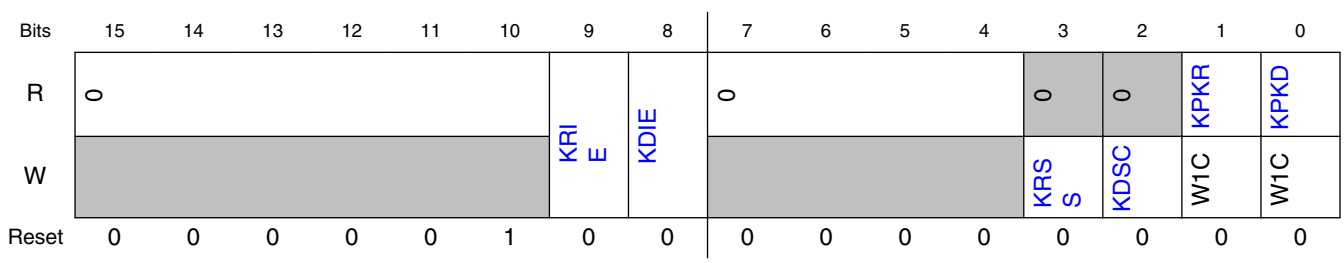
46.6.1.3.1 Offset

Register	Offset
KPSR	2h

46.6.1.3.2 Function

The Keypad Status Register reflects the state of the key press detect circuit. The KPP_KPSR register is byte- or half-word-addressable.

46.6.1.3.3 Diagram



46.6.1.3.4 Fields

Field	Function
15-10 —	- Reserved
9 KRIE	KRIE Keypad Release Interrupt Enable. The software should ensure that the interrupt for a Key Release event is masked until it has entered the key pressed state, and vice versa, unless this activity is desired (as might be the case when a repeated interrupt is to be generated). The synchronizer chains are capable of being initialized to detect repeated key presses or releases. If they are not initialized when the corresponding event flag is cleared, false interrupts may be generated for depress (or release) events shorter than the length of the corresponding chain. 0b - No interrupt request is generated when KPKR is set. 1b - An interrupt request is generated when KPKR is set.
8 KDIE	KDIE Keypad Key Depress Interrupt Enable. Software should ensure that the interrupt for a Key Release event is masked until it has entered the key pressed state, and vice-versa, unless this activity is desired (as might be the case when a repeated interrupt is to be generated). The synchronizer chains are capable of being initialized to detect repeated key presses or releases. If they are not initialized when the corresponding event flag is cleared, false interrupts may be generated for depress (or release) events shorter than the length of the corresponding chain. 0b - No interrupt request is generated when KPKD is set. 1b - An interrupt request is generated when KPKD is set.
7-4 —	- Reserved, should be cleared
3 KRSS	KRSS Key Release Synchronizer Set. Self-clear bit. The Key release synchronizer is set by writing a logic one into this bit. Reads return a value of "0". 0b - No effect 1b - Set bits which sets keypad release synchronizer chain
2 KDSC	KDSC Key Depress Synchronizer Clear. Self-clear bit. The Key depress synchronizer is cleared by writing a logic "1" into this bit. Reads return a value of "0". 0b - No effect 1b - Set bits that clear the keypad depress synchronizer chain
1 KPKR	KPKR Keypad Key Release. The keypad key release (KPKR) status bit is set when all enabled rows are detected high after synchronization (the KPKR status bit will be set when cleared by a reset). The KPKR bit may be used to generate a maskable key release interrupt. The key release synchronizer may be set high by software after scanning the keypad to ensure a known state. Due to the logic function of the release and depress synchronizer chains, it is possible to see the re-assertion of a status flag (KPKD or KPKR) if it is cleared by software prior to the system exiting the state it represents. Reset value of register is "0" as long as reset is asserted. However when reset is de-asserted, the value of the register depends upon the external row pins and can become "1". 0b - No key release detected 1b - All keys have been released

Table continues on the next page...

Field	Function
0	KPKD
KPKD	Keypad Key Depress. The keypad key depress (KPKD) status bit is set when one or more enabled rows are detected low after synchronization. The KPKD status bit remains set until cleared by the software. The KPKD bit may be used to generate a maskable key depress interrupt. If desired, the software may clear the key press synchronizer chain to allow a repeated interrupt to be generated while a key remains pressed. In this case, a new interrupt will be generated after the synchronizer delay (4 cycles of the low frequency reference clock (ipg_clk_32k) elapses if a key remains pressed. This functionality can be used to detect a long key press. This allows detection of additional key presses of the same key or other keys. Due to the logic function of the release and depress synchronizer chains, it is possible to see the re-assertion of a status flag (KPKD or KPKR) if it is cleared by the software prior to the system exiting the state it represents. 0b - No key presses detected 1b - A key has been depressed

46.6.1.4 Keypad Data Direction Register (KDDR)

46.6.1.4.1 Offset

Register	Offset
KDDR	4h

46.6.1.4.2 Function

The bits in the KPP_KDDR control the direction of the keypad port pins. The upper eight bits in the register affect the pins designated as column strobes, while the lower eight bits affect the row sense pins. Setting any bit in this register configures the corresponding pin as an output. Clearing any bit in this register configures the corresponding port pin as an input. For the Keypad Row DDR, an internal pull-up is enabled if the corresponding bit is clear. This register is cleared by a reset, configuring all pins as inputs. The KPP_KDDR register is byte- or half-word addressable.

NOTE

When a pin is used as row pin for keypad purposes, all corresponding pull-ups should be enabled at the upper level (for example, IOMUX) when the bit in KRDD is cleared.

46.6.1.4.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	KCDD								KRDD							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

46.6.1.4.4 Fields

Field	Function
15-8 KCDD	KCDD Keypad Column Data Direction Register. Setting a bit configures the corresponding COL*n* pin as an output (where *n* = 7 through 0). 00000000b - COLn pin is configured as an input. 00000001b - COLn pin is configured as an output.
7-0 KRDD	KRDD Keypad Row Data Direction. Setting a bit configures the corresponding ROW*n* pin as an output (where *n* = 7 through 0). 00000000b - ROWn pin configured as an input. 00000001b - ROWn pin configured as an output.

46.6.1.5 Keypad Data Register (KPDR)

46.6.1.5.1 Offset

Register	Offset
KPDR	6h

46.6.1.5.2 Function

This 16-bit register is used to access the column and row data. Data written to this register is stored in an internal latch, and for each pin configured as an output, the stored data is driven onto the pin. A read of this register returns the value on the pin for those bits configured as inputs. Otherwise, the value read is the value stored in the register.

The KPP_KPDR register is byte- or half-word addressable. This register is not initialized by a reset. Valid data should be written to this register before any bits are configured as outputs.

46.6.1.5.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	KCD								KRD							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

46.6.1.5.4 Fields

Field	Function
15-8 KCD	KCD Keypad Column Data. A read of these bits returns the value on the pin for those bits configured as inputs. Otherwise, the value read is the value stored in the register. 0 Read/Write "0" from/to column ports 1 Read/Write "1" from/to column ports
7-0 KRD	KRD Keypad Row Data. A read of these bits returns the value on the pin for those bits configured as inputs. Otherwise, the value read is the value stored in the register. 0 Read/Write "0" from/to row ports 1 Read/Write "1" from/to row ports

Chapter 47
Low Power Inter-Integrated Circuit (LPI2C)

47.1 Chip-specific LPI2C information

Table 47-1. Reference links to related information

Topic	Related module(s)	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
Interrupts, DMA Events and XBAR Assignments	-	Interrupts, DMA Events and XBAR Assignments

NOTE

For LPI2C on this device, only LPI2C1 fully supports the 5 wires, while other instances (LPI2C2, ...) have only 2 wires.

47.2 Overview

The LPI2C is a low power Inter-Integrated Circuit (I2C) module that supports an efficient interface to an I^2C bus as a master and/or as a slave.

- Implements logic support for standard-mode, fast-mode, fast-mode plus and ultra-fast modes of operation.
- Uses little CPU overhead, with DMA offloading of FIFO register accesses.
- Continues operating in stop modes if an appropriate clock is available.

The LPI2C module also complies with the System Management Bus (SMBus) Specification, version 3. The SMBus is a single-ended simple two-wire bus, which is typically used for low bandwidth communications.

NOTE

The I^2C (Inter-Integrated Circuit) serial bus is multi-master, multi-slave, packet-switched, and single-ended, and is often used to attach microcontroller ICs to lower-speed peripheral ICs.

47.2.1 Block diagram

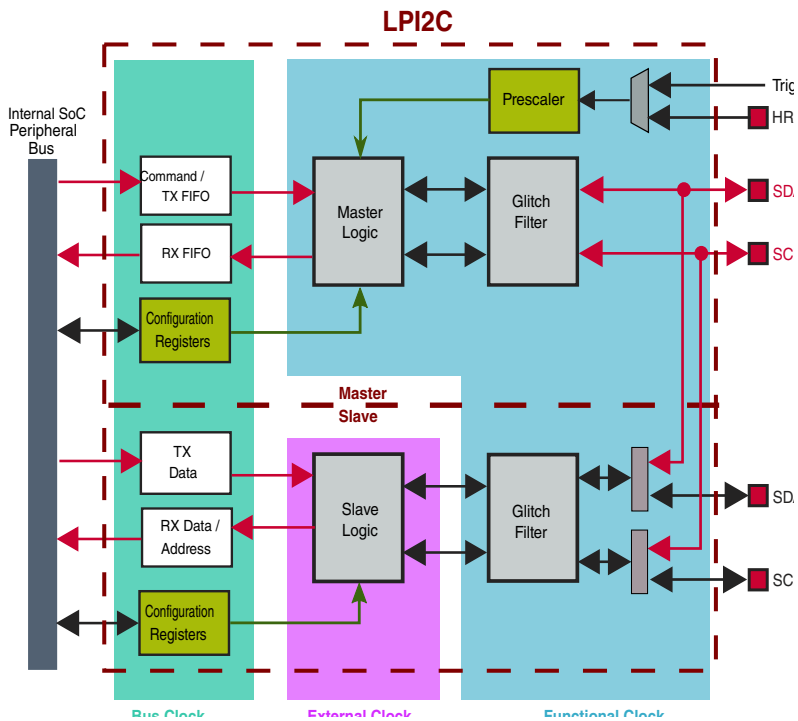


Figure 47-1. LPI2C block diagram

47.2.2 Features

The LPI2C supports:

- Standard, Fast, Fast+ and Ultra Fast modes are supported
- High speed mode (HS) in slave mode
- High speed mode (HS) in master mode, if SCL pin implements current source pull-up (device-specific)

- Multi-master support, including synchronization and arbitration. Multi-master means any number of master nodes can be present. Additionally, master and slave roles may be changed between messages (after a STOP is sent).
- Clock stretching: Sometimes multiple I2C nodes may be driving the lines at the same time. If any I2C node is driving a line low, then that line will be low. I2C nodes that are starting to transmit a logical one (by letting the line float high) can detect that the line is low, and thereby know that another I2C node is active at the same time.
  - When node detection is used on the SCL line, it is called *clock stretching*, and clock stretching is used as a I2C flow control mechanism.
  - When node detection is used on the SDA line, it is called *arbitration*, and arbitration ensures that there is only one I2C node transmitter at a time.
- General call, 7-bit and 10-bit addressing
- Software reset, START byte and Device ID (also require software support)

The LPI2C master supports:

- Command/transmit FIFO of 4 words (8-bit transmit data + 3-bit command).
- Receive FIFO of 4 words (8-bit receive data).
- Command FIFO will wait for idle I2C bus before initiating transfer
- Command FIFO can initiate (repeated) START and STOP conditions and one or more master-receiver transfers
- STOP condition can be generated from command FIFO, or generated automatically when the transmit FIFO is empty
- Host request input to control the start time of an I2C bus transfer
- Flexible receive data match can generate interrupt on data match and/or discard unwanted data
- Flag and optional interrupt to signal Repeated START condition, STOP condition, loss of arbitration, unexpected NACK, and command word errors
- Supports configurable bus idle timeout and pin-stuck-low timeout

The LPI2C slave supports:

- Separate I2C slave registers to minimize software overhead because of master/slave switching
- Support for 7-bit or 10-bit addressing, address range, SMBus alert and general call address
- Transmit data register that supports interrupt or DMA requests
- Receive data register that supports interrupt or DMA requests
- Software-controllable ACK or NACK, with optional clock stretching on ACK/NACK bit
- Configurable clock stretching, to avoid transmit FIFO underrun and receive FIFO overrun errors
- Flag and optional interrupt at end of packet, STOP condition, or bit error detection

47.3 Functional description

47.3.1 Master Mode

The LPI2C master logic operates independently from the slave logic to perform all master mode transfers on the I2C bus.

47.3.1.1 Transmit and Command FIFO commands

The transmit FIFO stores command data to initiate the various I2C operations. The following operations can be initiated through commands in the transmit FIFO:

- START or Repeated START condition with address byte and expecting ACK or NACK.
- Transmit data (this is the default for zero extended byte writes to the transmit FIFO).
- Receive 1-256 bytes of data (can also be configured to discard receive data and not store in receive FIFO).
- STOP condition (can also be configured to send STOP condition when transmit FIFO is empty).

Multiple transmit and receive commands can be inserted between the START condition and STOP conditon; transmit and receive commands must not be interleaved (to comply with the I2C specification). The receive data command and the receive data and discard commands can be interleaved, to ensure that only the desired received data is stored in the receive FIFO (or compared with the data match logic).

The LPI2C master automatically transmits a NACK on the last byte of a receive data command unless the next command in the FIFO is also a receive data command. A NACK is also automatically transmitted if the transmit FIFO is empty when a receive data command completes.

The LPI2C master supports 10-bit addressing through a (repeated) START condition, followed by a transmit data byte containing the second address byte, followed by any number of data bytes with the master-transmit data.

A START or Repeated START condition that is expecting a NACK (for example,HS-mode master code) must be followed by a STOP or (repeated) START condition.

47.3.1.2 Master operations

Whenever the LPI2C is enabled, it monitors the I2C bus to detect when the I2C bus is idle (MSR[BBF]). The I2C bus is no longer considered idle if either SCL or SDA are low, and the I2C bus becomes idle if a STOP condition is detected or if a bus idle timeout is detected (as configured by MCFGR2[BUSIDLE]). After the I2C bus is idle, the transmit FIFO is not empty, and the host request is either asserted or disabled, then the LPI2C master initiates a transfer on the I2C bus. This involves the following steps:

- Wait the bus idle time equal to (MCCR0[CLKLO] + 1) multiplied by the prescaler (MCFGR1[PRESCALE]).
- Transmit a START condition and address byte using the timing configuration in Master Clock Configuration 0 (MCCR0); if a high speed mode transfer is configured, then the timing configuration from Master Clock Configuration 1 (MCCR1) is used instead.
- Perform master-transmit or master-receive transfers, as configured by the transmit FIFO.
- Transmit NACK on the last byte of a master-receive transfer, unless the next command in the transmit FIFO is also a receive data command and the transmit FIFO is not empty.
- Transmit a Repeated START or STOP condition as configured by the transmit FIFO and/or MCFGR1[AUTOSTOP]. A repeated START can change which timing configuration register is used.

When the LPI2C master is disabled (either due to MCR[MEN] being clear or automatically due to mode entry), the LPI2C continues to empty the transmit FIFO until a STOP condition is transmitted. However, the LPI2C will no longer stall the I2C bus waiting for the transmit or receive FIFO, and after the transmit FIFO is empty, the LPI2C generates a STOP condition automatically.

The LPI2C master can stall the I2C bus under certain conditions; this results in SCL pulled low continuously on the first bit of a byte, until the condition is removed:

- LPI2C master is enabled and busy, the transmit FIFO is empty, and MCFGR1[AUTOSTOP] is clear.
- LPI2C master is enabled and receiving data, receive data is not being discarded (due to command or receive data match), and the receive FIFO is full.

47.3.1.3 Receive FIFO and Data Matching

The receive FIFO is used to store receive data during master-receiver transfers. Receive data can also be configured to discard receive data instead of storing in the receive FIFO; this is configured by the command word in the transmit FIFO.

Receive data supports a receive data match function that can match received data against one of two bytes or against a masked data byte. The data match function can also be configured to compare only the first one or two received data words since the last (repeated) START condition. Receive data that is already discarded due to the command word cannot cause the data match to set, and delaya the match on the first received data word until after the discarded data is received. The receiver match function can also be configured to discard all receive data until a data match is detected, using the MCFGR0[RDMO] control bit. When clearing the MCFGR0[RDMO] control bit following a data match, clear MCFGR0[RDMO] before clearing MSR[DMF], to allow all subsequent data to be received.

47.3.1.4 Timing Parameters

The following timing parameters can be configured by the LPI2C master. Parameters are configured separately for high speed mode (Master Clock Configuration 1 (MCCR1)) and other modes (Master Clock Configuration 0 (MCCR0)). This allows the high speed mode master code to be sent using the regular timing parameters, and then switch to the high speed mode timing (following a repeated START) until the next STOP condition.

The LPI2C master timing parameters in LPI2C functional clock cycles are configured as follows. They must be configured to meet the I2C timing specification for the required mode.

Table 47-2. Timing Parameters

I2C Specification Timing Parameter	I2C Specification Timing Symbol	LPI2C Timing Parameter (LPI2C functional clock cycles)
SCL clock period	tSCL	(CLKHI + CLKLO + 2 + SCL_LATENCY) x (2 ^ PRESCALE)
hold time (repeated) START condition	tHD:STA	(SETHOLD +1) x (2 ^ PRESCALE)
LOW period of the SCL clock	tLOW	(CLKLO + 1) x (2 ^ PRESCALE)
HIGH period of the SCL clock	tHIGH	(CLKHI + 1 + SCL_LATENCY) x (2 ^ PRESCALE)
setup time for a repeated START condition or STOP condition	tSU:STA, tSU:STO	(SETHOLD + 1 + SCL_LATENCY) x (2 ^ PRESCALE)
data hold time	tHD:DAT	(DATAVD + 1) x (2 ^ PRESCALE)
data setup time	tSU:DAT	(SDA_LATENCY + 1) x (2 ^ PRESCALE)

Table continues on the next page...

Table 47-2. Timing Parameters (continued)

I2C Specification Timing Parameter	I2C Specification Timing Symbol	LPI2C Timing Parameter (LPI2C functional clock cycles)
bus free time between a STOP and START condition	tBUF	(CLKLO + 1 + SDA_LATENCY) x (2 ^ PRESCALE)
data valid time, data valid acknowledge time	tVD:DAT, tVD:ACK	(DATAVD + 1) x (2 ^ PRESCALE)

The latency parameters are defined in the following table, these parameters assume the risetime is less than one LPI2C functional clock cycle. The risetime depends on a number of factors, including the I/O propagation delay, the I2C bus loading and the external pull-up resistor sizing. A larger rise time increasea the number of cycles that the signal takes to propagate through the synchronizer (and glitch filter), which increases the latency.

Table 47-3. Synchronization Latency

Timing Parameter	Timing Definition
SCL_LATENCY	ROUNDDOWN ((2 + FILTSCL + SCL_RISETIME) / (2 ^ PRESCALE))
SDA_LATENCY	ROUNDDOWN ((2 + FILTSDA + SDA_RISETIME) / (2 ^ PRESCALE))

The following timing restrictions must be enforced to avoid unexpected START or STOP conditions on the I2C bus, or to avoid unexpected START or STOP conditions detected by the LPI2C master. The timing restrictions can be summarized as **SDA cannot change when SCL is high outside of a transmitted (repeated) START or STOP condition**.

Table 47-4. LPI2C Timing Parameter Restrictions

Timing Parameter	Minimum	Maximum	Comment
CLKLO	0x03	-	Also: CLKLO x (2 ^ PRESCALE) > SCL_LATENCY
CLKHI	0x01	-	Configure CLKHI to meet the duty cycle requirements in the I2C specification
SETHOLD	0x02	-	Also: SETHOLD x (2 ^ PRESCALE) > SDA_LATENCY
DATAVD	0x01	CLKLO - SDA_LATENCY - 1	Configure DATAVD to meet the data hold requirement in the I2C specification
FILTSCL	0x00	[CLKLO x (2 ^ PRESCALE)] - 3	FILTSCL and FILTSDA are the only parameters not multiplied by (2 ^ PRESCALE)
FILTSDA	FILTSCL	[CLKLO x (2 ^ PRESCALE)] - 3	Configuring FILTSDA greater than FILTSCL can delay the SDA input to compensate for board level skew
BUSIDLE	(CLKLO+SETHOLD+2) x 2	-	Must also be greater than (CLKHI+1)

The timing parameters must be configured to meet the requirements of the I2C specification; this depends on the mode being supported and the LPI2C functional clock frequency. When switching between two modes using the different clock configuration registers (for example, Fast and HS-mode), the PRESCALE factor must remain constant between the modes. Some example timing configurations are provided below.

Table 47-5. LPI2C Example Timing Configurations

I2C Mode	Clock Frequency	Baud Rate	PRESCALE	FILTSCL / FILTSDA	SETHOLD	CLKLO	CLKHI	DATAVD
Fast	8 MHz	400 kbps	0x0	0x0/0x0	0x04	0x0B	0x05	0x02
Fast+	8 MHz	1 Mbps	0x0	0x0/0x0	0x02	0x03	0x01	0x01
Fast	48 MHz	400 kbps	0x0	0x1/0x1	0x1D	0x3E	0x35	0x0F
Fast	48 MHz	400 kbps	0x2	0x1/0x1	0x07	0x11	0x0B	0x03
Fast+	48 MHz	1 Mbps	0x2	0x1/0x1	0x03	0x06	0x04	0x04
HS-mode	48 MHz	3.2 Mbps	0x0	0x0/0x0	0x07	0x08	0x03	0x01
Fast	60 MHz	400 kbps	0x1	0x2/0x2	0x11	0x28	0x1F	0x08
Fast+	60 MHz	1 Mbps	0x1	0x2/0x2	0x07	0x0F	0x0B	0x01
HS-mode	60 MHz	3.33 Mbps	0x1	0x0/0x0	0x04	0x04	0x02	0x01

47.3.1.5 Error Conditions

The LPI2C master monitors for errors while it is active, the following conditions generates an error flag and block a new START condition from being sent, until the flag is cleared by software:

- A START or STOP condition is detected and is not generated by the LPI2C master (sets MSR[ALF]).
- Transmitting data on SDA and different values are being received (sets MSR[ALF]).
- NACK is detected when transmitting data, and MCFGR1[IGNACK] is clear (sets MSR[NDF]).
- NACK is detected and is expecting ACK for the address byte, and MCFGR1[IGNACK] is clear (sets MSR[NDF]).
- ACK is detected and is expecting NACK for the address byte, and MCFGR1[IGNACK] is clear (sets MSR[NDF]).
- Transmit FIFO is requesting to transmit or receive data without a START condition (sets MSR[FEF]).
- SCL (or SDA if MCFGR1[TIMECFG] is set) is low for (MCFGR2[TIMELOW] * 256) prescaler cycles without a pin transition (sets MSR[PLTF]).

Software must respond to the MSR[PLTF] flag to terminate the existing command either cleanly (by clearing MCR[MEN]), or abruptly (by setting MCR[RST]).

The MCFGR2[BUSIDLE] field can be used to force the I2C bus to be considered idle when SCL and SDA remain high for (BUSIDLE+1) prescaler cycles. The I2C bus is normally considered idle when the LPI2C master is first enabled, but when BUSIDLE is configured greater than zero then SCL and/or SDA must be high for (BUSIDLE+1) prescaler cycles before the I2C bus is first considered idle.

47.3.1.6 Pin Configuration

- **Open-drain support:** The LPI2C master defaults to open-drain configuration of the SDA and SCL pins. Support for true open drain depends on the specific device, and requires the pins where LPI2C pins are muxed to support true open drain.
- **High Speed mode support:** Support for high speed mode also depends on the specific device, and requires the SCL pin to support the current source pull-up required in the I2C specification.
- **Ultra-Fast mode support:** The LPI2C master also supports the output-only push-pull function required for I2C ultra-fast mode using the SDA and SCL pins. Support for ultra-fast mode also requires the MCFGR1[IGNACK] bit to be set.
- **Push-pull 2-wire support:** A push-pull 2-wire configuration is also available to the LPI2C master that may support a partial high speed mode, if the LPI2C is the only master and all I2C pins on the bus are at the same voltage. This configures the SCL pin as push-pull for every clock except the 9th clock pulse, to allow high speed mode compatible slaves to perform clock stretching. In this mode, the SDA pin is tristated for master-receive data bits and master-transmit ACK/NACK bits, and is configured as push-pull at other times. To avoid the risk of contention when SDA is push-pull, the pin can be configured for open-drain operation, as part of the device-specific configuration.
- **Push-pull 4-wire support:** The push-pull 4-wire configuration separates the SCL input data and output data into separate pins, and separates the SDA input data and output data into separate pins. The SCL/SDA pins are used for input data; the SCLS/SDAS pins are used for output data, with configurable polarity. This simplifies external connections when connecting the LPI2C to the I2C bus through external level shifters or discrete components. When using this 4-wire configuration, the LPI2C master logic and LPI2C slave logic are not able to connect to separate I2C buses.

47.3.2 Slave Mode

To perform all slave mode transfers on the I2C bus, the LPI2C slave logic operates independently from the LPI2C master logic.

47.3.2.1 Address Matching

The LPI2C slave can be configured:

- To match one of two addresses, using either 7-bit or 10-bit addressing modes for each address
- To match a range of addresses in either 7-bit or 10-bit addressing modes
- To match the General Call Address, and generate appropriate flags
- To match the SMBus Alert Address, and generate appropriate flags
- To detect the high speed mode master code, and to disable the digital filters and output valid delay time until the next STOP condition is detected

After a valid address is matched, the LPI2C slave automatically performs slave-transmit or slave-receive transfers until:

- A NACK is detected (unless MCFGR1[IGNACK] is set)
- A bit error is detected (the LPI2C slave is driving SDA, but a different value is sampled)
- A (repeated) START or STOP condition is detected

47.3.2.2 Transmit and Receive Data

- The Transmit and Receive Data registers are double-buffered and only update during a slave-transmit and slave-receive transfer, respectively.
- The slave address *that was received* can be configured to be read from either the Receive Data register (for example, when using DMA to transfer data), or from the Address Status register.
- The Transmit Data register can be configured to only request data after a slave-transmit transfer is detected, or to request new data whenever the Transmit Data register is empty.
- The Transmit Data register should only be written when the Transmit Data flag is set.
- The Receive Data register should only be read when the Received Data flag is set (or the Address Valid flag is set and SCFGR1[RXCFG] = 1).
- The Address Status register should only be read when the Address Valid flag is set.

47.3.2.3 Clock Stretching

The LPI2C slave supports many configurable options for when clock stretching is performed. The following conditions can be configured to perform clock stretching:

- During the 9th clock pulse of the address byte and the Address Valid flag is set.

- During the 9th clock pulse of a slave-transmit transfer and the Transmit Data flag is set.
- During the 9th clock pulse of a slave-receive transfer and the Receive Data flag is set.
- During the 8th clock pulse of an address byte or a slave-receive transfer and the Transmit ACK flag is set. In high speed mode, this is disabled.
- Clock stretching can also be extended for CLKHOLD cycles, to allow additional setup time to sample the SDA pin externally. In high speed mode, this is disabled.

Unless extended by the CLKHOLD configuration, clock stretching extends for one peripheral bus clock cycle after SDA updates when clock stretching is enabled.

47.3.2.4 Timing Parameters

The LPI2C slave can configure the following timing parameters. These parameters are disabled when SCR[FILTEN] is clear, when SCR[FILTDZ] is set in Doze mode, and when LPI2C slave detects high speed mode. When disabled, the LPI2C slave is clocked directly from the I2C bus, and may not satisfy all timing requirements of the I2C specification (such as SDA minimum hold time in Standard/Fast mode).

- SDA data valid time from SCL negation to SDA update
- SCL hold time when clock stretching is enabled to increase setup time when sampling SDA externally
- SCL glitch filter time
- SDA glitch filter time

The LPI2C slave imposes the following restrictions on the timing parameters.

- FILTSDA must be configured to greater than or equal to FILTSCL (unless compensating for board level skew between SDA and SCL).
- DATAVD must be configured less than the minimum SCL low period.

47.3.2.5 Error Conditions

The LPI2C slave can detect the following error conditions:

- Bit error flag sets when the LPI2C slave is driving SDA, but samples a different value than what is expected.

- FIFO error flag sets due to a transmit data underrun or a receive data overrun. To eliminate the possibility of underrun and overrun occurring, enable clock stretching.
- FIFO error flag also sets due to an address overrun when SCFGR1[RXCFG] is set, otherwise an address overrun is not flagged. To eliminate the possibility of overrun occurring, enable clock stretching.

The LPI2C slave does not implement a timeout due to SCL and/or SDA being stuck low. If this detection is required, then the LPI2C master logic should be used and so software can reset the LPI2C slave when this condition is detected.

47.3.3 Low power modes

Table 47-6. LPI2C low power modes

Mode	LPI2C Operation
Run	Normal operations
Stop	Can continue operating in stop mode if the Doze Enable bit is clear (MCR[DOZEN] = 0) and LPI2C uses an external or internal clock source that remains operating during stop mode.

47.3.4 Debug mode

Table 47-7. LPI2C Debug mode

Mode	LPI2C Operation
Debug	Can continue operating in Debug mode if the Debug Enable bit is set (MCR[DBGEN] = 1).

47.3.5 Interrupts and DMA requests

Depending on the specific device, interrupts and DMA requests can be combined in some ways:

- The LPI2C master and slave interrupts may be combined
- The LPI2C master and slave transmit DMA requests may be combined
- The LPI2C master and slave receive DMA requests may be combined

47.3.5.1 Master mode

The next table lists the master mode sources that can generate LPI2C master interrupts and LPI2C master transmit/receive DMA requests.

Table 47-8. Master Interrupts and DMA Requests

Master Status (MSR)		Description	Can generate		
Flag	Name		Interrupt?	DMA Request?	Low Power Wakeup?
TDF	Transmit Data Flag	Data can be written to Transmit FIFO, as configured by the Transmit FIFO Watermark MFCR[TXWATER]	Y	TX	Y
RDF	Receive Data Flag	Data can be read from the Receive FIFO, as configured by the Receive FIFO Watermark MFCR[RXWATER]	Y	RX	Y
EPF	End Packet Flag	Master has transmitted a Repeated START or STOP condition	Y	N	Y
SDF	STOP Detect Flag	Master has transmitted a STOP condition	Y	N	Y
NDF	NACK Detect Flag	• During an address byte, the master expected an ACK but detected a NACK • During an address byte, the master expected a NACK but detected an ACK • During a master-transmitter data byte, the master detected a NACK	Y	N	Y
ALF	Arbitration Lost Flag	• The master lost arbitration due to a START/STOP condition detected at the wrong time • Or the master was transmitting data but received different data than the data that was transmitted	Y	N	Y
FEF	FIFO Error Flag	The master is expecting a START condition in the Command FIFO, but the next entry in the Command FIFO is not a START condition	Y	N	Y
PLTF	Pin Low Timeout Flag	Pin low timeout is enabled and SCL (or SDA if configured) is low for longer than the configured timeout	Y	N	Y
DMF	Data Match Flag	The received data matches the configured data match, but the received data is not discarded due to a command FIFO entry	Y	N	Y
MBF	Master Busy Flag	LPI2C master is busy transmitting/receiving data	N	N	N
BBF	Bus Busy Flag	LPI2C master is enabled and activity is detected on the I2C bus, but a STOP condition has not been detected and a bus idle timeout (if enabled) has not occurred.	N	N	N

47.3.5.2 Slave mode

The next table lists the slave mode sources that can generate LPI2C slave interrupts and the LPI2C slave transmit/receive DMA requests.

Table 47-9. Slave Interrupts and DMA Requests

Slave Status (SSR)		Description	Can generate		
Flag	Name		Interrupt?	DMA Request?	Low Power Wakeup?
TDF	Transmit Data Flag	Data can be written to the Slave Transmit Data (STDR)	Y	TX	Y
RDF	Receive Data Flag	Data can be read from the Slave Receive Data (SRDR)	Y	RX	Y
AVF	Address Valid Flag	Address can be read from the Slave Address Status (SASR)	Y	RX	Y
TAF	Transmit ACK Flag	ACK/NACK can be written to the Slave Transmit ACK (STAR)	Y	N	Y
RSF	Repeated Start Flag	Slave has detected an address match followed by a Repeated START condition	Y	N	Y
SDF	STOP Detect Flag	Slave has detected an address match followed by a STOP condition	Y	N	Y
BEF	Bit Error Flag	Slave was transmitting data, but received different data than what was transmitted	Y	N	Y
FEF	FIFO Error Flag	• Transmit data underrun • Receive data overrun • Address status overrun (when Receive Data Configuration SCFGR1[RXCFG] = 1). FEF flag can only set when clock stretching is disabled.	Y	N	Y
AM0F	Address Match 0 Flag	Slave detected an address match with SAMR[ADDR0] field	Y	N	N
AM1F	Address Match 1 Flag	Slave detected an address match with SAMR[ADDR1] field or using an address range	Y	N	N
GCF	General Call Flag	Slave detected an address match with the General Call address	Y	N	N
SARF	SMBus Alert Response Flag	Slave detected an address match with the SMBus Alert address	Y	N	N
SBF	Slave Busy Flag	LPI2C slave is busy receiving an address byte or is transmitting/receiving data	N	N	N
BBF	Bus Busy Flag	LPI2C slave is enabled and a START condition is detected on I2C bus, but a STOP condition has not been detected	N	N	N

47.3.6 Clocks

Table 47-10. LPI2C clocks

LPI2C Functional clock	The LPI2C functional clock is asynchronous to the bus clock and can remain enabled in low power modes to support I2C bus transfers by the LPI2C master. The functional clock is also used by the LPI2C slave to support digital filter and data hold time configurations. The LPI2C master divides the functional clock by a prescaler and the resulting frequency must be at least 8 times faster than the I2C bus bandwidth.
External clock	The LPI2C slave logic is clocked directly from the external pins SCL and SDA (or SCLS and SDAS if master and slave are implemented on separate pins). This allows the LPI2C slave to remain operational, even when the LPI2C functional clock is disabled. **NOTE:** The LPI2C slave digital filter must be disabled if the LPI2C functional clock is disabled, and this can affect compliance with some of the timing parameters of the I2C specification, such as the data hold time.
Bus clock	The bus clock is only used for bus accesses to the control and configuration registers. The bus clock frequency must be sufficient to support the data bandwidth requirements of the LPI2C master and slave registers.

For device-specific clocking information, refer to the Clocking chapter.

47.3.7 Resets

Table 47-11. LPI2C Resets

Chip reset	The logic and registers for the LPI2C master and slave are reset to their default state on a chip reset.
Software reset	• The LPI2C master implements a software reset bit in its Control Register. The MCR[RST] bit resets all master logic and registers to their default state, except for the MCR register itself. • The LPI2C slave implements a software reset bit in its Control Register. The SCR[RST] bit resets all slave logic and registers to their default state, except for the SCR register itself.
FIFO reset	• The LPI2C master implements write-only control bits that reset the transmit FIFO (MCR[RTF]) and receive FIFO (MCR[RRF]). After a FIFO is reset, that FIFO is empty. • The LPI2C slave implements write-only control bits that reset the transmit data register (SCR[RTF] and receive data register (SCR[RRF]). After a data register is reset, that data register is empty.

47.3.8 Peripheral Triggers

The connection of the LPI2C peripheral triggers to other peripherals depend upon the specific device being used.

Table 47-12. LPI2C Triggers

Trigger	Description
Master Output Trigger	The LPI2C master generates an output trigger that can be connected to other peripherals on the device. The master output trigger asserts on both a Repeated START or STOP condition, and the master output trigger remains asserted for one cycle of the LPI2C functional clock divided by the prescaler.
Slave Output Trigger	The LPI2C slave generates an output trigger that can be connected to other peripherals on the device. The slave output trigger asserts on both a Repeated START or STOP condition that occurs after a slave address match, and the slave output trigger remains asserted until the next slave SCL pin negation.
Input Trigger	To control the start of a LPI2C bus transfer, the LPI2C input trigger can be selected instead of the HREQ input. The input trigger is synchronized and to be detected, the input trigger must assert for at least 2 cycles of the LPI2C functional clock divided by the PRESCALE configuration. When the LPI2C is busy, the HREQ input (and therefore the input trigger) is ignored.

47.4 Signal descriptions

Table 47-13. Signals

Signal	Name	2-Wire Scheme	4-Wire Scheme	I/O
SCL	LPI2C clock line	SCL	In 4-wire mode, this is the SCL input pin.	I/O
SDA	LPI2C data line	SDA	In 4-wire mode, this is the SDA input pin.	I/O
SCLS	Secondary I2C clock line	Not used	In 4-wire mode, this is the SCLS output pin. If LPI2C master/slave are configured to use separate pins, then this is the LPI2C slave SCL pin.	I/O
SDAS	Secondary I2C data line	Not used	In 4-wire mode, this is the SDAS output pin. If LPI2C master/slave are configured to use separate pins, then this is the LPI2C slave SDA pin.	I/O
HREQ	Host request	If host request is asserted and the I2C bus is idle, then it initiates an LPI2C master transfer. HREQ is an additional pin separate from the 2-wire or 4-wire scheme.		I

The I^2C 2-wire serial bus diagram shows the 2 signal connection.

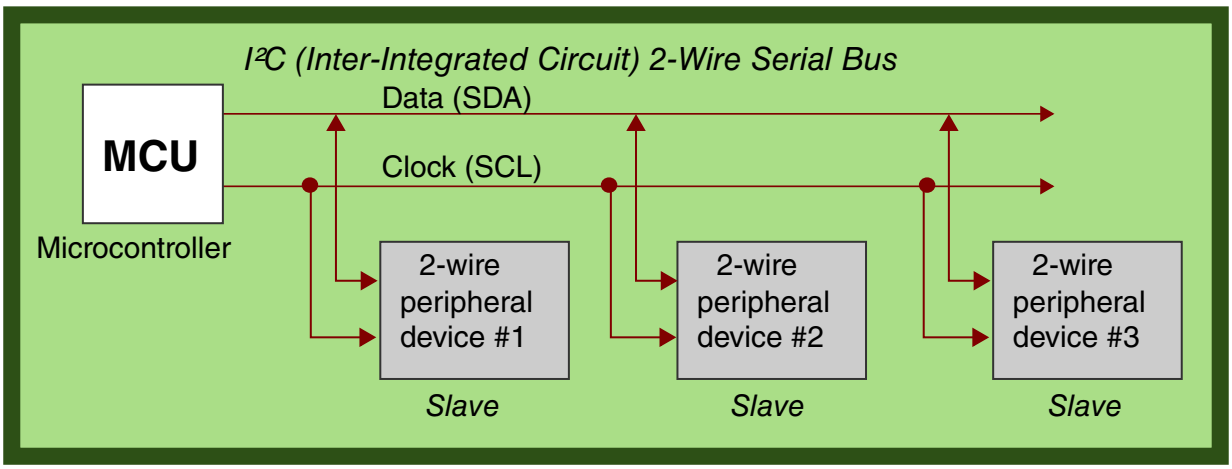


Figure 47-2. I²C 2-wire serial bus

The I²C 4-wire serial bus diagram shows a possible 4 signal connection.

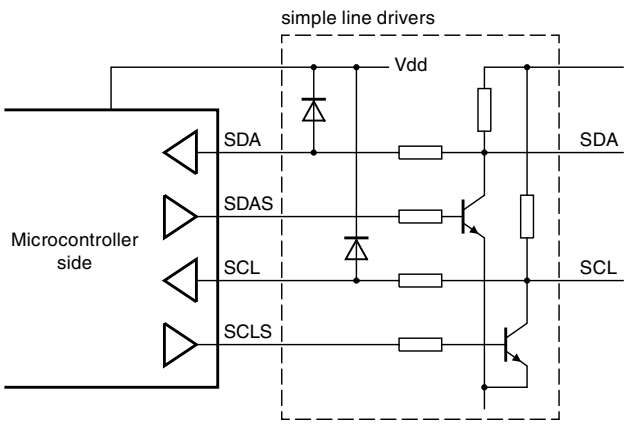


Figure 47-3. I²C 4-wire serial bus

47.5 Memory Map and Registers

NOTE

Writing a Read-Only (RO) register or reading a Write-Only (WO) register can cause bus errors. This module will not check if programmed values in the registers are correct; the application software must ensure that valid programmed values are being written.

47.5.1 LPI2C register descriptions

47.5.1.1 LPI2C memory map

LPI2C1 base address: 403F_0000h Wire

LPI2C2 base address: 403F_4000h Wire3 on MM

LPI2C3 base address: 403F_8000h Wire1 (Wire2 on MM)

LPI2C4 base address: 403F_C000h Wire2 (Wire1 on MM)

Offset	Register	Width (In bits)	Access	Reset value
0h	Version ID (VERID)	32	RO	0100_0003h
4h	Parameter (PARAM)	32	RO	0000_0202h
10h	Master Control (MCR)	32	RW	0000_0000h
14h	Master Status (MSR)	32	W1C	0000_0001h
18h	Master Interrupt Enable (MIER)	32	RW	0000_0000h
1Ch	Master DMA Enable (MDER)	32	RW	0000_0000h
20h	Master Configuration 0 (MCFGR0)	32	RW	0000_0000h
24h	Master Configuration 1 (MCFGR1)	32	RW	0000_0000h
28h	Master Configuration 2 (MCFGR2)	32	RW	0000_0000h
2Ch	Master Configuration 3 (MCFGR3)	32	RW	0000_0000h
40h	Master Data Match (MDMR)	32	RW	0000_0000h
48h	Master Clock Configuration 0 (MCCR0)	32	RW	0000_0000h
50h	Master Clock Configuration 1 (MCCR1)	32	RW	0000_0000h
58h	Master FIFO Control (MFCR)	32	RW	0000_0000h
5Ch	Master FIFO Status (MFSR)	32	RO	0000_0000h
60h	Master Transmit Data (MTDR)	32	WO	0000_0000h
70h	Master Receive Data (MRDR)	32	RO	0000_4000h
110h	Slave Control (SCR)	32	RW	0000_0000h
114h	Slave Status (SSR)	32	W1C	0000_0000h
118h	Slave Interrupt Enable (SIER)	32	RW	0000_0000h
11Ch	Slave DMA Enable (SDER)	32	RW	0000_0000h
124h	Slave Configuration 1 (SCFGR1)	32	RW	0000_0000h
128h	Slave Configuration 2 (SCFGR2)	32	RW	0000_0000h
140h	Slave Address Match (SAMR)	32	RW	0000_0000h
150h	Slave Address Status (SASR)	32	RO	0000_4000h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
154h	Slave Transmit ACK (STAR)	32	RW	0000_0000h
160h	Slave Transmit Data (STDR)	32	WO	0000_0000h
170h	Slave Receive Data (SRDR)	32	RO	0000_4000h

47.5.1.2 Version ID (VERID)

47.5.1.2.1 Offset

Register	Offset
VERID	0h

47.5.1.2.2 Function

Contains version numbers for the module design and feature set.

47.5.1.2.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	MAJOR								MINOR							
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FEATURE															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

47.5.1.2.4 Fields

Field	Function
31-24 MAJOR	Major Version Number Returns the major version number for the module design specification. Read-only field.
23-16 MINOR	Minor Version Number Returns the minor version number for the module design specification. Read-only field.

Table continues on the next page...

Field	Function
15-0 FEATURE	Feature Specification Number Returns the feature set number. Read-only field. 0000000000000010b - Master only, with standard feature set 0000000000000011b - Master and slave, with standard feature set

47.5.1.3 Parameter (PARAM)

47.5.1.3.1 Offset

Register	Offset
PARAM	4h

47.5.1.3.2 Function

Contains parameter values that were implemented in the module.

47.5.1.3.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0				MRXFIFO				0				MTXFIFO			
W																
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0

47.5.1.3.4 Fields

Field	Function
31-16 —	Reserved
15-12 —	Reserved
11-8	Master Receive FIFO Size

Table continues on the next page...

Field	Function
MRXFIFO	Configures the number of words in the master receive FIFO to $2^{MRXFIFO}$
7-4 —	Reserved
3-0 MTXFIFO	Master Transmit FIFO Size Configures the number of words in the master transmit FIFO to $2^{MTXFIFO}$

47.5.1.4 Master Control (MCR)

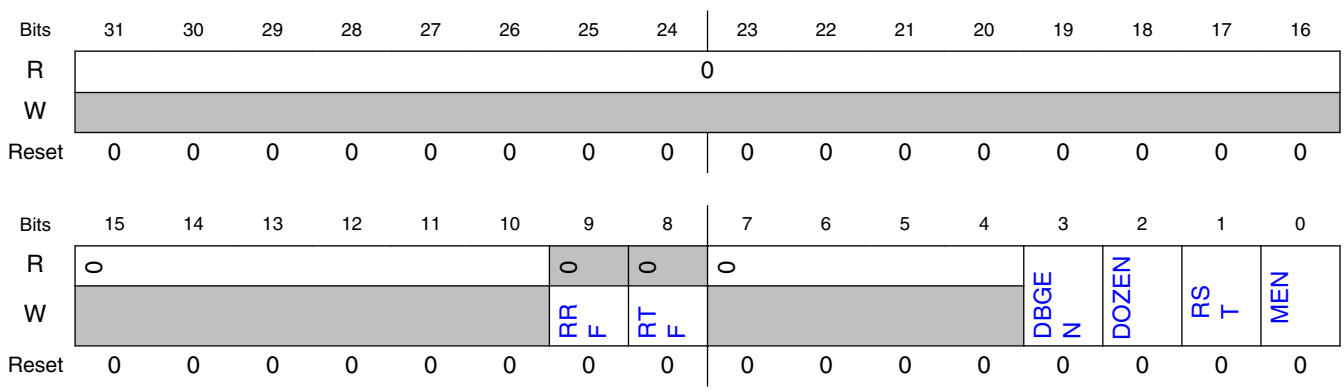
47.5.1.4.1 Offset

Register	Offset
MCR	10h

47.5.1.4.2 Function

The MCR register contains resets, debug enable, and other master control settings.

47.5.1.4.3 Diagram



47.5.1.4.4 Fields

Field	Function
31-10 —	Reserved
9	Reset Receive FIFO

Table continues on the next page...

Field	Function
RRF	Resets the Receive FIFO. 0b - No effect 1b - Receive FIFO is reset
8 RTF	Reset Transmit FIFO Resets the Transmit FIFO. 0b - No effect 1b - Transmit FIFO is reset
7-4 —	Reserved
3 DBGEN	Debug Enable Enables the Master in Debug mode. 0b - Master is disabled in debug mode 1b - Master is enabled in debug mode
2 DOZEN	Doze mode enable Enables or disables the master in Doze mode. 0b - Master is enabled in Doze mode 1b - Master is disabled in Doze mode
1 RST	Software Reset Resets all internal master logic and registers, except the Master Control (MCR) register. RST remains set until cleared by software. The reset takes effect immediately and remains asserted until negated by software. There is no minimum delay required before clearing the software reset. 0b - Master logic is not reset 1b - Master logic is reset
0 MEN	Master Enable Enables the Master logic. 0b - Master logic is disabled 1b - Master logic is enabled

47.5.1.5 Master Status (MSR)

47.5.1.5.1 Offset

Register	Offset
MSR	14h

47.5.1.5.2 Function

MSR contains status flags for transmit and receive data; Start and Stop conditions; and bus and master busy or idle status.

47.5.1.5.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0						BBF	MBF	0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	DMF	PLTF	FEF	ALF	NDF	SDF	EPF	0						RDF	TDF
W		W1C	W1C	W1C	W1C	W1C	W1C	W1C								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

47.5.1.5.4 Fields

Field	Function
31-26 —	Reserved
25 BBF	Bus Busy Flag Indicates that the I2C bus is busy. 0b - I2C Bus is idle 1b - I2C Bus is busy
24 MBF	Master Busy Flag Indicates that the I2C Master is busy. 0b - I2C Master is idle 1b - I2C Master is busy
23-16 —	Reserved
15 —	Reserved
14 DMF	Data Match Flag Indicates that the received data has matched the MDMR[MATCH0] and/or MDMR[MATCH1] fields (as configured by MCFGR1[MATCFG]. Received data *that is discarded due to CMD field* does not cause DMF to set. 0b - Have not received matching data 1b - Have received matching data
13 PLTF	Pin Low Timeout Flag Sets when the SCL and/or SDA input is low for more than PINLOW cycles (Pin Low Timeout, MCFGR3[PINLOW]), even when the LPI2C master is idle. • Software is responsible for resolving the pin low condition. • PLTF cannot be cleared as long as the pin low timeout continues. • Before the LPI2C can initiate a START condition, the PLTF must be cleared. 0b - Pin low timeout has not occurred or is disabled 1b - Pin low timeout has occurred

Table continues on the next page...

Field	Function
12 FEF	FIFO Error Flag Detects an attempt to send or receive data without first generating a (repeated) START condition. This can occur if the transmit FIFO underflows when MCFGR1[AUTOSTOP] = 1. When FEF is set, the LPI2C master sends a STOP condition (if busy), and does not initiate a new START condition until FEF has been cleared. 0b - No error 1b - Master sending or receiving data without a START condition
11 ALF	Arbitration Lost Flag Set if either of these conditions exist: • The LPI2C master transmits a logic one and detects a logic zero on the I2C bus • The LPI2C master detects a START or STOP condition while the LPI2C master is transmitting data When ALF sets, the LPI2C master releases the I2C bus (goes idle), and the LPI2C master does not initiate a new START condition until the ALF has been cleared. 0b - Master has not lost arbitration 1b - Master has lost arbitration
10 NDF	NACK Detect Flag Set if the LPI2C master detects a NACK it was not expecting when transmitting an address or data. When set, the master does not initiate a new START condition until NDF has been cleared. If a NACK is expected for a given address (as configured by the command word), then the NDF sets if a NACK is not generated. When NDF is set, the LPI2C master automatically transmits a STOP condition if MCFGR1[AUTOSTOP] = 1, or the transmit FIFO is not empty. 0b - Unexpected NACK was not detected 1b - Unexpected NACK was detected
9 SDF	STOP Detect Flag Set when the LPI2C master generates a STOP condition. 0b - Master has not generated a STOP condition 1b - Master has generated a STOP condition
8 EPF	End Packet Flag Set when the LPI2C master generates either a repeated START condition or a STOP condition. Does not set when the master first generates a START condition. 0b - Master has not generated a STOP or Repeated START condition 1b - Master has generated a STOP or Repeated START condition
7-2 —	Reserved
1 RDF	Receive Data Flag RDF sets whenever the number of words in the receive FIFO is greater than MFCR[RXWATER]. 0b - Receive Data is not ready 1b - Receive data is ready
0 TDF	Transmit Data Flag TDF sets whenever the number of words in the transmit FIFO is equal or less than MFCR[TXWATER]. 0b - Transmit data is not requested 1b - Transmit data is requested

47.5.1.6 Master Interrupt Enable (MIER)

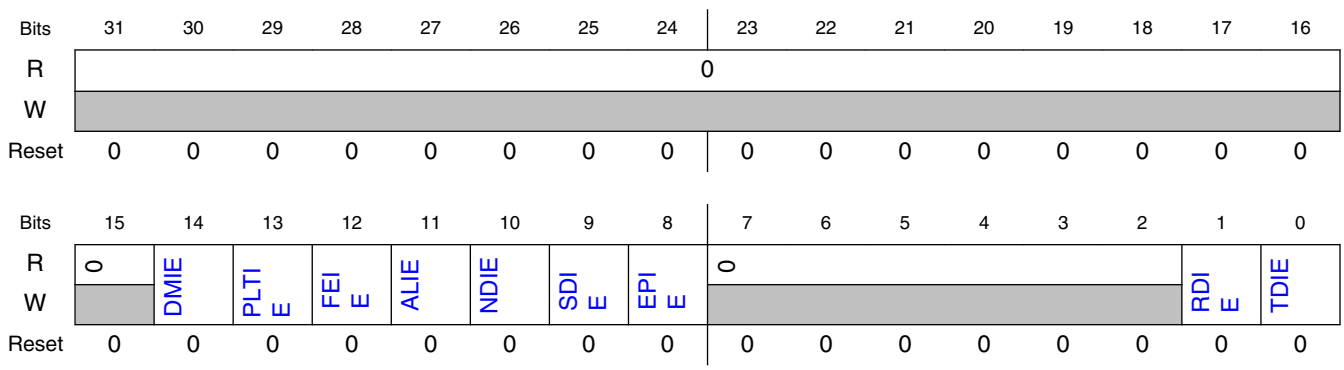
47.5.1.6.1 Offset

Register	Offset
MIER	18h

47.5.1.6.2 Function

MIER contains transmit and receive data interrupt enables; Start, Stop, and Nack detect interrupt enables; and DMA interrupt enables.

47.5.1.6.3 Diagram



47.5.1.6.4 Fields

Field	Function
31-16 —	Reserved
15 —	Reserved
14 DMIE	Data Match Interrupt Enable Enables interrupt for data match. 0b - Disabled 1b - Enabled
13 PLTIE	Pin Low Timeout Interrupt Enable Enables interrupt for pin low timeout. 0b - Disabled 1b - Enabled
12	FIFO Error Interrupt Enable

Table continues on the next page...

Field	Function
FEIE	Enables interrupt for FIFO error. 0b - Enabled 1b - Disabled
11 ALIE	Arbitration Lost Interrupt Enable Enables interrupt for arbitration lost. 0b - Disabled 1b - Enabled
10 NDIE	NACK Detect Interrupt Enable Enables interrupt for NACK detect. 0b - Disabled 1b - Enabled
9 SDIE	STOP Detect Interrupt Enable Enables interrupt for STOP detect. 0b - Disabled 1b - Enabled
8 EPIE	End Packet Interrupt Enable Enables interrupt for end packet. 0b - Disabled 1b - Enabled
7-2 —	Reserved
1 RDIE	Receive Data Interrupt Enable Enables interrupt for receive data. 0b - Disabled 1b - Enabled
0 TDIE	Transmit Data Interrupt Enable Enables interrupt for transmit data. 0b - Disabled 1b - Enabled

47.5.1.7 Master DMA Enable (MDER)

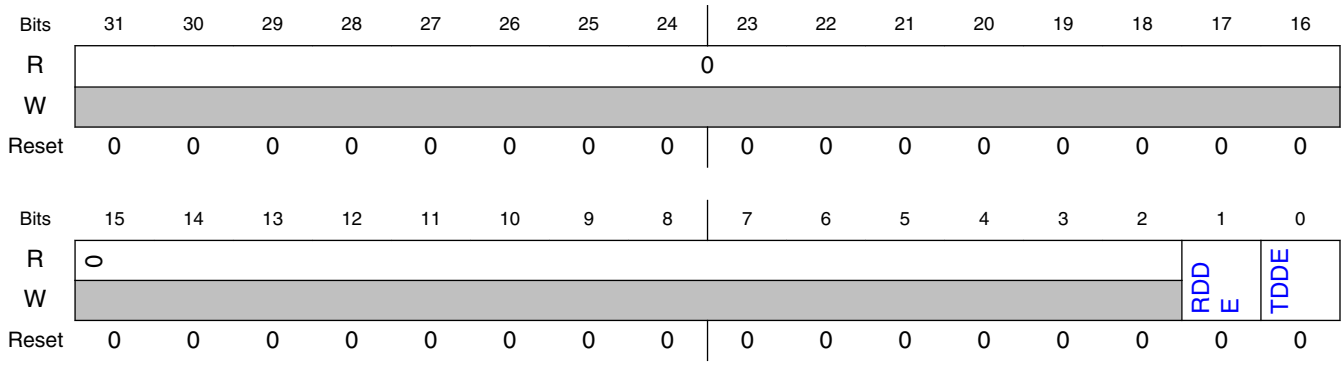
47.5.1.7.1 Offset

Register	Offset
MDER	1Ch

47.5.1.7.2 Function

MDER contains DMA transmit, request, and receive enables.

47.5.1.7.3 Diagram



47.5.1.7.4 Fields

Field	Function
31-2 —	Reserved
1 RDDE	Receive Data DMA Enable Enables DMA receive data. 0b - DMA request is disabled 1b - DMA request is enabled
0 TDDE	Transmit Data DMA Enable Enables DMA transmit data. 0b - DMA request is disabled 1b - DMA request is enabled

47.5.1.8 Master Configuration 0 (MCFGR0)

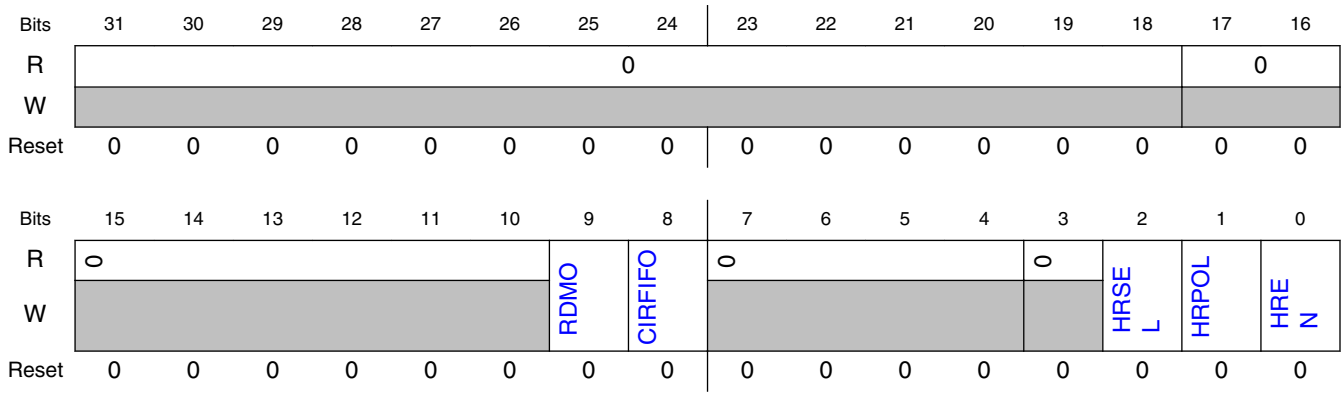
47.5.1.8.1 Offset

Register	Offset
MCFGR0	20h

47.5.1.8.2 Function

MCFGR0 contains host settings and other receive and transfer settings.

47.5.1.8.3 Diagram



47.5.1.8.4 Fields

Field	Function
31-18 —	Reserved
17-16 —	Reserved
15-10 —	Reserved
9 RDMO	Receive Data Match Only When enabled, all received data that does not cause the Data Match Flag (MSR[DMF]) to set is discarded. After MSR[DMF] is set, the RDMO configuration is ignored. When disabling RDMO, clear RDMO before clearing MSR[DMF], to ensure that no receive data is lost. 0b - Received data is stored in the receive FIFO 1b - Received data is discarded unless the the Data Match Flag (MSR[DMF]) is set
8 CIRFIFO	Circular FIFO Enable When enabled, the transmit FIFO read pointer is saved to a temporary register. The transmit FIFO empties as normal, but after the LPI2C master is idle and the transmit FIFO is empty, then the read pointer value will be restored from the temporary register. This causes the contents of the transmit FIFO to be cycled through repeatedly. If MCFGR1[AUTOSTOP] is set, then a STOP condition is sent whenever the transmit FIFO is empty and the read pointer is restored. 0b - Circular FIFO is disabled 1b - Circular FIFO is enabled
7-4 —	Reserved
3 —	Reserved
2 HRSEL	Host Request Select Selects the source of the host request input. When host request is enabled, HRSEL should remain static (the Host Request Select should not change). 0b - Host request input is pin HREQ 1b - Host request input is input trigger

Table continues on the next page...

Field	Function
1 HRPOL	Host Request Polarity Configures the polarity of the host request pin. When host request is enabled, HRPOL should remain static (Host Request Polarity should not change). 0b - Active low 1b - Active high
0 HREN	Host Request Enable When enabled, the LPI2C master only initiates a START condition if the host request input is asserted and the bus is idle. A repeated START is not affected by the host request. 0b - Host request input is disabled 1b - Host request input is enabled

47.5.1.9 Master Configuration 1 (MCFGR1)

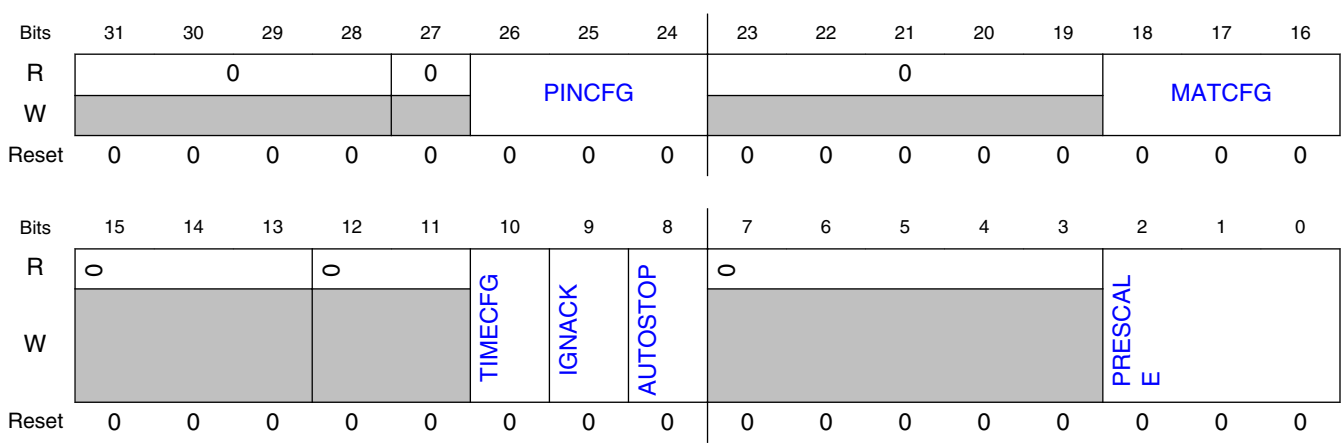
47.5.1.9.1 Offset

Register	Offset
MCFGR1	24h

47.5.1.9.2 Function

The MCFGR1 should only be written when the I2C Master is disabled.

47.5.1.9.3 Diagram



47.5.1.9.4 Fields

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td>31-28
—</td><td>Reserved</td></tr>
<tr><td>27
—</td><td>Reserved</td></tr>
<tr><td>26-24
PINCFG</td><td>Pin Configuration
Configures the pin mode for LPI2C.
Table 47-14. 2-pin / 4-pin pin configurations for masters and slaves
<table>
<tr><th>PINCFG</th><th>SCL / SDA pins</th><th>SCLS / SDAS pins</th></tr>
<tr><td>000</td><td>Bi-directional open drain for master and slave</td><td>Not used</td></tr>
<tr><td>001</td><td>Output-only (ultra-fast mode) open drain for master and slave</td><td>Not used</td></tr>
<tr><td>010</td><td>Bi-directional push-pull for master and slave</td><td>Not used</td></tr>
<tr><td>011</td><td>Input only for master and slave</td><td>Output-only push-pull for master and slave</td></tr>
<tr><td>100</td><td>Bi-directional open drain for master</td><td>Bi-directional open drain for slave</td></tr>
<tr><td>101</td><td>Output-only (ultra-fast mode) open drain for master</td><td>Output-only open drain for slave</td></tr>
<tr><td>110</td><td>Bi-directional push-pull for master</td><td>Bi-directional push-pull for slave</td></tr>
<tr><td>111</td><td>Input only for master and slave</td><td>Inverted output-only push-pull for master and slave</td></tr>
</table>
000b - 2-pin open drain mode
001b - 2-pin output only mode (ultra-fast mode)
010b - 2-pin push-pull mode
011b - 4-pin push-pull mode
100b - 2-pin open drain mode with separate LPI2C slave
101b - 2-pin output only mode (ultra-fast mode) with separate LPI2C slave
110b - 2-pin push-pull mode with separate LPI2C slave
111b - 4-pin push-pull mode (inverted outputs)</td></tr>
<tr><td>23-19
—</td><td>Reserved</td></tr>
<tr><td>18-16
MATCFG</td><td>Match Configuration
Configures the condition that causes MSR[DMF] to set. See Master Data Match (MDMR).
000b - Match is disabled
001b - Reserved
010b - Match is enabled (1st data word equals MDMR[MATCH0] OR MDMR[MATCH1])
011b - Match is enabled (any data word equals MDMR[MATCH0] OR MDMR[MATCH1])
100b - Match is enabled (1st data word equals MDMR[MATCH0] AND 2nd data word equals MDMR[MATCH1])
101b - Match is enabled (any data word equals MDMR[MATCH0] AND next data word equals MDMR[MATCH1])
110b - Match is enabled (1st data word AND MDMR[MATCH1] equals MDMR[MATCH0] AND MDMR[MATCH1])</td></tr>
</table>

Table continues on the next page...

Field	Function
	111b - Match is enabled (any data word AND MDMR[MATCH1] equals MDMR[MATCH0] AND MDMR[MATCH1])
15-13 —	Reserved
12-11 —	Reserved
10 TIMECFG	Timeout Configuration TIMECFG configures the timeout settings for the Pin Low Timeout Flag field (MSR[PLTF]). 0b - MSR[PLTF] sets if SCL is low for longer than the configured timeout 1b - MSR[PLTF] sets if either SCL or SDA is low for longer than the configured timeout
9 IGNACK	IGNACK When set, the received NACK field is ignored and assumed to be ACK. IGNACK bit is required to be set in Ultra-Fast Mode. 0b - LPI2C Master receives ACK and NACK normally 1b - LPI2C Master treats a received NACK as if it (NACK) was an ACK
8 AUTOSTOP	Automatic STOP Generation When enabled, a STOP condition is generated whenever the LPI2C master is busy and the transmit FIFO is empty. The STOP condition can also be generated using a transmit FIFO command. 0b - No effect 1b - STOP condition is automatically generated whenever the transmit FIFO is empty and the LPI2C master is busy
7-3 —	Reserved
2-0 PRESCALE	Prescaler Configures the clock prescaler used for all LPI2C master logic, except for the digital glitch filters. 000b - Divide by 1 001b - Divide by 2 010b - Divide by 4 011b - Divide by 8 100b - Divide by 16 101b - Divide by 32 110b - Divide by 64 111b - Divide by 128

47.5.1.10 Master Configuration 2 (MCFGR2)

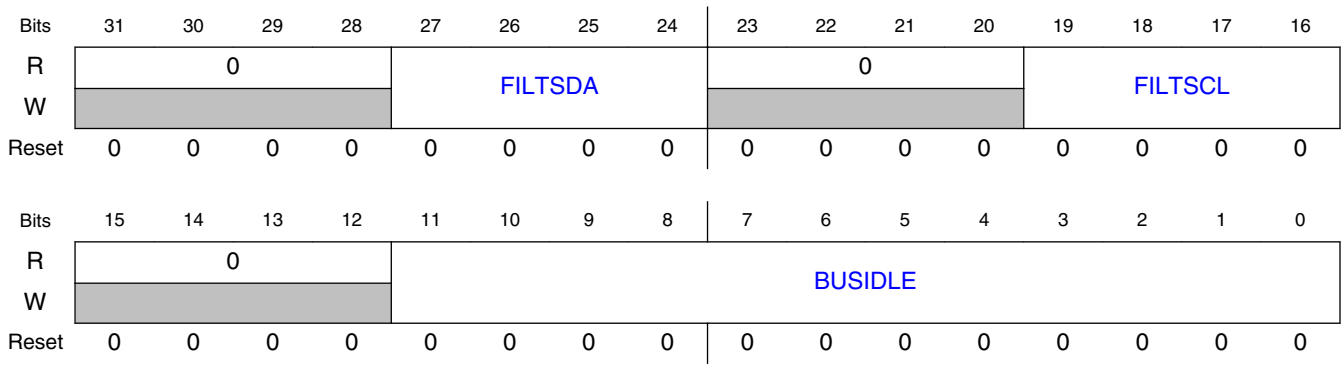
47.5.1.10.1 Offset

Register	Offset
MCFGR2	28h

47.5.1.10.2 Function

The Master Configuration Register 2 should only be written when the I2C Master is disabled.

47.5.1.10.3 Diagram



47.5.1.10.4 Fields

Field	Function
31-28 —	Reserved
27-24 FILTSDA	Glitch Filter SDA Configures the I2C master digital glitch filters for SDA input. • A configuration of 0 disables the glitch filter • Glitches equal to or less than FILTSDA cycles long are filtered out and ignored • The latency through the glitch filter is equal to FILTSDA cycles, and must be configured to be less than the minimum SCL low or high period • The glitch filter cycle count is not affected by the PRESCALE configuration, and the glitch filter cycle count is automatically bypassed in High Speed mode
23-20 —	Reserved
19-16 FILTSCL	Glitch Filter SCL Configures the I2C master digital glitch filters for SCL input. • A configuration of 0 disables the glitch filter • Glitches equal to or less than FILTSCL cycles long are filtered out and ignored.The FILTSCL cycles are based on the functional clock. • The latency through the glitch filter is equal to FILTSCL cycles, and must be configured to be less than the minimum SCL low or high period • The glitch filter cycle count is not affected by the PRESCALE configuration, and the glitch filter cycle count is automatically bypassed in High Speed mode
15-12 —	Reserved
11-0 BUSIDLE	Bus Idle Timeout Configures the bus idle timeout period in clock cycles.

Field	Function
	• If both SCL and SDA are high for longer than BUSIDLE cycles, then the I2C bus is assumed to be idle and the master can generate a START condition • When Bus Idle Timeout is set to zero, the Bus Idle Timeout is disabled

47.5.1.11 Master Configuration 3 (MCFGR3)

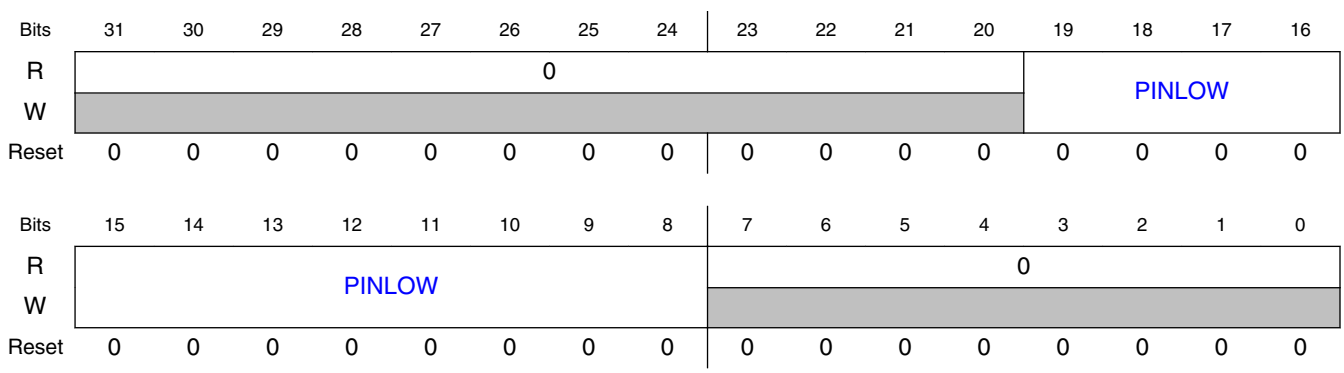
47.5.1.11.1 Offset

Register	Offset
MCFGR3	2Ch

47.5.1.11.2 Function

The MCFGR3 should only be written when the I2C Master is disabled.

47.5.1.11.3 Diagram



47.5.1.11.4 Fields

Field	Function
31-20 —	Reserved
19-8 PINLOW	Pin Low Timeout Configures the pin low timeout flag in clock cycles. • If SCL or, either SCL or SDA, is low for longer than (PINLOW * 256) cycles, then MSR[PLTF] is set. • When Pin Low Timeout is set to zero, the Pin Low Timeout feature is disabled
7-0	Reserved

Field	Function
—	

47.5.1.12 Master Data Match (MDMR)

47.5.1.12.1 Offset

Register	Offset
MDMR	40h

47.5.1.12.2 Function

The MDMR should only be written when the I2C Master is disabled or idle.

47.5.1.12.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0								MATCH1							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								MATCH0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

47.5.1.12.4 Fields

Field	Function
31-24 —	Reserved
23-16 MATCH1	Match 1 Value Compared against the received data when receive data match is enabled.
15-8 —	Reserved
7-0 MATCH0	Match 0 Value Compared against the received data when receive data match is enabled.

47.5.1.13 Master Clock Configuration 0 (MCCR0)

47.5.1.13.1 Offset

Register	Offset
MCCR0	48h

47.5.1.13.2 Function

The MCCR0 cannot be changed when the I2C master is enabled and is used for standard, fast, fast-mode plus and ultra-fast transfers.

47.5.1.13.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0		DATAVD						0		SETHOLD					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0		CLKHI						0		CLKLO					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

47.5.1.13.4 Fields

Field	Function
31-30 —	Reserved
29-24 DATAVD	Data Valid Delay Minimum number of cycles (minus one) that is used as the data hold time for SDA. Must be configured less than the minimum SCL low period.
23-22 —	Reserved
21-16 SETHOLD	Setup Hold Delay Minimum number of cycles (minus one) that is used by the master for these conditions: • Hold time for a START

Table continues on the next page...

Field	Function
	• Setup and hold time for a repeated START • Setup time for a STOP The setup time is extended by the time it takes to detect a rising edge on the external SCL pin. Ignoring any additional board delay due to external loading, this time is equal to (2 + FILTSCL) / 2^PRESCALE cycles.
15-14 —	Reserved
13-8 CLKHI	Clock High Period Minimum number of cycles (minus one) that the SCL clock is driven high by the master. The SCL high time is extended by the time it takes to detect a rising edge on the external SCL pin. Ignoring any additional board delay due to external loading, this time is equal to (2 + FILTSCL) / 2^PRESCALE cycles.
7-6 —	Reserved
5-0 CLKLO	Clock Low Period Minimum number of cycles (minus one) that the SCL clock is driven low by the master. The Clock Low Period value is also used for the minimum bus free time between a STOP and a START condition; this is extended by the time it takes to detect a rising edge on the external SCL pin. Ignoring any additional board delay due to external loading, this time is equal to (2 + FILTSCL) / 2^PRESCALE cycles.

47.5.1.14 Master Clock Configuration 1 (MCCR1)

47.5.1.14.1 Offset

Register	Offset
MCCR1	50h

47.5.1.14.2 Function

The MCCR1 cannot be changed when the I2C master is enabled and is used for high speed mode transfers. The separate clock configuration for high speed mode allows arbitration to take place in Fast mode (with timing configured by Master Clock Configuration 0 (MCCR0)), before switching to high speed mode (with timing configured by MCCR1).

47.5.1.14.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0		DATAVD						0		SETHOLD					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0		CLKHI						0		CLKLO					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

47.5.1.14.4 Fields

Field	Function
31-30 —	Reserved
29-24 DATAVD	Data Valid Delay Minimum number of cycles (minus one) that is used as the data hold time for SDA. The DATAVD must be configured to be less than the minimum SCL low period.
23-22 —	Reserved
21-16 SETHOLD	Setup Hold Delay Minimum number of cycles (minus one) that is used by the master for these conditions: • Hold time for a START condition • Setup and hold time for a repeated START condition • Setup time for a STOP condition The setup time is extended by the time it takes to detect a rising edge on the external SCL pin. Ignoring any additional board delay due to external loading, this time is equal to (2 + FILTSCL) / 2^PRESCALE cycles.
15-14 —	Reserved
13-8 CLKHI	Clock High Period Minimum number of cycles (minus one) that the SCL clock is driven high by the master. The SCL high time is extended by the time it takes to detect a rising edge on the external SCL pin. Ignoring any additional board delay due to external loading, this time is equal to (2 + FILTSCL) / 2^PRESCALE cycles.
7-6 —	Reserved
5-0 CLKLO	Clock Low Period Minimum number of cycles (minus one) that the SCL clock is driven low by the master. The CLKLO value is also used for the minimum bus free time between a STOP and a START condition; this is extended by the time it takes to detect a rising edge on the external SCL pin. Ignoring any additional board delay due to external loading, this time is equal to (2 + FILTSCL) / 2^PRESCALE cycles.

47.5.1.15 Master FIFO Control (MFCR)

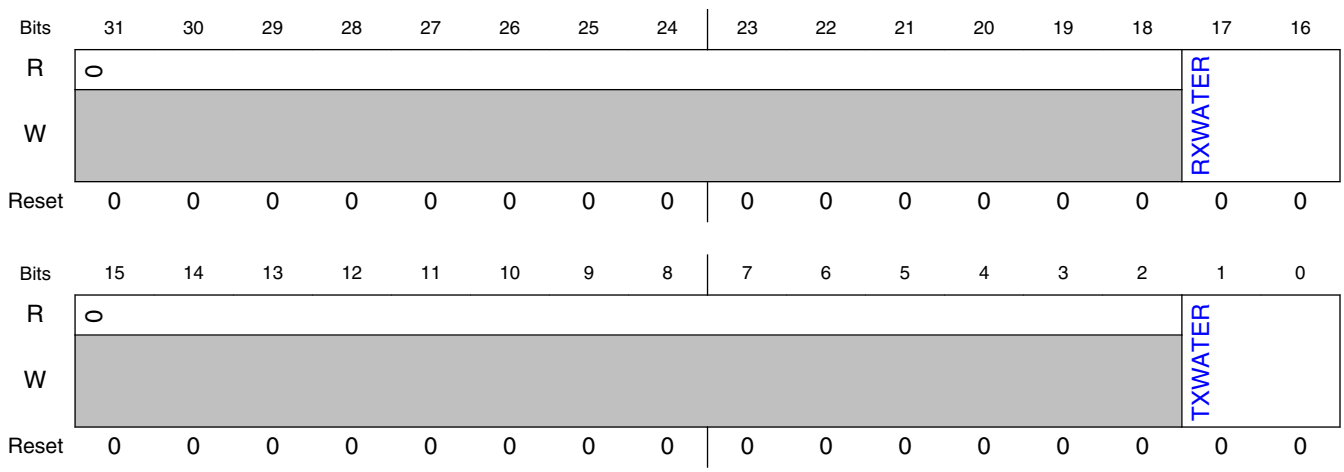
47.5.1.15.1 Offset

Register	Offset
MFCR	58h

47.5.1.15.2 Function

The Master FIFO control register is only used in Stop mode when the MFCR register is static (i.e., the MFCR register is not changing).

47.5.1.15.3 Diagram



47.5.1.15.4 Fields

Field	Function
31-18 —	Reserved
17-16 RXWATER	Receive FIFO Watermark The Receive Data Flag (SSR[RDF]) is set whenever the number of words in the receive FIFO is greater than RXWATER. Writing a value equal to or greater than the FIFO size truncates the value.
15-2 —	Reserved
1-0	Transmit FIFO Watermark

Field	Function
TXWATER	The Transmit Data Flag (SSR[TDF]) is set whenever the number of words in the transmit FIFO is equal or less than TXWATER. Writing a value equal to or greater than the FIFO size truncates the value.

47.5.1.16 Master FIFO Status (MFSR)

47.5.1.16.1 Offset

Register	Offset
MFSR	5Ch

47.5.1.16.2 Function

MFSR gives the number of words in the transmit and receive FIFO.

47.5.1.16.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0													RXCOUNT		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0													TXCOUNT		
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

47.5.1.16.4 Fields

Field	Function
31-19 —	Reserved
18-16 RXCOUNT	Receive FIFO Count Returns the number of words in the receive FIFO.
15-3 —	Reserved
2-0 TXCOUNT	Transmit FIFO Count Returns the number of words in the transmit FIFO.

47.5.1.17 Master Transmit Data (MTDR)

47.5.1.17.1 Offset

Register	Offset
MTDR	60h

47.5.1.17.2 Function

- An 8-bit write to the CMD field is ignored and does not increment the FIFO write pointer.
- An 8-bit write to the DATA field zero-extends the CMD field and increments the FIFO write pointer.
- A 16-bit or 32-bit writes both the CMD and DATA fields and increments the FIFO write pointer.

47.5.1.17.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	0					CMD			DATA							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

47.5.1.17.4 Fields

Field	Function
31-11 —	Reserved
10-8 CMD	Command Data 000b - Transmit DATA[7:0] 001b - Receive (DATA[7:0] + 1) bytes 010b - Generate STOP condition 011b - Receive and discard (DATA[7:0] + 1) bytes

Table continues on the next page...

Field	Function
	100b - Generate (repeated) START and transmit address in DATA[7:0] 101b - Generate (repeated) START and transmit address in DATA[7:0]. This transfer expects a NACK to be returned. 110b - Generate (repeated) START and transmit address in DATA[7:0] using high speed mode 111b - Generate (repeated) START and transmit address in DATA[7:0] using high speed mode. This transfer expects a NACK to be returned.
7-0 DATA	Transmit Data Performing an 8-bit write to DATA zero-extends the CMD field.

47.5.1.18 Master Receive Data (MRDR)

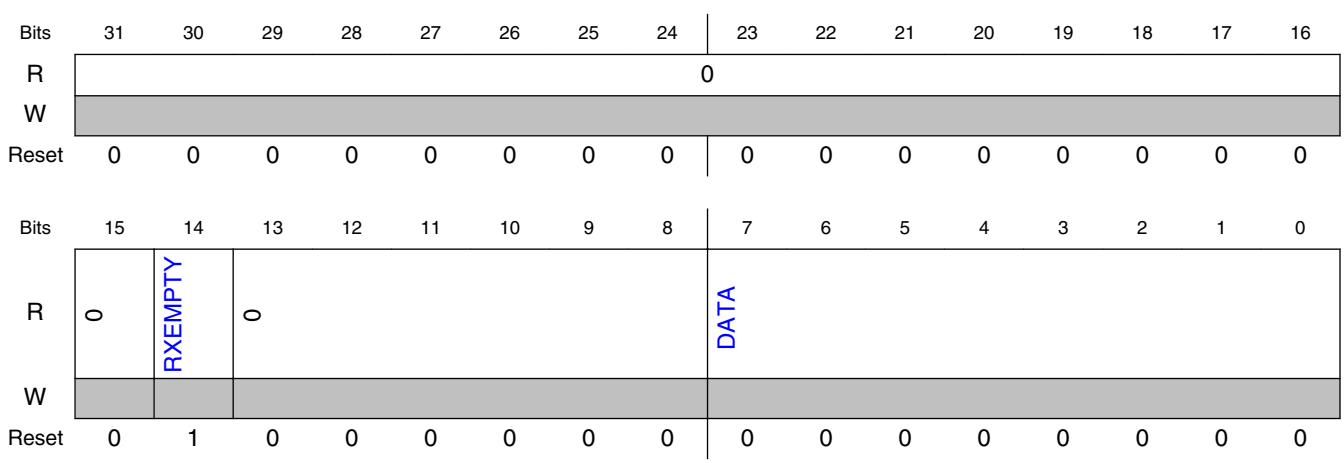
47.5.1.18.1 Offset

Register	Offset
MRDR	70h

47.5.1.18.2 Function

Reading the Receive Data register returns the data received by the I2C master that has not been discarded.

47.5.1.18.3 Diagram



47.5.1.18.4 Fields

Field	Function
31-15 —	Reserved
14 RXEMPTY	RX Empty RXEMPTY gives the empty status of the Master receive data FIFO. 0b - Receive FIFO is not empty 1b - Receive FIFO is empty
13-8 —	Reserved
7-0 DATA	Receive Data Receive data can be discarded due to the CMD field, or the master can be configured to discard non-matching data.

47.5.1.19 Slave Control (SCR)

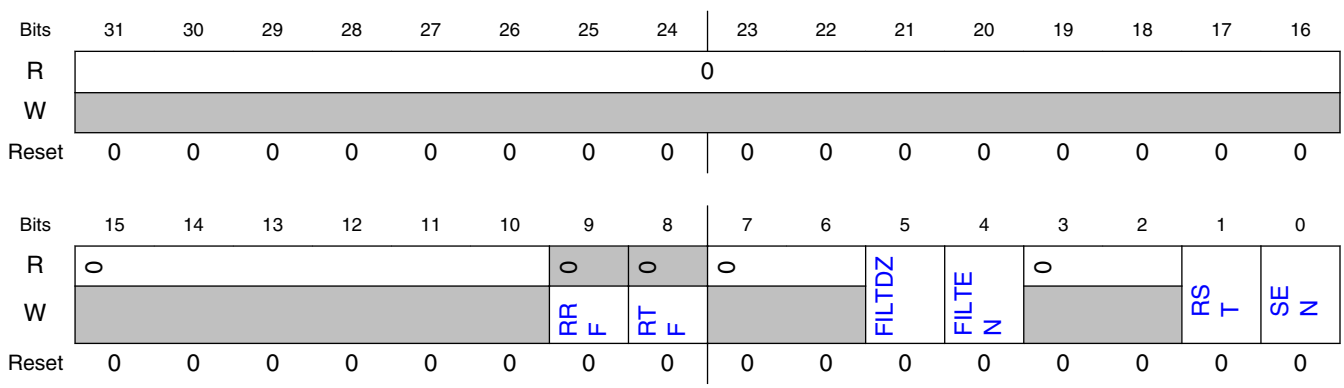
47.5.1.19.1 Offset

Register	Offset
SCR	110h

47.5.1.19.2 Function

SCR contains resets and other slave control settings.

47.5.1.19.3 Diagram



47.5.1.19.4 Fields

Field	Function
31-10 —	Reserved
9 RRF	Reset Receive FIFO Resets the receive FIFO to b be empty. 0b - No effect 1b - Receive Data Register is now empty
8 RTF	Reset Transmit FIFO Resets the transmit FIFO to be empty. 0b - No effect 1b - Transmit Data Register is now empty
7-6 —	Reserved
5 FILTDZ	Filter Doze Enable FILTDZ should only be updated when the I2C Slave is disabled. 0b - Filter remains enabled in Doze mode 1b - Filter is disabled in Doze mode
4 FILTEN	Filter Enable FILTEN should only be updated when the I2C Slave is disabled. 0b - Disable digital filter and output delay counter for slave mode 1b - Enable digital filter and output delay counter for slave mode
3-2 —	Reserved
1 RST	Software Reset The reset takes effect immediately and remains asserted until negated by software. There is no minimum delay required before clearing the software reset. 0b - Slave mode logic is not reset 1b - Slave mode logic is reset
0 SEN	Slave Enable Enables the I2C Slave mode. 0b - I2C Slave mode is disabled 1b - I2C Slave mode is enabled

47.5.1.20 Slave Status (SSR)

47.5.1.20.1 Offset

Register	Offset
SSR	114h

47.5.1.20.2 Function

SSR contains status flags for transmit and receive data; error conditions; and bus and slave busy or idle status.

47.5.1.20.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0						BBF	SBF	0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SARF	GCF	AM1F	AM0F	FEF	BEF	SDF	RSF	0				TAF	AVF	RDF	TDF
W					W1C	W1C	W1C	W1C								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

47.5.1.20.4 Fields

Field	Function
31-26 —	Reserved
25 BBF	Bus Busy Flag Indicates if an I2C bus is idle or busy. 0b - I2C Bus is idle 1b - I2C Bus is busy
24 SBF	Slave Busy Flag Indicates if an I2C slave is idle or busy. 0b - I2C Slave is idle 1b - I2C Slave is busy
23-16 —	Reserved
15 SARF	SMBus Alert Response Flag • SARF is cleared by reading the Slave Address Status (SASR) register • SARF cannot generate an asynchronous wakeup 0b - SMBus Alert Response is disabled or not detected 1b - SMBus Alert Response is enabled and detected
14 GCF	General Call Flag Indicates whether a slave has detected the General Call Address.

Table continues on the next page...

Field	Function
	• General Call Flag is cleared by reading the Slave Address Status (SASR) register • General Call Flag cannot generate an asynchronous wakeup 0b - Slave has not detected the General Call Address or the General Call Address is disabled 1b - Slave has detected the General Call Address
13 AM1F	Address Match 1 Flag Indicates that the received address has matched the ADDR1 field or ADDR0 to ADDR1 range as configured by SCFGR1[ADDRCFG]. • Address Match 1 Flag is cleared by reading the Slave Address Status (SASR) register • Address Match 1 Flag cannot generate an asynchronous wakeup 0b - Have not received an ADDR1 or ADDR0/ADDR1 range matching address 1b - Have received an ADDR1 or ADDR0/ADDR1 range matching address
12 AM0F	Address Match 0 Flag Indicates that the received address has matched the ADDR0 field as configured by SCFGR1[ADDRCFG]. • AM0F is cleared by reading the Slave Address Status (SASR) register • AM0F cannot generate an asynchronous wakeup 0b - Have not received an ADDR0 matching address 1b - Have received an ADDR0 matching address
11 FEF	FIFO Error Flag FEF can only be set when clock stretching is disabled. 0b - FIFO underflow or overflow was not detected 1b - FIFO underflow or overflow was detected
10 BEF	Bit Error Flag BEF sets if the LPI2C slave transmits a logic one and detects a logic zero on the I2C bus. The slave ignores the rest of the transfer until the next (repeated) START condition. 0b - Slave has not detected a bit error 1b - Slave has detected a bit error
9 SDF	STOP Detect Flag SDF sets when the LPI2C slave detects a STOP condition and if the LPI2C slave matched the last address byte. 0b - Slave has not detected a STOP condition 1b - Slave has detected a STOP condition
8 RSF	Repeated Start Flag RSF sets when the LPI2C slave detects a repeated START condition and if the LPI2C slave matched the last address byte. The RSF does not set when the slave first detects a START condition. 0b - Slave has not detected a Repeated START condition 1b - Slave has detected a Repeated START condition
7-4 —	Reserved
3 TAF	Transmit ACK Flag TAF is cleared by writing to the Slave Transmit ACK (STAR) register. 0b - Transmit ACK/NACK is not required 1b - Transmit ACK/NACK is required
2 AVF	Address Valid Flag AVF is cleared by reading the Slave Address Status (SASR) register. When SCFGR1[RXCFG] = 1, SSR[AVF] is also cleared by reading the Slave Receive Data (SRDR) register. 0b - Address Status Register is not valid

Table continues on the next page...

Field	Function
	1b - Address Status Register is valid
1 RDF	Receive Data Flag RDF is cleared by reading the Slave Receive Data (SRDR) register. When SCFGR1[RXCFG] = 1, RDF is not cleared when reading the Slave Receive Data (SRDR) register and if SSR[AVF] is set. 0b - Receive data is not ready 1b - Receive data is ready
0 TDF	Transmit Data Flag TDF is cleared by writing the to the Slave Transmit Data (STDR) register. When SCFGR1[TXCFG] = 0 and if a NACK or a Repeated START or STOP condition is detected, then TDF is also cleared. 0b - Transmit data not requested 1b - Transmit data is requested

47.5.1.21 Slave Interrupt Enable (SIER)

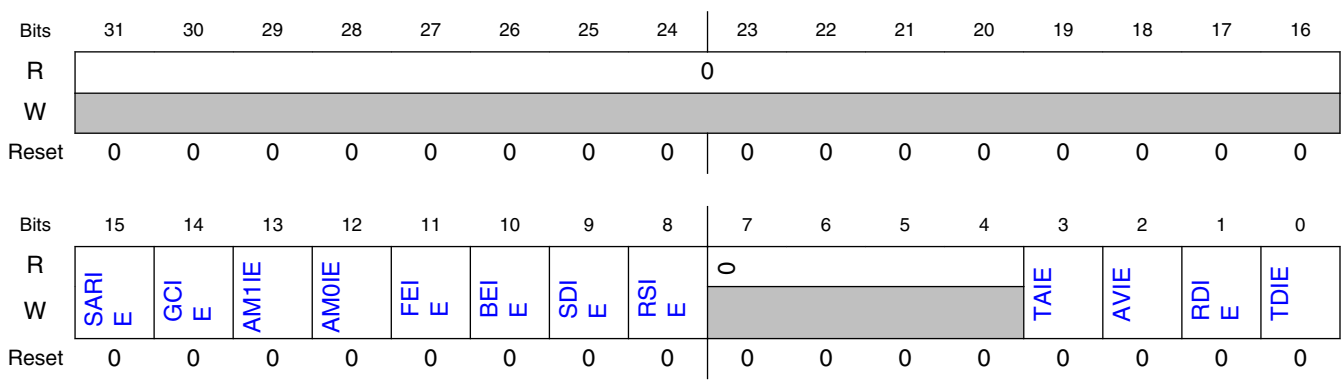
47.5.1.21.1 Offset

Register	Offset
SIER	118h

47.5.1.21.2 Function

SIER contains transmit and receive data interrupt enables; Start and Stop detect interrupt enables; and other slave interrupt enables.

47.5.1.21.3 Diagram



47.5.1.21.4 Fields

Field	Function
31-16 —	Reserved
15 SARIE	SMBus Alert Response Interrupt Enable Enables interrupt for SMBus alert response. 0b - Disabled 1b - Enabled
14 GCIE	General Call Interrupt Enable Enables interrupt for general call. 0b - Disabled 1b - Enabled
13 AM1IE	Address Match 1 Interrupt Enable Enables interrupt for address match 1. 0b - Disabled 1b - Enabled
12 AM0IE	Address Match 0 Interrupt Enable Enables interrupt for address match 0. 0b - Disabled 1b - Enabled
11 FEIE	FIFO Error Interrupt Enable Enables interrupt for FIFO error. 0b - Disabled 1b - Enabled
10 BEIE	Bit Error Interrupt Enable Enables interrupt for bit error. 0b - Disabled 1b - Enabled
9 SDIE	STOP Detect Interrupt Enable Enables interrupt for STOP detect. 0b - Disabled 1b - Enabled
8 RSIE	Repeated Start Interrupt Enable Enables interrupt for repeated start. 0b - Disabled 1b - Enabled
7-4 —	Reserved
3 TAIE	Transmit ACK Interrupt Enable Enables interrupt for transmit ACK. 0b - Disabled 1b - Enabled
2 AVIE	Address Valid Interrupt Enable Enables interrupt for valid address. 0b - Disabled 1b - Enabled

Table continues on the next page...

Field	Function
1 RDIE	Receive Data Interrupt Enable Enables interrupt for receive data. 0b - Disabled 1b - Enabled
0 TDIE	Transmit Data Interrupt Enable Enables interrupt for transmit data. 0b - Disabled 1b - Enabled

47.5.1.22 Slave DMA Enable (SDER)

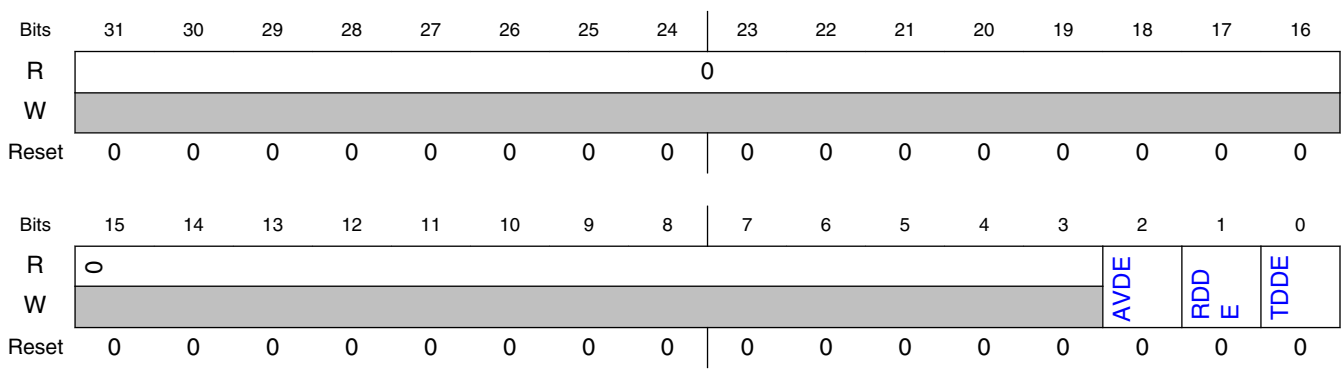
47.5.1.22.1 Offset

Register	Offset
SDER	11Ch

47.5.1.22.2 Function

SDER contains the transmit, request, and receive enables for DMA.

47.5.1.22.3 Diagram



47.5.1.22.4 Fields

Field	Function
31-3 —	Reserved

Table continues on the next page...

Field	Function
2 AVDE	Address Valid DMA Enable The Address Valid DMA request is shared with the Receive Data DMA request. If both Address Valid DMA request and Receive Data DMA request are enabled, then set SCFGR1[RXCFG] = 1, to allow the DMA to read the address from the Slave Receive Data (SRDR) register. 0b - DMA request is disabled 1b - DMA request is enabled
1 RDDE	Receive Data DMA Enable Enables receive data for DMA. 0b - DMA request is disabled 1b - DMA request is enabled
0 TDDE	Transmit Data DMA Enable Enables transmit data for DMA. 0b - DMA request is disabled 1b - DMA request is enabled

47.5.1.23 Slave Configuration 1 (SCFGR1)

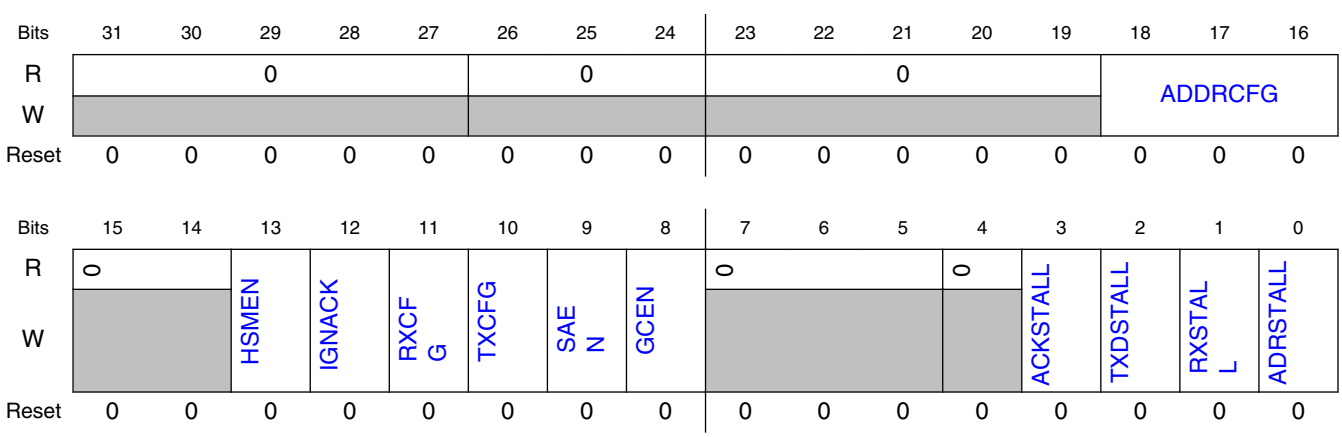
47.5.1.23.1 Offset

Register	Offset
SCFGR1	124h

47.5.1.23.2 Function

The Slave Configuration Register 1 should only be written when the I2C Slave is disabled.

47.5.1.23.3 Diagram



47.5.1.23.4 Fields

Field	Function
31-27 —	Reserved
26-24 —	Reserved
23-19 —	Reserved
18-16 ADDRCFG	Address Configuration Configures the condition that causes an address to match. 000b - Address match 0 (7-bit) 001b - Address match 0 (10-bit) 010b - Address match 0 (7-bit) or Address match 1 (7-bit) 011b - Address match 0 (10-bit) or Address match 1 (10-bit) 100b - Address match 0 (7-bit) or Address match 1 (10-bit) 101b - Address match 0 (10-bit) or Address match 1 (7-bit) 110b - From Address match 0 (7-bit) to Address match 1 (7-bit) 111b - From Address match 0 (10-bit) to Address match 1 (10-bit)
15-14 —	Reserved
13 HSMEN	High Speed Mode Enable Enables detection of the High-Speed Mode master code of slave address 0000_1XX, but does not cause an address match on this code. When set and any HS-mode master code is detected, SCR[FILTEN] and SCFGR1[ACKSTALL] bits are ignored until the next STOP condition is detected. 0b - Disables detection of HS-mode master code 1b - Enables detection of HS-mode master code
12 IGNACK	Ignore NACK When Ignore NACK is set, the LPI2C slave continues transfers after a NACK is detected. Ignore NACK bit is required to be set in Ultra-Fast mode. 0b - Slave ends transfer when NACK is detected 1b - Slave does not end transfer when NACK detected
11 RXCFG	Receive Data Configuration 0b - Reading the Receive Data register returns received data and clears the Receive Data flag (MSR[RDF]). 1b - Reading the Receive Data register when the Address Valid flag (SSR[AVF])is set, returns the Address Status register and clear the Address Valid flag. Reading the Receive Data register when the Address Valid flag is clear, returns received data and clears the Receive Data flag (MSR[RDF]).
10 TXCFG	Transmit Flag Configuration The transmit data flag always asserts before a NACK is detected at the end of a slave-transmit transfer. This can cause an extra word to be written to the transmit data FIFO. • When TXCFG = 0, the Transmit Data register is automatically emptied when a slave-transmit transfer is detected. This causes the transmit data flag to assert whenever a slave-transmit transfer is detected, and causes the transmit data flag to negate at the end of the slave-transmit transfer. • When TXCFG = 1, the Transmit Data flag asserts whenever the Transmit Data register is empty, and the Transmit Data flag negates when the Transmit Data register is full. This allows the Transmit Data register to be filled before a slave-transmit transfer is detected, but can cause the Transmit Data register to be written before a NACK is detected on the last byte of a slave transmit transfer.

Table continues on the next page...

Field	Function
	0b - Transmit Data Flag only asserts during a slave-transmit transfer when the Transmit Data register is empty 1b - Transmit Data Flag asserts whenever the Transmit Data register is empty
9 SAEN	SMBus Alert Enable Enables a match on an SMBus alert. 0b - Disables match on SMBus Alert 1b - Enables match on SMBus Alert
8 GCEN	General Call Enable Enables a general call address. 0b - General Call address is disabled 1b - General Call address is enabled
7-5 —	Reserved
4 —	Reserved
3 ACKSTALL	ACK SCL Stall Enables SCL clock stretching during slave-transmit address byte(s) and slave-receiver address and data byte(s), to allow software to write the Slave Transmit ACK (STAR) register before the ACK or NACK is transmitted. Clock stretching occurs when transmitting the 9th bit, and is therefore not compatible with high speed mode. When ACKSTALL is enabled, there is no need to set either RX SCL Stall (SCFGR1[RXSTALL]) or Address SCL Stall (SCFGR1[ADRSTALL]). 0b - Clock stretching is disabled 1b - Clock stretching is enabled
2 TXDSTALL	TX Data SCL Stall Enables SCL clock stretching when the SSR[TDF] = 1 during a slave-transmit transfer. Clock stretching occurs following the 9th bit, and is therefore compatible with high speed mode. 0b - Clock stretching is disabled 1b - Clock stretching is enabled
1 RXSTALL	RX SCL Stall Enables SCL clock stretching when SSR[RDF] = 1 during a slave-receive transfer. Clock stretching occurs following the 9th bit, and is therefore compatible with high speed mode. 0b - Clock stretching is disabled 1b - Clock stretching is enabled
0 ADRSTALL	Address SCL Stall Enables SCL clock stretching when SSR[AVF] = 1. Clock stretching only occurs following the 9th bit, and is therefore compatible with high speed mode. 0b - Clock stretching is disabled 1b - Clock stretching is enabled

47.5.1.24 Slave Configuration 2 (SCFGR2)

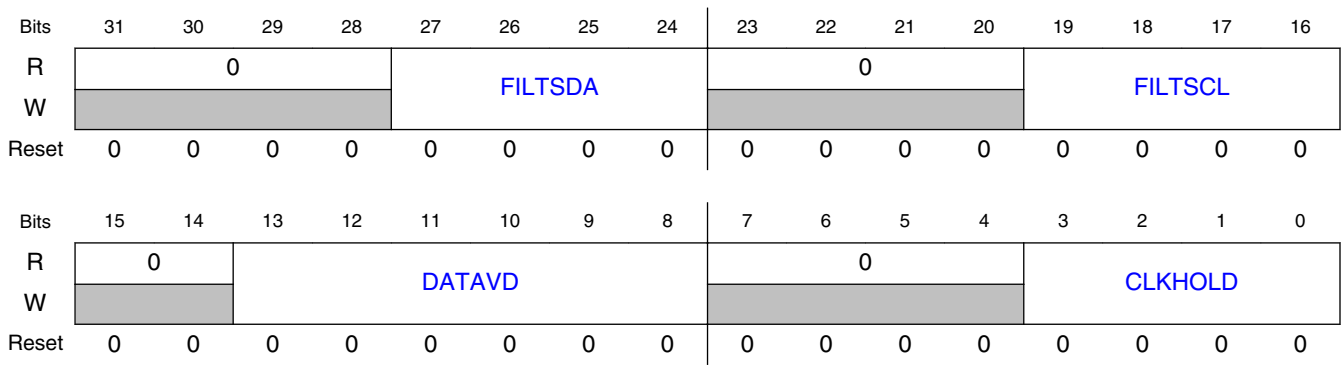
47.5.1.24.1 Offset

Register	Offset
SCFGR2	128h

47.5.1.24.2 Function

The Slave Configuration Register 2 should only be written when the I2C Slave is disabled.

47.5.1.24.3 Diagram



47.5.1.24.4 Fields

Field	Function
31-28 —	Reserved
27-24 FILTSDA	Glitch Filter SDA Configures the I2C slave digital glitch filters for SDA input. • A configuration of 0 disables the glitch filter • Glitches equal to or less than FILTSDA cycles long are filtered out and ignored • The latency through the glitch filter is equal to FILTSDA+3 cycles, and must be configured to be less than the minimum SCL low or high period • The glitch filter cycle count is not affected by the PRESCALE configuration, and the glitch filter cycle count is disabled in high speed mode
23-20 —	Reserved
19-16 FILTSCL	Glitch Filter SCL Configures the I2C slave digital glitch filters for SCL input. • A configuration of 0 disables the glitch filter • Glitches equal to or less than FILTSCL cycles long are filtered out and ignored

Table continues on the next page...

Field	Function
	• The latency through the glitch filter is equal to FILTSCL+3 cycles, and must be configured to be less than the minimum SCL low or high period • The glitch filter cycle count is not affected by the PRESCALE configuration, and the glitch filter cycle count is disabled in high speed mode
15-14 —	Reserved
13-8 DATAVD	Data Valid Delay Configures the SDA data valid delay time for the I2C slave, and is equal to FILTSCL+DATAVD+3 cycles. • The data valid delay must be configured to be less than the minimum SCL low period • The I2C slave data valid delay time is not affected by the PRESCALE configuration, and the I2C slave data valid delay time is disabled in high speed mode
7-4 —	Reserved
3-0 CLKHOLD	Clock Hold Time Configures the minimum clock hold time for the I2C slave, when clock stretching is enabled. • The minimum hold time is equal to CLKHOLD+3 cycles • The I2C slave clock hold time is not affected by the PRESCALE configuration, and the I2C slave clock hold time is disabled in high speed mode

47.5.1.25 Slave Address Match (SAMR)

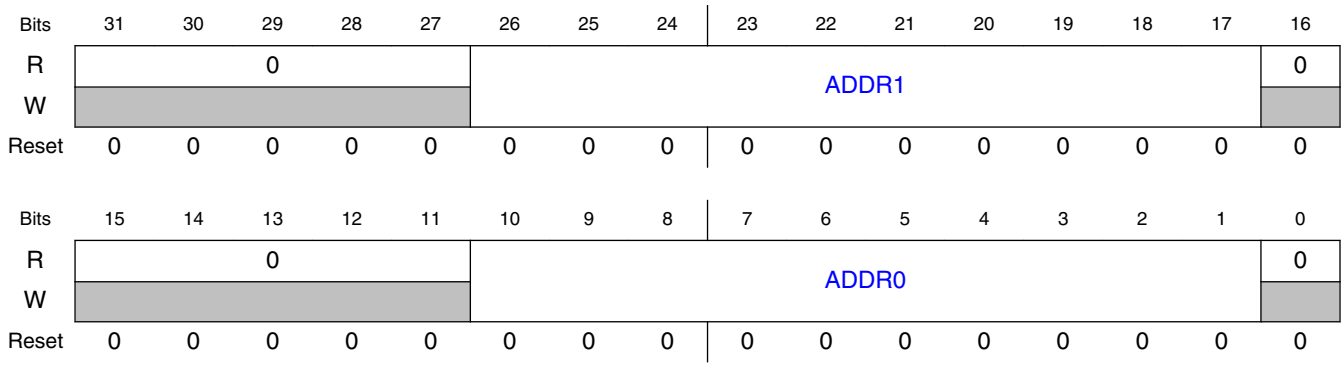
47.5.1.25.1 Offset

Register	Offset
SAMR	140h

47.5.1.25.2 Function

The SAMR should only be written when the I2C Slave is disabled.

47.5.1.25.3 Diagram



47.5.1.25.4 Fields

Field	Function
31-27 —	Reserved
26-17 ADDR1	Address 1 Value Compared against the received address to detect the Slave Address. • In 10-bit mode, the first address byte is compared to { 11110, ADDR1[26:25] } and the second address byte is compared to ADDR1[24:17] • In 7-bit mode, the address is compared to ADDR1[23:17]
16-11 —	Reserved
10-1 ADDR0	Address 0 Value Compared against the received address to detect the Slave Address. • In 10-bit mode, the first address byte is compared to { 11110, ADDR0[10:9] } and the second address byte is compared to ADDR0[8:1] • In 7-bit mode, the address is compared to ADDR0[7:1] •
0 —	Reserved

47.5.1.26 Slave Address Status (SASR)

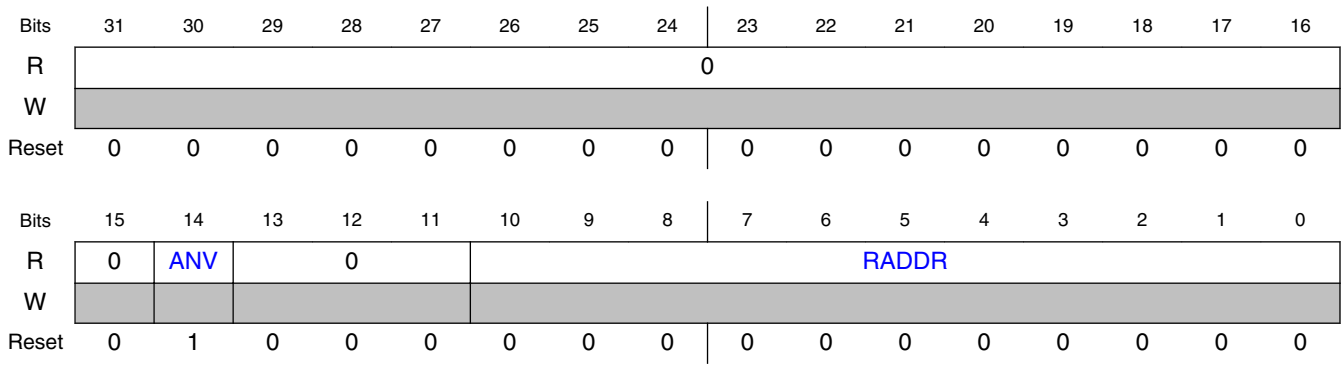
47.5.1.26.1 Offset

Register	Offset
SASR	150h

47.5.1.26.2 Function

SASR contains the read-only RADDR and ANV fields.

47.5.1.26.3 Diagram



47.5.1.26.4 Fields

Field	Function
31-15 —	Reserved
14 ANV	Address Not Valid 0b - Received Address (RADDR) is valid 1b - Received Address (RADDR) is not valid
13-11 —	Reserved
10-0 RADDR	Received Address The Received Address updates whenever the AMF is set; the AMF is cleared by reading the Slave Address Status register. • In 7-bit mode, the address byte is stored in RADDR[7:0] • In 10-bit mode, the first address byte is { 11110, RADDR[10:9], RADDR[0] } and the second address byte is RADDR[8:1]. The R/W bit is therefore always stored in RADDR[0]

47.5.1.27 Slave Transmit ACK (STAR)

47.5.1.27.1 Offset

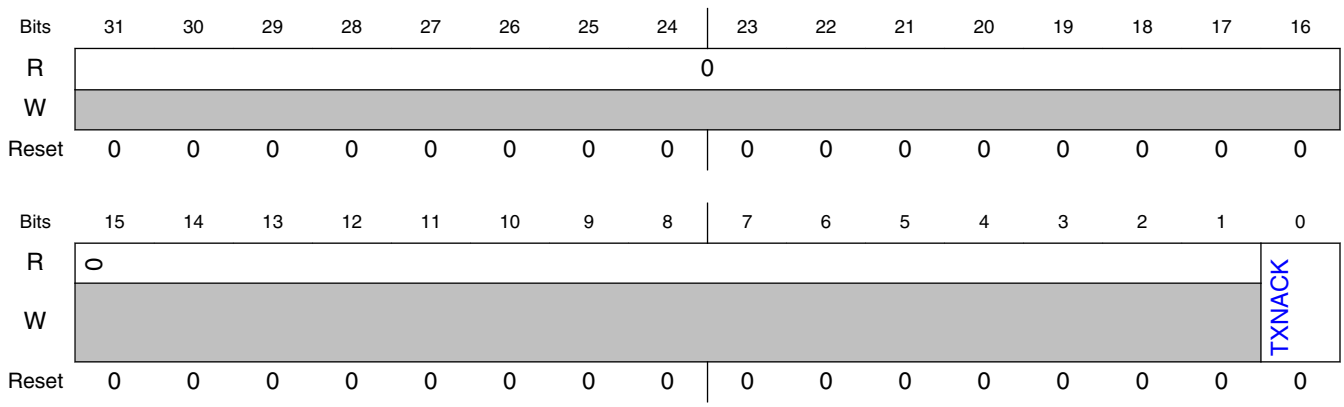
Register	Offset
STAR	154h

47.5.1.27.2 Function

STAR can only be written when the ACK SCL Stall bit is set (SCFGR1[ACKSTALL]).

- The SCFGR1[ACKSTALL] bit enables clock stretching during the ACK/NACK bit slot, and during this time, the STAR register can be written by software.
- The logic ensures that the clock stretching continues for at least 1 bus clock cycle after the STAR register is updated.
- This clock stretching time can be extended more using the Clock Hold Time field (SCFGR2[CLKHOLD]).

47.5.1.27.3 Diagram



47.5.1.27.4 Fields

Field	Function
31-1 —	Reserved
0 TXNACK	Transmit NACK After receiving each word, software can transmit either an ACK (logic 0) or a NACK (logic 1); Transmit NACK selects which to use: ACK or NACK. • When SCFGR1[ACKSTALL] is set, a Transmit NACK must be written once for each matching address byte and each received word. ACKSTALL must be set, because that stalls the data transfer until software reads the received word (and decides whether to respond with an ACK or NACK). • To configure the default ACK/NACK, Transmit NACK can also be written when LPI2C Slave is disabled or idle. 0b - Write a Transmit ACK for each received word 1b - Write a Transmit NACK for each received word

47.5.1.28 Slave Transmit Data (STDR)

47.5.1.28.1 Offset

Register	Offset
STDR	160h

47.5.1.28.2 Function

Clock stretching (enabled or disabled) affects when the transmit data is transferred. The SCFGR1[TXDSTALL] bit enables clock stretching during the 1st data bit of a slave-transmit transfer.

- **If clock stretching is enabled (SCFGR1[TXDSTALL] = 1)**, then the transmit data transfer is stalled until the STDR is updated. Clock stretching is extended by at least 1 bus clock cycle after STDR is updated, and clock stretching can be delayed even more using the Clock Hold Time field (SCFGR2[CLKHOLD].
- **If clock stretching is disabled (SCFGR1[TXDSTALL] = 0)**, then the transmit data should be written before the start of the slave-transmit transfer; otherwise (that is, if the transmit data is not written before the start of the slave-transmit transfer), the FIFO Error Flag (SSR[FEF]) sets.

47.5.1.28.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	0								DATA							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

47.5.1.28.4 Fields

Field	Function
31-8 —	Reserved
7-0	Transmit Data

Field	Function
DATA	Writing to the Slave Transmit Data Register (STDR) stores I2C slave transmit data *in* the Slave Transmit Data Register

47.5.1.29 Slave Receive Data (SRDR)

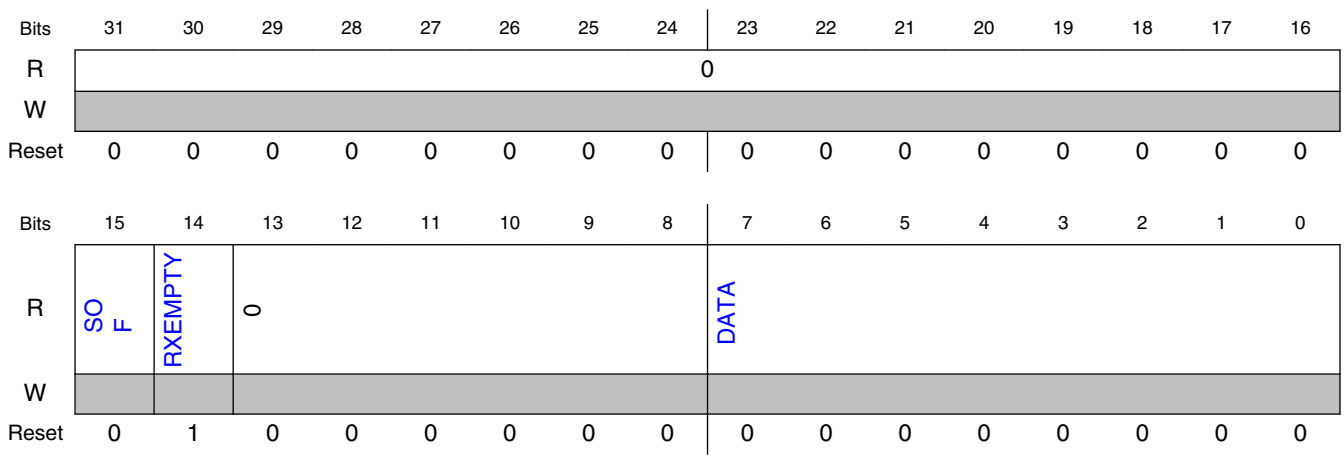
47.5.1.29.1 Offset

Register	Offset
SRDR	170h

47.5.1.29.2 Function

Reading the Slave Receive Data Register returns the data received by the I2C slave.

47.5.1.29.3 Diagram



47.5.1.29.4 Fields

Field	Function
31-16 —	Reserved
15 SOF	Start Of Frame Indicates whether this is the first data word. 0b - Indicates this is not the first data word since a (repeated) START or STOP condition

Table continues on the next page...

Field	Function
	1b - Indicates this is the first data word since a (repeated) START or STOP condition
14 RXEMPTY	RX Empty Gives the empty status of the receive data register. 0b - The Receive Data Register is not empty 1b - The Receive Data Register is empty
13-8 —	Reserved
7-0 DATA	Receive Data Contains data received by the I2C slave.

Chapter 48
Low Power Serial Peripheral Interface (LPSPI)

48.1 Chip-specific LPSPI information

Table 48-1. Reference links to related information

Topic	Related module(s)	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX

NOTE

The output trigger in "Peripheral Triggers" section is not applicable for this device.

48.2 Overview

LPSPI is a low-power Serial Peripheral Interface (SPI) module that supports an efficient interface to an SPI bus, either as a master and/or as a slave. The SPI bus is a synchronous serial communication interface used in embedded systems, typically to perform short distance communications between microcontrollers and peripheral devices, on printed circuit boards. Typical applications include interfacing to Secure Digital cards and LCD displays.

48.2.1 Block diagram

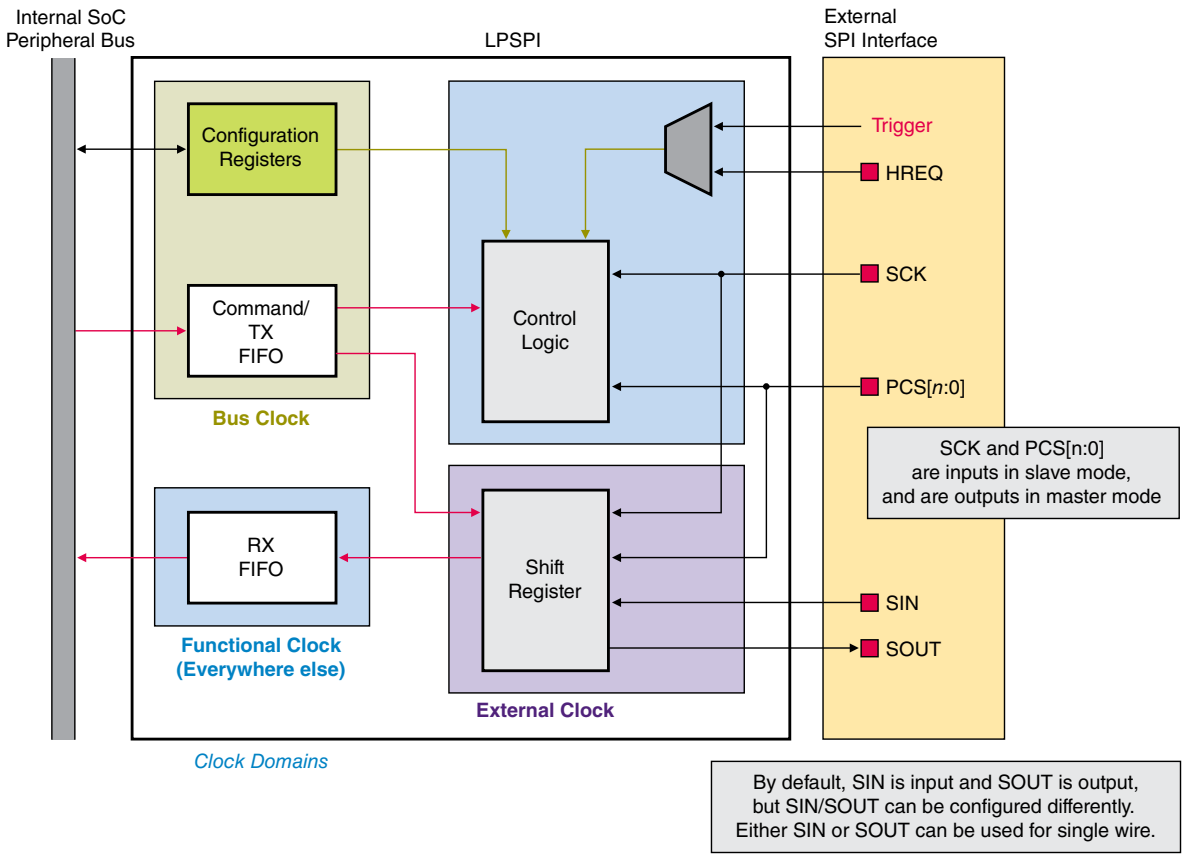


Figure 48-1. Block diagram

48.2.2 Features

LPSPI supports:

- Uses little CPU overhead, with DMA offloading of FIFO register accesses
- Continues operating in stop modes, if configured to do so and an appropriate clock is available
- Supports DMA accesses and generates a DMA request
- Word size of 32 bits
- Configurable clock polarity and clock phase
- Master operation supporting up to 4 peripheral chip selects
- Slave operation
- Command/transmit FIFO of 16 words
- Receive FIFO of 16 words
- Flexible timing parameters in Master mode, including SCK frequency and delays between PCS and SCK edges

- Full duplex transfers supporting 1-bit transmit and receive on each clock edge
- Half duplex transfers supporting 1-bit transmit or receive on each clock edge
- Half duplex transfers supporting 2-bit transmit or receive on each clock edge (master only)
- Half duplex transfers supporting 4-bit transmit or receive on each clock edge (master only)
- Host request can be used to control the start of a SPI bus transfer (master only)
- Receive data match logic supporting wakeup on data match

48.3 Functional description

48.3.1 Master mode

48.3.1.1 Transmit and Command FIFO commands

The transmit and command FIFO is a combined FIFO that includes both transmit data words and command words.

- Transmit data words are stored to the transmit/command FIFO, by writing the Transmit Data (TDR).
- Command words are stored to the transmit/command FIFO, by writing the Transmit Command (TCR).

When a command word is at the top of the transmit/command FIFO, the actions that can occur depend upon whether the LPSPI module is either busy or between frames. See Continuous Transfer bit, TCR[CONT] and Continuing Command bit, TCR[CONTC]

Table 48-2. Possible actions when a command word is at the top of the transmit/command FIFO

Condition	Action
If LPSPI is between frames	then the command word is pulled from the FIFO, and that command word controls all subsequent transfers.
If LPSPI is busy and the Continuing Command bit (TCR[CONTC]) is cleared	then the SPI frame completes at the end of the existing word, ignoring the FRAMESZ configuration. The command word is then pulled from the FIFO and that command word controls all subsequent transfers (or until the next update to the command word). Note that a command word with TCR[CONTC] = 0 always terminates (stop) the existing transfer regardless of the previous TCR[CONT] value.
If LPSPI is busy and the existing TCR[CONT] bit is set and the new TCR[CONTC] value is set	then the command word must be updated at the frame boundary. The command word is pulled from the FIFO during the last SCK pulse of the existing frame (based on the FRAMESZ configuration), and the frame continues using the new command value for the rest of the frame (or until the next update to the command word).

Table 48-2. Possible actions when a command word is at the top of the transmit/command FIFO

Condition	Action
	When TCR[CONTC] is set, only the lower 24-bits of the command word are updated. If the command word is updated at a word boundary, then the transfer halts (stops) after that word. Essentially the TCR[CONTC] bit is ignored when not at a frame boundary, so the frame ends prematurely.

About TCR[CONT] and TCR[CONTC] :

- TCR[CONT] = 1 is used to keep PCS asserted at end of frame, allowing the transfer to continue.
- TCR[CONTC] = 1 is used to indicate that this command word should not terminate the existing frame, and the transfer can continue using the new command word.

TCR[CONTC] = 1 is restricted in the sense that the new command must load on a frame boundary, and the only way for a transfer to continue from a frame boundary, is when the previous command has TCR[CONT] = 1.

The current state of the existing command word is read from Transmit Command (TCR). It requires at least 3 LPSPI functional clock cycles for TCR to update after TCR is written (assuming an empty FIFO) and LPSPI must be enabled (CR[MEN] bit is set).

Writing the TCR does not initiate a SPI bus transfer, unless the TCR[TXMSK] = 1. When the TCR[TXMSK] bit is set, a new command word is not be loaded until the end of the existing frame (based on FRAMESZ configuration); at the end of the transfer, the TCR[TXMSK] bit clears.

In master mode, the LPSPI command word in TCR controls SPI attributes (using bits and fields in registers).

Table 48-3. LPSPI Command Word in Master mode

Transmit Command (TCR)		Description	Can this field be modified during a data transfer?
Field	Name		
CPOL	Clock Polarity	Configures the polarity of the SCK pin. Any change of CPOL value will cause a transition on the SCK pin.	N
CPHA	Clock Phase	Configures the clock phase of the transfer.	N
PRESCALE	Prescaler Value	Configures a prescaler used to divide the LPSPI functional clock, to generate the timing parameters of the SPI bus transfer. Changing PRESCALE in conjunction with PCS enables the LPSPI module to connect to different slave devices at different frequencies.	N

Table continues on the next page...

Table 48-3. LPSPI Command Word in Master mode (continued)

Transmit Command (TCR)		Description	Can this field be modified during a data transfer?
Field	Name		
PCS	Peripheral Chip Select	Configures which PCS asserts for the transfer; the polarity of PCS is static and configured by CFGR1[PCSPOL]. If CFGR1[PCSCFG] = 1. then PCS[3:2] should not be selected.	N
LSBF	LSB First	Configures if LSB (bit 0) or MSB (bit 31 for a 32-bit word) is transmitted or received first.	Y
BYSW	Byte Swap	Enables byte swap on each 32-bit word when transmitting and receiving data. Byte swapping can be useful when interfacing to devices that organize data as big-endian.	Y
CONT	Continuous Transfer	Configures for a continuous transfer that keeps PCS asserted between frames (as configured by FRAMESZ). A new command word is required to cause PCS to negate. Also supports changing the command word at the frame size boundaries.	Y
CONTC	Continuing Command	Indicates that this is a new command word for the existing continuous transfer. The CONTC bit when set must only be written to the transmit/ command FIFO on a frame boundary.	Y
RXMSK	Receive Data Mask	Masks the receive data and does not store the masked receive data to the receive FIFO or perform receive data matching. Useful for half-duplex transfers or to configure which fields are compared during receive data matching.	Y
TXMSK	Transmit Data Mask	Masks the transmit data, so that masked transmit data is not pulled from transmit FIFO, and the output data pin is tristated (unless configured by CFGR1[OUTCFG]). Useful for half-duplex transfers.	Y
WIDTH	Transfer Width	Configures the number of bits shifted on each SCK pulse. • 1-bit transfers support traditional SPI bus transfers in either half-duplex or full-duplex data formats. • 2-bit and 4-bit half-duplex transfers are useful for interfacing to QuadSPI memory devices, and at least one bit (TCR[TXMSK] or TCR[RXMSK]) must also be set.	Y
FRAMESZ	Frame Size	Configures the frame size in number of bits equal to (FRAMESZ + 1). • The minimum frame size is 8 bits. • The minimum word size is 2 bits; a frame size of 33 bits (or similar) is not supported. • If the frame size is larger than 32 bits, then the frame is divided into multiple words of 32-bits; each word is loaded from the transmit FIFO and stored in the receive FIFO separately. • If the size of the frame is not divisible by 32, then the last load of the transmit FIFO and store of the receive FIFO will contain the remainder bits. For example, a 72-bit transfer will consist of 3 words: the 1st and 2nd words are 32-bit, and the 3rd word is 8-bit.	Y

SPI bus transfers:

- LPSPI initiates a SPI bus transfer when all conditions are true:
  - Data is written to the transmit FIFO
  - And the HREQ pin is asserted (or the HREQ function is disabled)
  - And LPSPI is enabled

- To perform the SPI bus transfer, LPSPI uses the attributes configured in Transmit Command (TCR) and uses the timing parameters in Clock Configuration (CCR).
- The SPI bus transfer ends after the FRAMESZ configuration is reached, or at the end of a word when a new transmit command word is at the top of the transmit/command FIFO.
- The HREQ input is only checked on the next time that LPSPI goes idle (LPSPI completes the current transmit FIFO and there are no attribute updates in TCR.

Circular FIFO: The transmit/command FIFO also supports a Circular FIFO feature, which enables the LPSPI master to (periodically) repeat a short data transfer that fits within the transmit/command FIFO, without requiring additional FIFO accesses. When the circular FIFO is enabled (CFGR0[CIRFIFO] = 1), the current state of the FIFO read pointer is saved and the status flags do not update. After the transmit/command FIFO is considered empty and LPSPI is idle, the FIFO read pointer is restored with the saved version, so the contents of the transmit/command FIFO are not permanently pulled from the FIFO while circular FIFO mode is enabled.

48.3.1.2 Receive FIFO and Data Match

The receive FIFO stores receive data during SPI bus transfers. When TCR[RXMSK] = 1, the receive data is discarded instead of being stored in the receive FIFO.

- The receive data is written to the receive FIFO at the end of the frame.
- During a multiple word or continuous transfer, the receive data is also written to the receive FIFO at the same time as the new transmit data is read from the transmit FIFO.
- During a continuous transfer, if the transmit FIFO is empty, then the receive data is only written to the receive FIFO after the transmit FIFO is written or after TCR is written to end the frame.

Receive data supports a receive data match function that can match received data against one of two words in the DMR0 and DMR1 registers or against a masked data word. The receive data match function can also be configured to compare only the first one or two received data words since the start of the frame.

- Received data that is already discarded due to the TCR[RXMSK] bit cannot cause the data match to set, and delays the receive data match on the first received data word, until all discarded data is received.
- The receive data match function can also be configured to discard all received data until a data match is detected, using the CFGR0[RDMO] bit.
- After a receive data match, to allow all subsequent data to be received, clear the CFGR0[RDMO] bit, then clear the SR[DMF] bit.

48.3.1.3 Timing parameters

The timing parameters that are used for all SPI bus transfers are relative to the LPSPI functional clock divided by the PRESCALE configuration. Although the Clock Configuration (CCR) cannot be changed when the LPSPI module is busy, to support interfacing to different slave devices at different frequencies, the TCR[PRESCALE] configuration can be changed between SPI bus transfers using the Transmit Command Register (TCR).

NOTE

The minimum value below is the minimum counter value, but the clock configuration register values must also satisfy the datasheet specs based on the LPSPI functional clock frequency and prescaler value.

Table 48-4. LPSPI timing parameters

Clock Configuration (CCR)		Description	Min	Max
Field	Name			
SCKDIV	SCK Divider	Configures the SCK clock period to (SCKDIV + 2) cycles. When SCK Divider is configured to an odd number of cycles, the first half of the SCK cycle is one cycle longer than the second half of the SCK cycle.	0 (2 cycles)	255 (257 cycles)
DBT	Delay Between Transfers (PCS negation to PCS assertion)	Configures the minimum delay between PCS negation and the next PCS assertion to (DBT + 2) cycles. When the command word is updated between transfers, there is a minimum of (DBT/2) + 1 cycles between the command word update and any change on PCS pins.	0 (2 cycles)	255 (257 cycles)
DBT	Delay Between Transfers (last SCK edge to first SCK edge)	Configures the delay during a continuous transfer between the last SCK edge of a frame and the first SCK edge of the continuing frame to (DBT + 1) cycles. This is useful when the external slave requires a large delay between different words of an SPI bus transfer.	0 (1 cycle)	255 (256 cycles)
PCSSCK	PCS-to-SCK Delay	Configures the minimum delay between PCS assertion and the first SCK edge to (PCSSCK + 1) cycles.	0 (1 cycle)	255 (256 cycles)
SCKPCS	SCK-to-PCS Delay	Configures the minimum delay between the last SCK edge and the PCS assertion to (SCKPCS + 1) cycles.	0 (1 cycle)	255 (256 cycles)

This figure shows the timing settings controlled by TCR[CPHA], TCR[CPOL], CCR[SCKPCS], andCCR[PCSSCK].

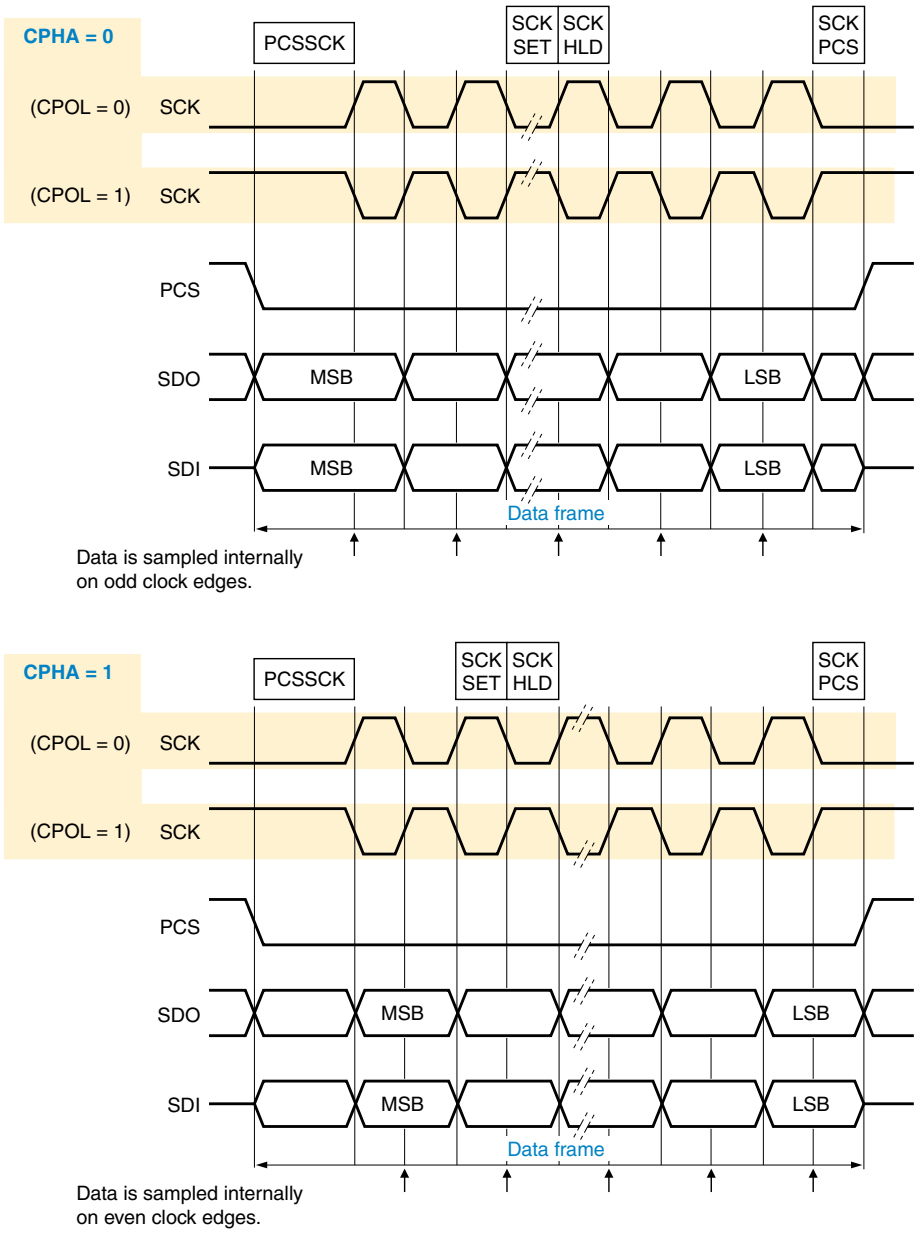


Figure 48-2. Clock Phase (TCR[CPHA]) timing diagram example

48.3.1.4 Pin configuration

- To swap directions or support half-duplex transfers on the same pin, the SIN and SOUT pins can be configured using CFGR1[PINCFG].
- To determine if an output data pin (like SOUT) tristates when PCS is negated, or if the output data pin simply retains the last value, use CFGR1[PCSCFG].
- When configuring for half-duplex transfers the output data pins must be configured to tristate when PCS is negated; use CFGR1[OUTCFG] CFGR1[OUTCFG].

- When performing half-duplex 2-bit transfers, CFGR1[PCSCFG] can be set to any value.
- When performing half-duplex 4-bit transfers, CFGR1[PCSCFG] must equal 0x1.

48.3.1.5 Clock loopback

Configure the LPSPI master to use either the output clock or a loopback clock to sample the receive data (for example, SIN). Use one of these these clocks to sample the input data:

- The SCK output clock directly
- A delayed version of the SCK output clock

The delayed version of the SCK is chosen by the SCK pin output delay, plus the SCK pin input delay, and is configured by setting CFGR1[SAMPLE]. Enabling the loopback version of the SCK can improve the setup time of the input data from the slave.

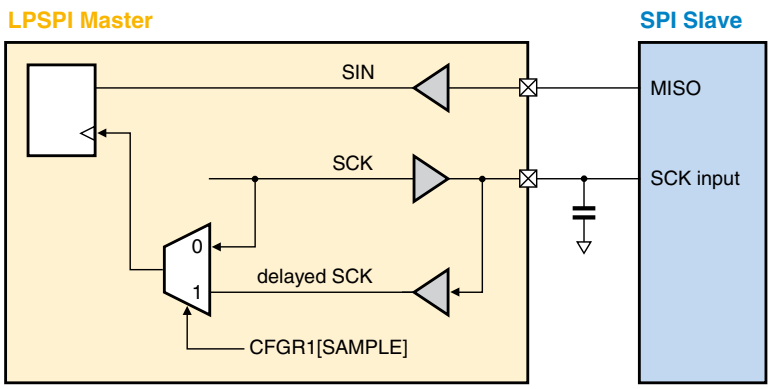


Figure 48-3. Clock loopback

See the chip data sheet for the specific input setup time in master loopback mode.

48.3.2 Slave mode

LPSPI slave mode:

- Uses the same shift register and logic that master mode uses
- Does not use the Clock Configuration Register (CCR)
- During SPI bus transfers, requires that the Transmit Command (TCR) register remain static (unchanging)

48.3.2.1 Transmit and Command FIFO commands

Before enabling LPSPI in slave mode, initialize the Transmit Command (TCR) register, although the TCR register does not update until after LPSPI is enabled. After being enabled, the TCR register should only be changed if LPSPI is idle. In slave mode, the LPSPI command word in TCR controls SPI attributes. Before the PCS input asserts, the transmit FIFO must be filled with transmit data, or the transmit error flag sets.

Table 48-5. LPSPI Command Word in Slave mode

Transmit Command (TCR)		Description
Field	Name	
CPOL	Clock Polarity	Configures the polarity of the external SCK input.
CPHA	Clock Phase	Configures the clock phase of transfer.
PRESCALE	Prescaler Value	Configures the LPSPI functional clock prescaler.
PCS	Peripheral Chip Select	Configures which PCS is used, the polarity of PCS is static and configured by CFGR1[PCSPOL]. If CFGR1[PCSCFG] is not equal to zero, then PCS[3:2] pins should not be selected.
LSBF	LSB First	Configures if LSB (bit 0) or MSB (bit 31 for a 32-bit word) is transmitted or received first.
BYSW	Byte Swap	Enables byte swap on each 32-bit word when transmitting and receiving data. Byte swapping can be useful when interfacing to devices that organize data as big-endian.
CONT	Continuous Transfer	When Continuous Transfer is set in slave mode and after the first FRAMESZ bits are transferred, the LPSPI will passthrough and transmit the received data until the next PCS negation. Whatever is shifted in on the receive data is shifted out as transmit data as if there was a 32-bit shift register.
CONTC	Continuing Command	CONTC bit is reserved (in slave mode).
RXMSK	Receive Data Mask	Masks the receive data and does not store the masked receive data to the receive FIFO or perform receive data matching. Useful for half-duplex transfers or to configure which fields are compared during receive data matching.
TXMSK	Transmit Data Mask	Masks the transmit data, so that the masked transmit data is not pulled from transmit FIFO, and the output data pin is tristated (unless configured by CFGR1[OUTCFG]). Useful for half-duplex transfers.
WIDTH	Transfer Width	Configures the number of bits shifted on each SCK pulse. • 1-bit transfers support traditional SPI bus transfers in either half-duplex or full-duplex data formats.
FRAMESZ	Frame Size	Configures the frame size in number of bits equal to (FRAMESZ + 1). • The minimum frame size is 8 bits. • The minimum word size is 2 bits; a frame size of 33 bits (or similar) is not supported. • If the frame size is larger than 32 bits, then the frame is divided into multiple words of 32-bits; each word is loaded from the transmit FIFO and stored in the receive FIFO separately. • If the size of the frame is not divisible by 32, then the last load of the transmit FIFO and store of the receive FIFO will contain the remainder bits. For example, a 72-bit transfer will consist of 3 words: the 1st and 2nd words are 32-bit, and the 3rd word is 8-bit.

48.3.2.2 Receive FIFO and data match

The receive FIFO stores receive data during SPI bus transfers. When TCR[RXMSK] = 1, the received data is discarded instead of storing the received data in the receive FIFO.

Receive data supports a receive data match function that can match received data against one of two words in the DMR0 and DMR1 registers or against a masked data word. The data match function can also be configured to compare only the first one or two received data words since the start of the frame.

- Received data that is already discarded because TCR[RXMSK] = 1 cannot cause the data match to set, and delays the match on the first received data word, until all discarded data is received.
- The receiver match function can also be configured to discard all received data until a data match is detected, using the CFGR0[RDMO] bit.
- After a receive data match, to allow all subsequent data to be received, first clear the CFGR0[RDMO] bit, then clear the SR[DMF] bit.

48.3.2.3 Clocked interface

The LPSPI module supports interfacing to external masters that provide only clock and data pins (PCS is not required). This interface requires:

- Using Clock Phase TCR[CPHA] = 1 (data is changed on the leading edge of SCK and captured on the following edge)
- Configuring the PCS input to be always asserted (CFGR1[PCSPOLn] = 1). For example, to configure PCS[0] to be always asserted, set PCSPOL[0] = 1, and do not configure PCS[0] in the pin muxing. The chip-level drives PCS to a certain value (ideally 1), CFGR1[PCSPOLn] could be used to invert that value.
- Setting CFGR1[AUTOPCS] = 1 (Automatic PCS generation is enabled). When CFGR1[AUTOPCS] = 1, a minimum of 4 LPSPI functional clock cycles (divided by PRESCALE configuration) is required between the last SCK edge of one word and the first SCK edge of the next word.

48.3.3 Low power modes

Table 48-6. Low power modes

Chip mode	LPSPI operation
Run	Normal operation
Stop	Can continue operating in stop mode if CR[DOZEN] = 0 and LPSPI is using an external or internal clock source that remains operating during stop mode.

48.3.4 Debug mode

Table 48-7. Debug mode

Chip mode	LPSPI operation
Debug (the core is in Debug/Halted mode)	Can continue operating in Debug mode, if the CR[DBGEN] = 1.

48.3.5 Interrupts and DMA Requests

The next table lists the slave mode sources (status flags) that can generate LPSPI interrupts and LPSPI slave transmit/receive DMA requests.

Table 48-8. LPSPI Interrupts and DMA Requests

Status (SR)		Description	Can generate		
Status Flag	Name		Interrupt?	DMA Request?	Low Power Wakeup?
TDF	Transmit Data Flag	Data can be written to transmit FIFO, as configured by the Transmit FIFO Watermark FCR[TXWATER]	Y	TX	Y
RDF	Receive Data Flag	Data can be read from the receive FIFO, as configured by the Receive FIFO Watermark FCR[RXWATER]	Y	RX	Y
WCF	Word Complete Flag	Word is complete, the last bit of the word has been sampled	Y	N	Y
FCF	Frame Complete Flag	Frame is complete, and PCS has negated	Y	N	Y
TCF	Transfer Complete Flag	Transfer is complete, PCS has negated, and the transmit/command FIFO is empty	Y	N	Y
TEF	Transmit Error Flag	Indicates a transmit/command FIFO underrun. In master mode when CFGR1[NOSTALL] = 0 (transfers stall when transmit FIFO is empty), the Transmit Error Flag bit cannot set.	Y	N	Y
REF	Receive Error Flag	Receive error flag, indicates a receive FIFO overflow.	Y	N	Y
DMF	Data Match Flag	Indicates that the received data has matched the configured data match value	Y	N	Y
MBF	Module Busy Flag	LPSPI is busy performing a SPI bus transfer	N	N	N

48.3.6 Clocks

Table 48-9. LPSPI Clocks

LPSPI Functional clock	• The LPSPI functional clock is asynchronous to the bus clock. • If the LPSPI functional clock remains enabled in low power modes, then LPSPI can perform SPI bus transfers and low power wakeups, in both master and slave modes. • The LPSPI divides the functional clock by a prescaler; the resulting frequency must be at least 2 times faster than the SPI external clock frequency (SCK).

External clock	• The LPSPI shift register is clocked directly by the SCK clock.
	• How the SCK clock is generated or supplied depends upon the mode (master or slave):
	  • In master mode: the SCK clock is generated internally.
	  • In slave mode: the SCK clock is supplied externally.

Bus clock	The bus clock is only used for bus accesses to the LPSPI control and configuration registers. The bus clock frequency must be high enough to support the data bandwidth requirements of the LPSPI registers, including the FIFOs.

For chip-specific clocking information, see the Clocking chapter.

48.3.7 Resets

Table 48-10. Resets

Chip reset	Resets the LPSPI logic and registers to their default state.
Software reset	• Resets the LPSPI logic and registers to their default state, except for the Control Register. • The LPSPI software reset is in the Control Register CR[RST].
FIFO resets	• Resets the transmit/command FIFO and the receive FIFO. • The Reset Transmit FIFO field, CR[RTF] and the Reset Receive FIFO field, CR[RRF] are write-only bits. • After being reset, a FIFO is empty.

48.3.8 Peripheral Triggers

The connection of the LPSPI peripheral triggers to other peripherals depend upon the specific device being used.

Table 48-11. LPSPI Triggers

Trigger	Description	Notes
Frame Output Trigger	The frame output trigger is an idle high signal, which: • asserts at the end of each frame (when PCS negates) • remains asserted until PCS next asserts	LPSPI generates 2 output triggers that can be connected to other peripherals on the device.
Word Output Trigger	The word output trigger:	

Table continues on the next page...

Table 48-11. LPSPI Triggers (continued)

Trigger	Description	Notes
	• asserts at the end of each received word • remains asserted for 1 SCK period	
Input Trigger	To control the start of a LPSPI bus transfer, the LPSPI input trigger can be selected instead of the HREQ input. • The LPSPI input trigger is synchronized, and must assert for at least 2 cycles of the LPSPI functional clock divided by the PRESCALE configuration, so that the input trigger can be detected. • When the LPSPI module is busy, the HREQ input (and therefore the LPSPI input trigger) is ignored. • Both HREQ and the LPSPI input trigger are ignored when the module is busy. They are used to start a new transfer when the module is idle.	

48.4 Signal descriptions

Table 48-12. Signals

Signal	Name	Description	I/O
SCK	Serial clock	• Input in slave mode • Output in master mode	I/O
PCS[0]	Peripheral Chip Select	• Input in slave mode • Output in master mode	I/O
PCS[1] / HREQ	Peripheral Chip Select or Host Request	Host Request pin is selected when CFGR0[HREN] = 1 and CFGR0[HRSEL] = 0 • Input in either slave mode or when used as master Host Request • Output in master mode	I/O
PCS[2] / DATA[2]	Peripheral Chip Select or data pin 2 during parallel data transfers	When CFGR1[PCSCFG] = 0: • Input in slave mode • Output in master mode When CFGR1[PCSCFG] = 1: • Input in half-duplex parallel data receive transfers • Output in half-duplex parallel data transmit transfers	I/O
PCS[3] / DATA[3]	Peripheral Chip Select or data pin 3 during parallel data transfers	When CFGR1[PCSCFG] = 0: • Input in slave mode • Output in master mode When CFGR1[PCSCFG] = 1: • Input in half-duplex parallel data receive transfers • Output in half-duplex parallel data transmit transfers	I/O
SOUT / DATA[0]	Serial Data Output	Can be configured as serial data input signal	I/O

Table continues on the next page...

Table 48-12. Signals (continued)

Signal	Name	Description	I/O
		• Used as data pin 0 in half-duplex parallel data transfers	
SIN / DATA[1]	Serial Data Input	Can be configured as serial data output signal • Used as data pin 1 in half-duplex parallel data transfers	I/O

48.5 Memory map and registers

NOTE

Writing a Read-Only (RO) register or reading a Write-Only (WO) register can cause bus errors. LPSPI does not check if programmed values in the registers are correct, so application software must take care to write valid values only.

48.5.1 LPSPI register descriptions

48.5.1.1 LPSPI memory map

LPSPI1 base address: 4039_4000h SPI2

LPSPI2 base address: 4039_8000h

LPSPI3 base address: 4039_C000h SPI1

LPSPI4 base address: 403A_0000h SPI

Offset	Register	Width (In bits)	Access	Reset value
0h	Version ID (VERID)	32	RO	0102_0004h
4h	Parameter (PARAM)	32	RO	0004_0404h
10h	Control (CR)	32	RW	0000_0000h
14h	Status (SR)	32	W1C	0000_0001h
18h	Interrupt Enable (IER)	32	RW	0000_0000h
1Ch	DMA Enable (DER)	32	RW	0000_0000h
20h	Configuration 0 (CFGR0)	32	RW	0000_0000h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
24h	Configuration 1 (CFGR1)	32	RW	0000_0000h
30h	Data Match 0 (DMR0)	32	RW	0000_0000h
34h	Data Match 1 (DMR1)	32	RW	0000_0000h
40h	Clock Configuration (CCR)	32	RW	0000_0000h
58h	FIFO Control (FCR)	32	RW	0000_0000h
5Ch	FIFO Status (FSR)	32	RO	0000_0000h
60h	Transmit Command (TCR)	32	RW	0000_001Fh
64h	Transmit Data (TDR)	32	WO	0000_0000h
70h	Receive Status (RSR)	32	RO	0000_0002h
74h	Receive Data (RDR)	32	RO	0000_0000h

48.5.1.2 Version ID (VERID)

48.5.1.2.1 Offset

Register	Offset
VERID	0h

48.5.1.2.2 Function

Contains version numbers for the module design and feature set.

48.5.1.2.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	MAJOR								MINOR							
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FEATURE															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

48.5.1.2.4 Fields

Field	Function
31-24 MAJOR	Major Version Number Returns the major version number for the module specification. Read-only field.
23-16 MINOR	Minor Version Number Returns the minor version number for the module specification. Read-only field.
15-0 FEATURE	Module Identification Number Returns the feature set number. Read-only field. 0000000000000100b - Standard feature set supporting a 32-bit shift register.

48.5.1.3 Parameter (PARAM)

48.5.1.3.1 Offset

Register	Offset
PARAM	4h

48.5.1.3.2 Function

Contains parameter values that were implemented in the module.

48.5.1.3.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0								PCSNUM							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RXFIFO								TXFIFO							
W																
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0

48.5.1.3.4 Fields

Field	Function
31-24	Reserved

Table continues on the next page...

Field	Function
—	
23-16 PCSNUM	PCS Number Sets the number of PCS pins supported by the peripheral.
15-8 RXFIFO	Receive FIFO Size Indicates the maximum number of words in the receive FIFO, which is 2^{RXFIFO}.
7-0 TXFIFO	Transmit FIFO Size Indicates the maximum number of words in the transmit FIFO, which is 2^{TXFIFO}.

48.5.1.4 Control (CR)

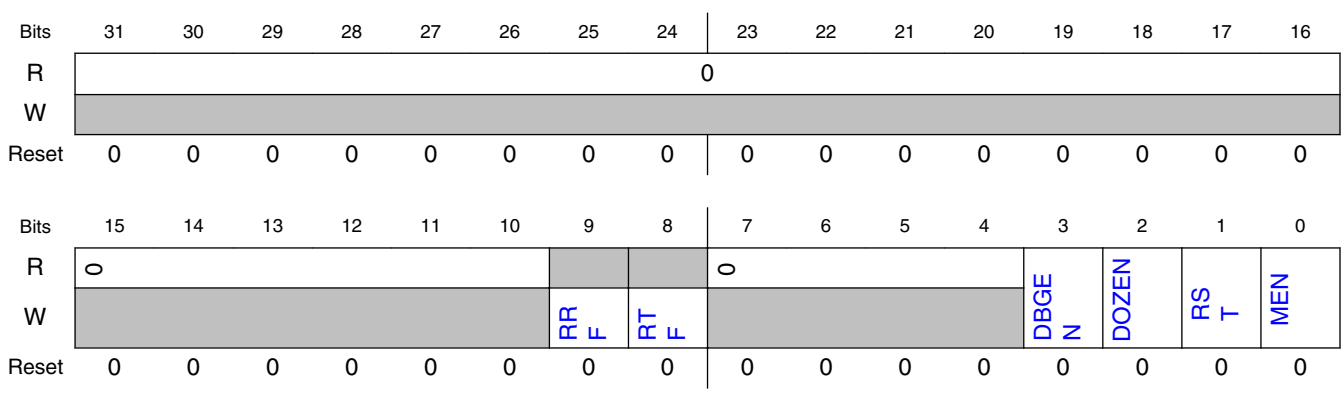
48.5.1.4.1 Offset

Register	Offset
CR	10h

48.5.1.4.2 Function

Contains fields associated with the module operation.

48.5.1.4.3 Diagram



48.5.1.4.4 Fields

Field	Function
31-10	Reserved

Table continues on the next page...

Field	Function
—	
9 RRF	Reset Receive FIFO 0b - No effect 1b - Reset the Receive FIFO. The register bit always reads zero.
8 RTF	Reset Transmit FIFO 0b - No effect 1b - Reset the Transmit FIFO. The register bit always reads zero.
7-4 —	Reserved
3 DBGEN	Debug Enable Enables or disables the LPSPI module in debug mode. Debug Enable bit should be updated only when the LPSPI module is disabled. 0b - LPSPI module is disabled in debug mode 1b - LPSPI module is enabled in debug mode
2 DOZEN	Doze Mode Enable Enables or disables the LPSPI module in Doze mode. Doze Mode Enable bit should be updated only when the LPSPI module is disabled. 0b - LPSPI module is enabled in Doze mode 1b - LPSPI module is disabled in Doze mode
1 RST	Software Reset Reset all internal logic and registers, except the Control Register. RST remains set until cleared by software. The reset takes effect immediately and remains asserted until negated by software. There is no minimum delay required before clearing the software reset. 0b - Module is not reset 1b - Module is reset
0 MEN	Module Enable 0b - Module is disabled 1b - Module is enabled

48.5.1.5 Status (SR)

48.5.1.5.1 Offset

Register	Offset
SR	14h

48.5.1.5.2 Function

Contains the status of data flow.

48.5.1.5.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0							MBF	0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0		DMF	REF	TEF	TCF	FCF	WCF	0						RDF	TDF
W			W1C	W1C	W1C	W1C	W1C	W1C								
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

48.5.1.5.4 Fields

Field	Function
31-25 —	Reserved
24 MBF	Module Busy Flag In Master mode, MBF asserts when there is data to transmit and LPSPI is able to transmit (eg: HREQ asserted, etc). It negates after the PCS negates and the LPSPI master has waited half the DBT time with no new data to transmit. Slave mode asserts MBF when LPSPI is enabled and PCS is asserted. 0b - LPSPI is idle 1b - LPSPI is busy
23-14 —	Reserved
13 DMF	Data Match Flag Indicates that the received data has matched the DMR0[MATCH0] and/or DMR1[MATCH1] fields (as configured by CFGR1[MATCFG]. 0b - Have not received matching data 1b - Have received matching data
12 REF	Receive Error Flag The Receive Error Flag sets when the Receiver FIFO overflows. When the Receive Error Flag is set, it is recommended to first end the transfer, empty the Receive FIFO, clear the Receive Error Flag and then restart the transfer from the beginning. 0b - Receive FIFO has not overflowed 1b - Receive FIFO has overflowed
11 TEF	Transmit Error Flag The Transmit Error Flag sets when the Transmit FIFO underruns. When the Transmit Error Flag is set, it is recommended to first end the transfer, clear the Transmit Error Flag and then restart the transfer from the beginning. 0b - Transmit FIFO underrun has not occurred 1b - Transmit FIFO underrun has occurred
10 TCF	Transfer Complete Flag In Master mode when LPSPI returns to idle state with the transmit FIFO empty, the Transfer Complete Flag sets. 0b - All transfers have not completed

Table continues on the next page...

Field	Function
	1b - All transfers have completed
9 FCF	Frame Complete Flag The Frame Complete Flag sets at the end of each frame transfer, when the PCS negates. 0b - Frame transfer has not completed 1b - Frame transfer has completed
8 WCF	Word Complete Flag The Word Complete Flag sets when the last bit of a received word is sampled. 0b - Transfer of a received word has not yet completed 1b - Transfer of a received word has completed
7-2 —	Reserved
1 RDF	Receive Data Flag The Receive Data Flag is set whenever the number of words in the receive FIFO is greater than the value in FCR[RXWATER]. 0b - Receive Data is not ready 1b - Receive data is ready
0 TDF	Transmit Data Flag The Transmit Data Flag is set whenever the number of words in the transmit FIFO is equal or less than the value in FCR[TXWATER]. 0b - Transmit data not requested 1b - Transmit data is requested

48.5.1.6 Interrupt Enable (IER)

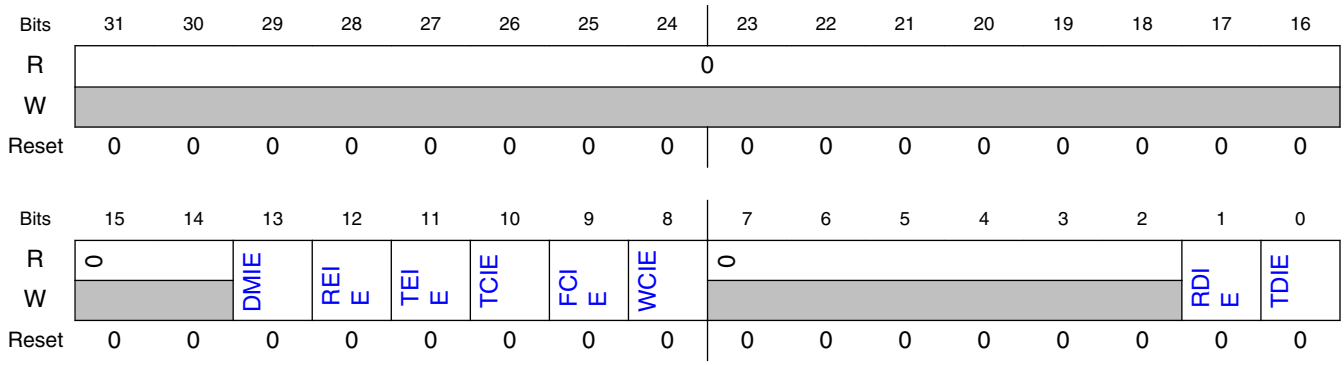
48.5.1.6.1 Offset

Register	Offset
IER	18h

48.5.1.6.2 Function

Enables interrupts based on data flow and errors.

48.5.1.6.3 Diagram



48.5.1.6.4 Fields

Field	Function
31-14 —	Reserved
13 DMIE	Data Match Interrupt Enable 0b - Disabled 1b - Enabled
12 REIE	Receive Error Interrupt Enable 0b - Disabled 1b - Enabled
11 TEIE	Transmit Error Interrupt Enable 0b - Disabled 1b - Enabled
10 TCIE	Transfer Complete Interrupt Enable 0b - Disabled 1b - Enabled
9 FCIE	Frame Complete Interrupt Enable 0b - Disabled 1b - Enabled
8 WCIE	Word Complete Interrupt Enable 0b - Disabled 1b - Enabled
7-2 —	Reserved
1 RDIE	Receive Data Interrupt Enable 0b - Disabled 1b - Enabled
0 TDIE	Transmit Data Interrupt Enable 0b - Disabled 1b - Enabled

48.5.1.7 DMA Enable (DER)

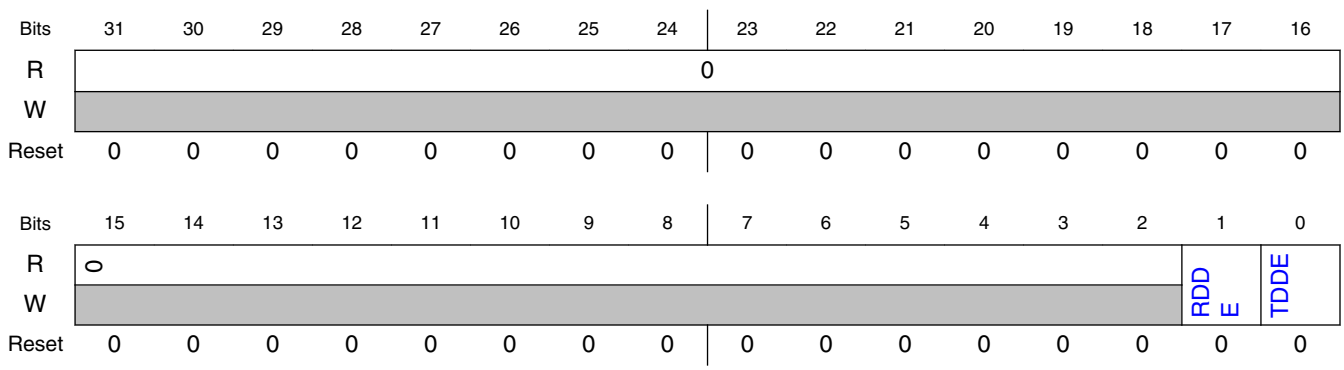
48.5.1.7.1 Offset

Register	Offset
DER	1Ch

48.5.1.7.2 Function

Enables DMA data flow.

48.5.1.7.3 Diagram



48.5.1.7.4 Fields

Field	Function
31-2 —	Reserved
1 RDDE	Receive Data DMA Enable 0b - DMA request is disabled 1b - DMA request is enabled
0 TDDE	Transmit Data DMA Enable 0b - DMA request is disabled 1b - DMA request is enabled

48.5.1.8 Configuration 0 (CFGR0)

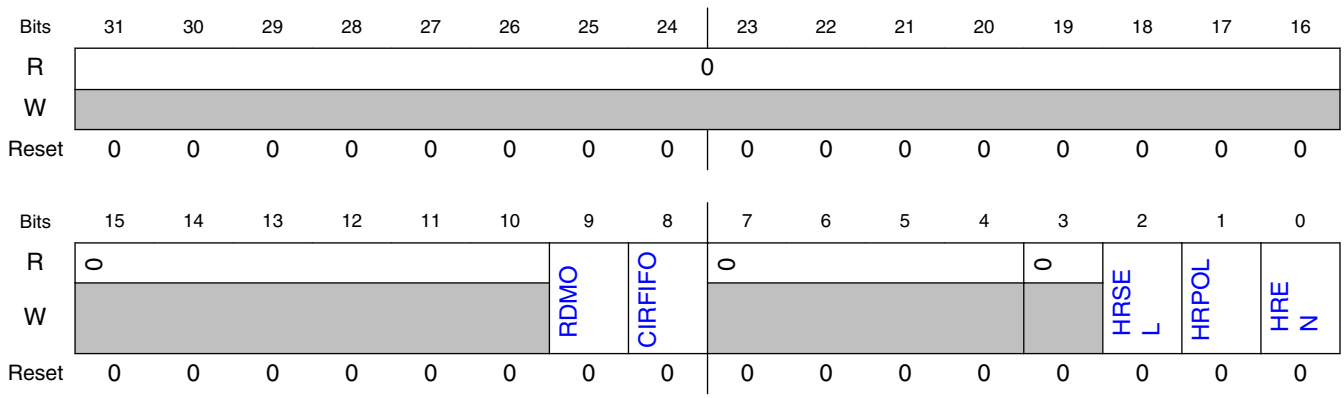
48.5.1.8.1 Offset

Register	Offset
CFGR0	20h

48.5.1.8.2 Function

Includes additional fields to configure LPSPI including Circular FIFO Enable and Receive Data Match Only.

48.5.1.8.3 Diagram



48.5.1.8.4 Fields

Field	Function
31-10 —	Reserved
9 RDMO	Receive Data Match Only When RDMO is enabled, all received data that does not cause the Data Match Flag (SR[DMF]) to set is discarded. • Set the RDMO bit when LPSPI is idle and SR[DMF] = 0. • After SR[DMF] = 1, the RDMO bit configuration is ignored. • When disabling RDMO and to ensure that no receive data is lost, before clearing SR[DMF], first clear RDMO. 0b - Received data is stored in the receive FIFO as in normal operations 1b - Received data is discarded unless the SR[DMF] = 1
8 CIRFIFO	Circular FIFO Enable When enabled, the transmit FIFO read pointer is saved to a temporary register. The transmit FIFO is emptied as it normally is, but after LPSPI is idle and the transmit FIFO is empty, then the read pointer value is restored from the temporary register. This restoring of the read pointer causes the contents of the transmit FIFO to be cycled through repeatedly.

Table continues on the next page...

Field	Function
	NOTE: The read pointer is restored for as long as the CIRFIFO is set. Writing additional words to the FIFO after setting CIRFIFO adds them to the end of the FIFO, up to the size of the transmit FIFO. 0b - Circular FIFO is disabled 1b - Circular FIFO is enabled
7-4 —	Reserved
3 —	Reserved
2 HRSEL	Host Request Select Selects the source of the host request input. When the host request function is enabled with the HREQ pin, the PCS[1] function is disabled. 0b - Host request input is the HREQ pin 1b - Host request input is the input trigger
1 HRPOL	Host Request Polarity Configures the polarity of the host request pin. 0b - HREQ pin is active high provided PCSPOL[1] is clear 1b - HREQ pin is active low provided PCSPOL[1] is clear
0 HREN	Host Request Enable When enabled in Master mode, LPSPI only starts a new SPI bus transfer if the host request input is asserted. When LPSPI is busy, the host request input is ignored. 0b - Host request is disabled 1b - Host request is enabled

48.5.1.9 Configuration 1 (CFGR1)

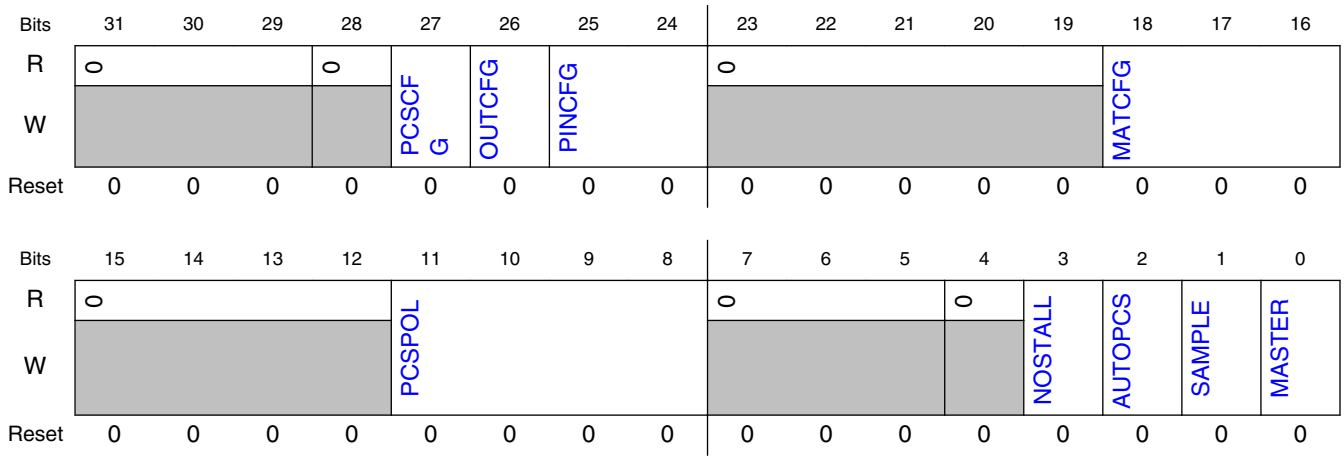
48.5.1.9.1 Offset

Register	Offset
CFGR1	24h

48.5.1.9.2 Function

CFGR1 should only be written when LPSPI is disabled.

48.5.1.9.3 Diagram



48.5.1.9.4 Fields

Field	Function
31-29 —	Reserved
28 —	Reserved
27 PCSCFG	Peripheral Chip Select Configuration When performing parallel transfers, PCSCFG must be configured to enable the desired transfer. 0b - PCS[3:2] are configured for chip select function 1b - PCS[3:2] are configured for half-duplex 4-bit transfers (PCS[3:2] = DATA[3:2])
26 OUTCFG	Output Configuration Configures if the output data is tristated between accesses (PCS is negated). If performing half-duplex transfers, the Output Configuration bit must be set. 0b - Output data retains last value when chip select is negated 1b - Output data is tristated when chip select is negated
25-24 PINCFG	Pin Configuration Configures which pins are used for input and output data during serial transfers. If performing parallel transfers, the Pin Configuration field must be 00. 00b - SIN is used for input data and SOUT is used for output data 01b - SIN is used for both input and output data, only half-duplex serial transfers are supported 10b - SOUT is used for both input and output data, only half-duplex serial transfers are supported 11b - SOUT is used for input data and SIN is used for output data
23-19 —	Reserved
18-16 MATCFG	Match Configuration Configures the condition that causes the DMF to set. **NOTE:** *Syntax:* * is boolean AND, + is boolean OR 000b - Match is disabled 001b - Reserved

Table continues on the next page...

Field	Function
	010b - Match is enabled is 1st data word is MATCH0 or MATCH1. Match is enabled, if 1st data word equals MATCH0 OR MATCH1, that is (*1st data word* = MATCH0 + MATCH1) 011b - Match is enabled on any data word equal MATCH0 or MATCH1. Match is enabled, if any data word equals MATCH0 OR MATCH1, that is (*any data word* = MATCH0 + MATCH1) 100b - Match is enabled on data match sequence. Match is enabled, if 1st data word equals MATCH0 AND 2nd data word equals MATCH1, that is [(*1st data word* = MATCH0) * (*2nd data word* = MATCH1)] 101b - Match is enabled on data match sequence. Match is enabled, if any data word equals MATCH0 AND the next data word equals MATCH1, that is [(*any data word* = MATCH0) * (*next data word* = MATCH1)] 110b - Match is enabled. Match is enabled, if (1st data word AND MATCH1) equals (MATCH0 AND MATCH1), that is [(*1st data word* * MATCH1) = (MATCH0 * MATCH1)] 111b - Match is enabled. Match is enabled, if (any data word AND MATCH1) equals (MATCH0 AND MATCH1), that is [(*any data word* * MATCH1) = (MATCH0 * MATCH1)]
15-12 —	Reserved
11-8 PCSPOL	Peripheral Chip Select Polarity Configures the polarity of each Peripheral Chip Select pin. Each PCSPOL bit position (let's call it *n*) corresponds to a PCS[*n*] pin. For example, PCSPOL[0] is chip select 0 (at PCS[0] pin), and PCSPOL[1] is chip select 1 (at PCS[1] pin). If a PCSPOL bit: • = 0, then the PCS[*n*] pin is active low. • = 1, then the PCS[*n*] pin is active high. **NOTE:** The entire PCSPOL field is not fully supported in every LPSPI module instance. Refer to the chip-specific information for LPSPI.
7-5 —	Reserved
4 —	Reserved
3 NOSTALL	No Stall In Master mode, LPSPI stalls transfers when the transmit FIFO is empty, ensuring that no transmit FIFO underrun can occur. Setting the No Stall bit disables this functionality. 0b - Transfers stall when the transmit FIFO is empty 1b - Transfers do not stall, allowing transmit FIFO underruns to occur
2 AUTOPCS	Automatic PCS For correct operations, the LPSPI slave normally requires the PCS to negate between frames. Setting the AUTOPCS bit causes LPSPI to generate an internal PCS signal at the end of each transfer word when the Clock Phase bit TCR[CPHA] = 1. • When the Automatic PCS bit is set, the SCK must remain idle for at least 4 LPSPI functional clock cycles (divided by the Prescaler Value TCR[PRESCALE] configuration) between each word, to ensure correct operations • In Master mode, the Automatic PCS bit is ignored 0b - Automatic PCS generation is disabled 1b - Automatic PCS generation is enabled
1 SAMPLE	Sample Point When set, the LPSPI master samples the input data on a delayed loopback SCK clock edge, which improves the setup time when sampling data. • The input data setup time in Master mode with delayed SCK edge is equal to the input data setup time in Slave mode • In Slave mode, the SAMPLE bit is ignored

Table continues on the next page...

Field	Function
	0b - Input data is sampled on SCK edge 1b - Input data is sampled on delayed SCK edge
0 MASTER	Master Mode Configures LPSPI in Master or Slave mode. The Master Mode bit directly controls the direction of the SCK and PCS pins. 0b - Slave mode 1b - Master mode

48.5.1.10 Data Match 0 (DMR0)

48.5.1.10.1 Offset

Register	Offset
DMR0	30h

48.5.1.10.2 Function

For the Data Match register to be available, set CFGR0[RDMO] = 1. Do not change the value stored in DMR0 while CFGR0[RDMO] is enabled.

48.5.1.10.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	MATCH0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MATCH0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

48.5.1.10.4 Fields

Field	Function
31-0 MATCH0	Match 0 Value When CFGR0[RDMO] = 1, the value in MATCH0 is compared against the received data.

48.5.1.11 Data Match 1 (DMR1)

48.5.1.11.1 Offset

Register	Offset
DMR1	34h

48.5.1.11.2 Function

The DMR1 register use depends on setting CFGR0[RDMO] = 1. Do not change the value stored in DMR1 while CFGR0[RDMO] is enabled.

48.5.1.11.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	MATCH1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MATCH1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

48.5.1.11.4 Fields

Field	Function
31-0 MATCH1	Match 1 Value When CFGR0[RDMO] = 1, the value in MATCH1 is compared against the received data.

48.5.1.12 Clock Configuration (CCR)

48.5.1.12.1 Offset

Register	Offset
CCR	40h

48.5.1.12.2 Function

The Clock Configuration Register is only used in Master mode, and the Clock Configuration Register cannot be changed when LPSPI is enabled.

48.5.1.12.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SCKPCS								PCSSCK							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DBT								SCKDIV							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

48.5.1.12.4 Fields

Field	Function
31-24 SCKPCS	SCK-to-PCS Delay In Master mode: configures the delay from the last SCK edge to the PCS negation. • The delay is equal to (SCKPCS + 1) cycles of the LPSPI functional clock divided by the Prescaler Value TCR[PRESCALE] configuration. • The minimum delay is 1 cycle. See Figure 48-2.
23-16 PCSSCK	PCS-to-SCK Delay In Master mode: configures the delay from the PCS assertion to the first SCK edge. • The delay is equal to (PCSSCK + 1) cycles of the LPSPI functional clock divided by the Prescaler Value TCR[PRESCALE] configuration. • The minimum delay is 1 cycle. See Figure 48-2.
15-8 DBT	Delay Between Transfers In Master mode: • Configures the delay from the PCS negation to the next PCS assertion. • The delay is equal to (DBT + 2) cycles of the LPSPI functional clock divided by the Prescaler Value TCR[PRESCALE] configuration.

Table continues on the next page...

Field	Function
	• The minimum delay is 2 cycles. • Half of the delay occurs before PCS assertion and the other half of the delay occurs after PCS negation. If the command word is updated between 2 transfers, then the command word is updated half-way between the PCS negation of the last transfer and PCS assertion of the next transfer. • The command word sets which PCS signal is used (of PCS[3:0]}, the polarity/phase of the SCK signal, and the Prescaler Value. • Configures the delay from the last SCK edge of a transfer word and the first SCK edge of the next transfer word, in a continuous transfer. • The delay is equal to (DBT + 1) cycles of the LPSPI functional clock divided by the Prescaler Value TCR[PRESCALE] configuration. • The minimum delay is 1 cycle.
7-0 SCKDIV	SCK Divider In Master mode, the SCK Divider configures the divide ratio of the SCK pin. • The SCK period is equal to (SCKDIV+2) cycles of LPSPI functional clock divided by the Prescaler Value TCR[PRESCALE] configuration. • The minimum SCK period is 2 cycles. • If the SCK period is an odd number of cycles, then the 1st half of the SCK period is 1 cycle longer than the 2nd half of the SCK period.

48.5.1.13 FIFO Control (FCR)

48.5.1.13.1 Offset

Register	Offset
FCR	58h

48.5.1.13.2 Function

Contains the RXWATER and TXWATER control fields.

48.5.1.13.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0								0				RXWATER			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								0				TXWATER			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

48.5.1.13.4 Fields

Field	Function
31-24 —	Reserved
23-20 —	Reserved
19-16 RXWATER	Receive FIFO Watermark The Receive Data Flag is set whenever the number of words in the receive FIFO is greater than RXWATER. Writing a value equal or greater than the FIFO size truncates the written value.
15-8 —	Reserved
7-4 —	Reserved
3-0 TXWATER	Transmit FIFO Watermark The Transmit Data Flag is set whenever the number of words in the transmit FIFO is equal or less than TXWATER. Writing a value equal or greater than the FIFO size truncates the written value.

48.5.1.14 FIFO Status (FSR)

48.5.1.14.1 Offset

Register	Offset
FSR	5Ch

48.5.1.14.2 Function

Contains the TXCOUNT and RXCOUNT fields, which hold the number of words currently stored in the FIFO.

48.5.1.14.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0								0			RXCOUNT				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								0			TXCOUNT				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

48.5.1.14.4 Fields

Field	Function
31-24 —	Reserved
23-21 —	Reserved
20-16 RXCOUNT	Receive FIFO Count Returns the number of words currently stored in the receive FIFO.
15-8 —	Reserved
7-5 —	Reserved
4-0 TXCOUNT	Transmit FIFO Count Returns the number of words currently stored in the transmit FIFO.

48.5.1.15 Transmit Command (TCR)

48.5.1.15.1 Offset

Register	Offset
TCR	60h

48.5.1.15.2 Function

Writes to either the Transmit Command Register or Transmit Data Register pushes the data into the transmit FIFO, in the order that the data are written. The Command Register should only be written using 32-bit writes. Command Register writes are tagged and cause the command register to update, after that entry reaches the top of the FIFO. This allows changes to the command word and the transmit data itself to be interleaved. That is, the writes to the two registers can be interleaved (write command word, then data word, then command word, and so on). Changing the command word causes all subsequent SPI bus transfers to be performed using the new command word.

- **In Master mode,** writing a new command word does not initiate a new transfer, unless TXMSK is set. Transfers are initiated by transmit data in the transmit FIFO, or by a new command word (with TXMSK set). Hardware clears TXMSK when the PCS negates.
- **In Master mode,** if the command word is changed before an existing frame has completed, then the existing frame terminates and the command word then updates. The command word can be changed during a continuous transfer, if CONTC of the new command word is set and the command word is written on a frame size boundary.
- **In Slave mode,** the command word should be changed only when LPSPI is idle and there is no SPI bus transfer.

Avoid register reading problems: Reading the Transmit Command Register returns the current state of the command register. Reading the Transmit Command Register at the same time that the Transmit Command Register is loaded from the transmit FIFO, can return an incorrect Transmit Command Register value. It is recommended:

- Read the Transmit Command Register when the transmit FIFO is empty,
- Read the Transmit Command Register more than once and then compare the returned values.

48.5.1.15.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	CPOL	CPHA	PRESCALE			Reserved	PCS		LSBF	BYSW	CONT	CONTC	RXMSK	TXMSK	WIDTH	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	Reserved				FRAMESZ											
W	0															
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

48.5.1.15.4 Fields

Field	Function
31 CPOL	Clock Polarity The Clock Polarity field is only updated when PCS negated. See Figure 48-2. 0b - The inactive state value of SCK is low 1b - The inactive state value of SCK is high
30 CPHA	Clock Phase The Clock Phase field is only updated when PCS negated. See Figure 48-2. 0b - Captured. Data is captured on the leading edge of SCK and **changed** on the following edge of SCK 1b - Changed. Data is changed on the leading edge of SCK and **captured** on the following edge of SCK
29-27 PRESCALE	Prescaler Value For all SPI bus transfers, the Prescaler value applied to the clock configuration register. The Prescaler Value field is only updated when PCS negated. 000b - Divide by 1 001b - Divide by 2 010b - Divide by 4 011b - Divide by 8 100b - Divide by 16 101b - Divide by 32 110b - Divide by 64 111b - Divide by 128
26 —	Reserved
25-24 PCS	Peripheral Chip Select Configures the peripheral chip select used for the transfer. The Peripheral Chip Select field is only updated when PCS negated. **NOTE:** The entire PCS field is not fully supported in every LPSPI module instance. See the chip-specific LPSPI information. 00b - Transfer using PCS[0] 01b - Transfer using PCS[1] 10b - Transfer using PCS[2] 11b - Transfer using PCS[3]
23 LSBF	LSB First 0b - Data is transferred MSB first 1b - Data is transferred LSB first
22 BYSW	Byte Swap Byte swap swaps the contents of [31:24] with [7:0] and [23:16] with [15:8] for each transmit data word read from the FIFO and for each received data word stored to the FIFO (or compared with match registers). 0b - Byte swap is disabled 1b - Byte swap is enabled
21	Continuous Transfer

Table continues on the next page...

Field	Function
CONT	• In Master mode, CONT keeps the PCS asserted at the end of the frame size, until a command word is received that starts a new frame. • In Slave mode, when CONT is enabled, LPSPI only transmits the first FRAMESZ bits; after which LPSPI transmits received data (assuming a 32-bit shift register) until the next PCS negation. 0b - Continuous transfer is disabled 1b - Continuous transfer is enabled
20 CONTC	Continuing Command In Master mode, the CONTC bit allows the command word to be changed within a continuous transfer. • The initial command word must enable continuous transfer (CONT = 1), • the continuing command must set this bit (CONTC = 1), • and the continuing command word must be loaded on a frame size boundary. For example, if the continuous transfer has a frame size of 64-bits, then a continuing command word must be loaded on a 64-bit boundary. 0b - Command word for start of new transfer 1b - Command word for continuing transfer
19 RXMSK	Receive Data Mask When set, receive data is masked (receive data is not stored in receive FIFO). 0b - Normal transfer 1b - Receive data is masked
18 TXMSK	Transmit Data Mask When set, transmit data is masked (no data is loaded from transmit FIFO and output pin is tristated). In Master mode, the TXMSK bit initiates a new transfer which cannot be aborted by another command word; the TXMSK bit is cleared by hardware at the end of the transfer. 0b - Normal transfer 1b - Mask transmit data
17-16 WIDTH	Transfer Width Configures between serial (1-bit) or parallel transfers. For half-duplex parallel transfers, either Receive Data Mask (RXMSK) or Transmit Data Mask (TXMSK) must be set. 00b - 1 bit transfer 01b - 2 bit transfer 10b - 4 bit transfer 11b - Reserved
15-12 —	Reserved
11-0 FRAMESZ	Frame Size Configures the frame size in number of bits equal to (FRAMESZ + 1). • The minimum frame size is 8 bits • The minimum word size is 2 bits; a frame size of 33 bits is not supported. • If the frame size is larger than 32 bits, then the frame is divided into multiple words of 32-bits; each word is loaded from the transmit FIFO and stored in the receive FIFO separately. • If the size of the frame is not divisible by 32, then the last load of the transmit FIFO and store of the receive FIFO contains the remainder bits. For example, a 72-bit transfer consists of 3 words: the 1st and 2nd words are 32 bits, and the 3rd word is 8 bits.

48.5.1.16 Transmit Data (TDR)

48.5.1.16.1 Offset

Register	Offset
TDR	64h

48.5.1.16.2 Function

Writes to either the Transmit Command Register or Transmit Data Register push the data into the transmit FIFO, in the order that the data was written. The Data Register can be written using 32-bit, 16-bit, or 8-bit writes, but each write pushes data into the FIFO with zero pushed in unwritten bytes.

48.5.1.16.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R																
W	DATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R																
W	DATA															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

48.5.1.16.4 Fields

Field	Function
31-0 DATA	Transmit Data Both 8-bit and 16-bit writes of transmit data zero-extend the data written and push the data into the transmit FIFO. To zero-extend 8-bit and 16-bit writes (to 32 bits), means that the higher order (most significant) empty parts of the 8-bit and 16-bit writes are filled with zeroes.

48.5.1.17 Receive Status (RSR)

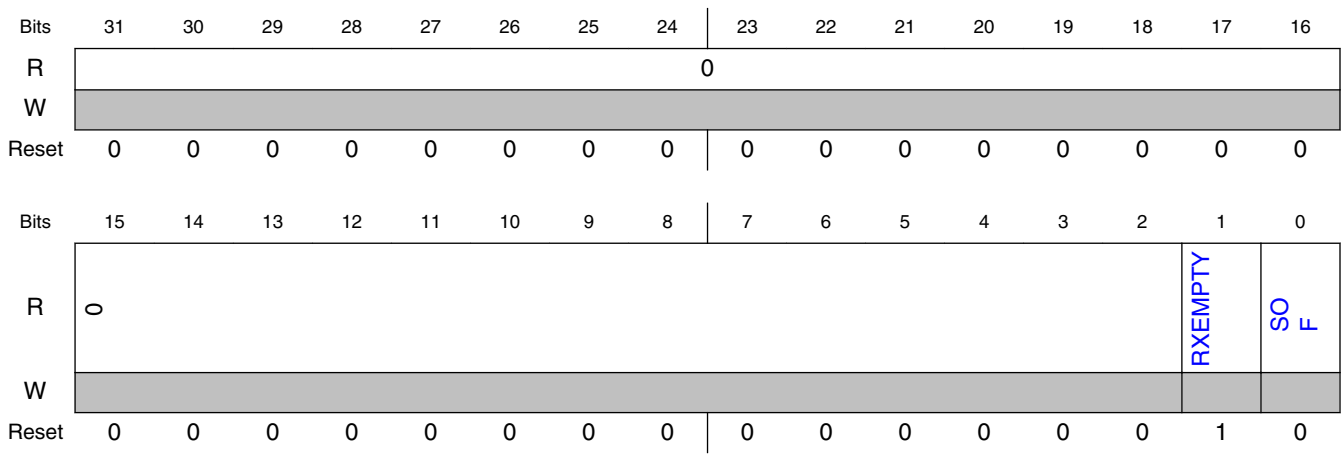
48.5.1.17.1 Offset

Register	Offset
RSR	70h

48.5.1.17.2 Function

Contains data flow status fields for receive FIFO.

48.5.1.17.3 Diagram



48.5.1.17.4 Fields

Field	Function
31-2 —	Reserved
1 RXEMPTY	RX FIFO Empty 0b - RX FIFO is not empty 1b - RX FIFO is empty
0 SOF	Start Of Frame Indicates that this is the first data word received after PCS assertion. 0b - Subsequent data word received after PCS assertion 1b - First data word received after PCS assertion

48.5.1.18 Receive Data (RDR)

48.5.1.18.1 Offset

Register	Offset
RDR	74h

48.5.1.18.2 Function

Pulls the first entry from the receive FIFO.

48.5.1.18.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DATA															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

48.5.1.18.4 Fields

Field	Function
31-0 DATA	Receive Data

Chapter 49
Low Power Universal Asynchronous Receiver/ Transmitter (LPUART)

49.1 Chip-specific LPUART information

Table 49-1. Reference links to related information

Topic	Related module(s)	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
Interrupts, DMA Events and XBAR Assignments	-	Interrupts, DMA Events and XBAR Assignments

49.2 Overview

The Low Power Universal Asynchronous Receiver/Transmitter (LPUART) module provides asynchronous, serial communication capability with external devices. LPUART supports non-return-to-zero (NRZ) encoding format and IrDA compatible infrared (low-speed) SIR format. The LPUART can continue operating while the processor is in low-power mode if an appropriate peripheral clock is available.

49.2.1 Block diagram

The following figure shows the transmitter portion of the LPUART.

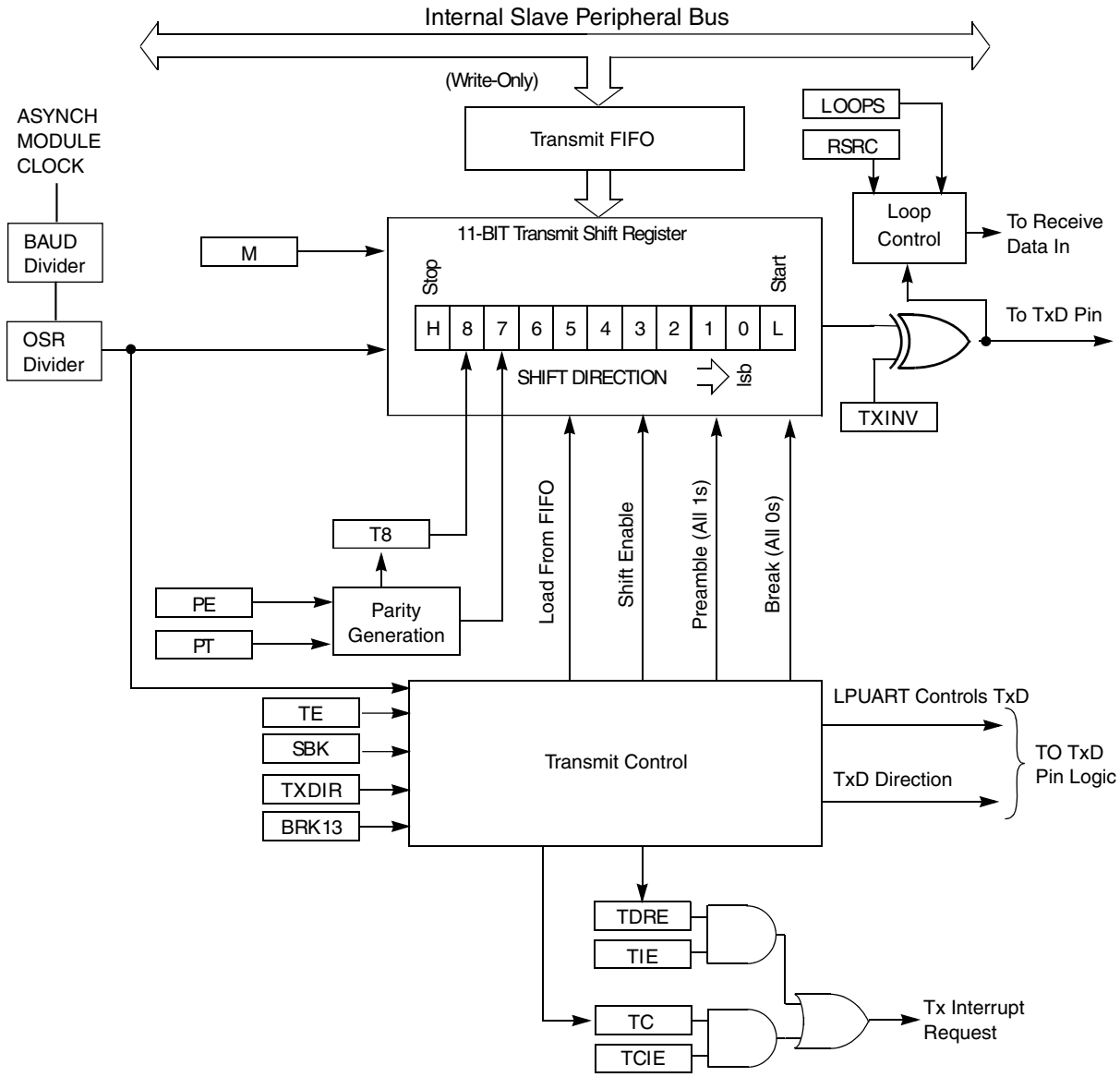


Figure 49-1. LPUART transmitter block diagram

The following figure shows the receiver portion of the LPUART.

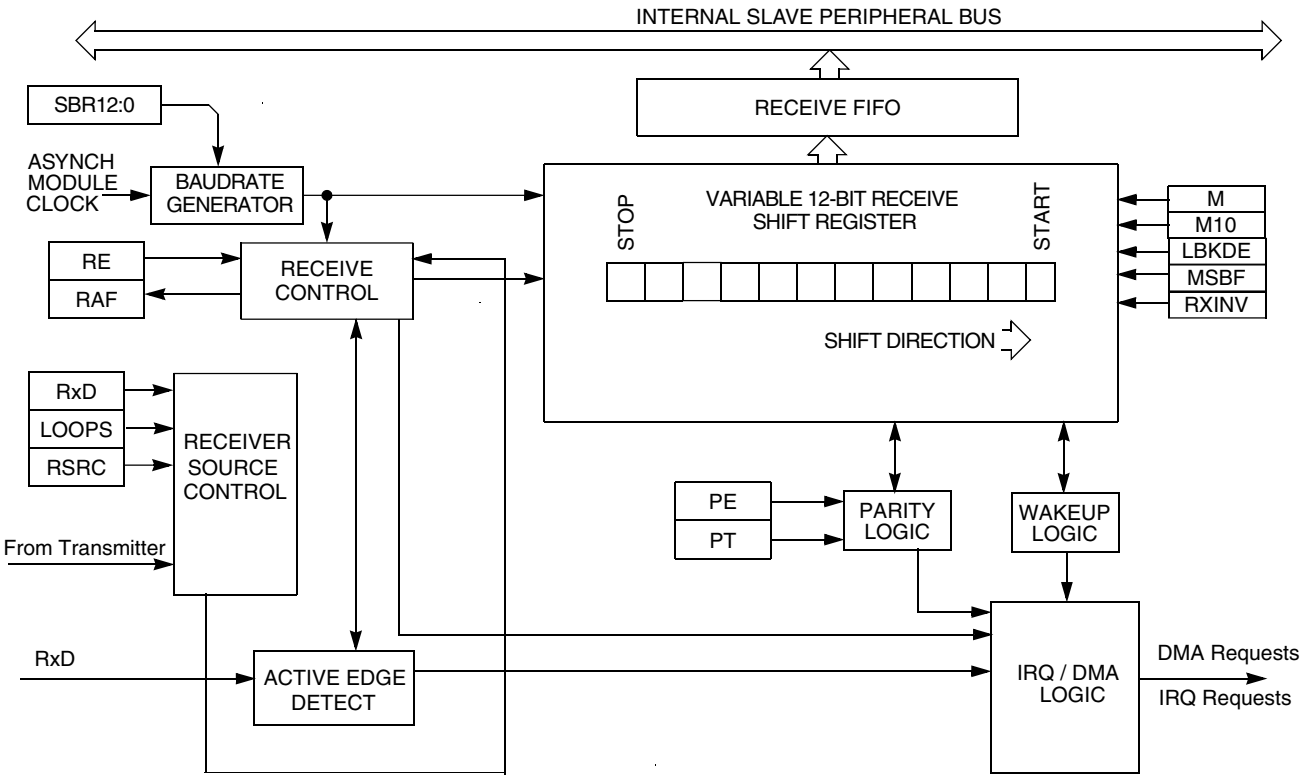


Figure 49-2. LPUART receiver block diagram

49.2.2 Features

Following are the features of the LPUART module.

- Full-duplex, standard non-return-to-zero (NRZ) format
- Programmable baud rates (13-bit modulo divider) with configurable oversampling ratio from 4x to 32x
- Asynchronous operation of transmit and receive baud rate with respect to the bus clock:
  - Baud rate can be configured independently of the bus clock frequency
  - Supports operation in low-power modes
- Interrupt, DMA or polled operation:
  - Transmit data register empty and transmission complete
  - Receive data register full
  - Receive overrun, parity error, framing error, and noise error
  - Idle receiver detect
  - Active edge on receive pin
  - Break detect supporting LIN
  - Receive data match
- Hardware parity generation and checking
- Programmable 7-bit, 8-bit, 9-bit or 10-bit character length

- Programmable 1-bit or 2-bit stop bits
- Support for three receiver wakeup methods:
  - Idle line wakeup
  - Address mark wakeup
  - Receive data match
- Automatic address matching to reduce ISR overhead:
  - Address mark matching
  - Idle line address matching
  - Address match start, address match end
- Optional 13-bit/11-bit break character generation
- Configurable idle length detection supporting 1, 2, 4, 8, 16, 32, 64, or 128 idle characters
- Selectable transmitter output and receiver input polarity
- Hardware flow control support for request to send (RTS) and clear to send (CTS) signals
- Selectable IrDA 1.4 return-to-zero-inverted (RZI) format with programmable pulse width
- Independent FIFO structure for transmit and receive
  - Separate configurable watermark for receive and transmit requests
  - Option for receiver to assert request after a configurable number of idle characters if receive FIFO is not empty

49.3 Functional description

The LPUART supports full-duplex, asynchronous, NRZ serial communication and comprises a baud rate generator, transmitter, and receiver block. The transmitter and receiver operate independently, although they use the same baud rate generator. The following describes each of the blocks of the LPUART.

49.3.1 Low-power modes

The LPUART remains functional during low power modes, provided the CTRL[DOZEEN] bit is clear and the LPUART functional clock remain enabled. The LPUART can generate an interrupt or DMA request to cause a wakeup from low power modes.

The LPUART can be configured to be disabled in low power modes, when the CTRL[DOZEEN] bit is set. In this case the transmitter and receiver finish transmitting/receiving the current word.

If the LPUART is disabled in low power modes, then it can generate a wakeup via the STAT[RXEDGIF] flag if the receiver detects an active edge.

NOTE

See the chip-specific information to know the specific low-power modes available on your device.

49.3.2 Debug mode

The LPUART remains functional in debug mode.

49.3.3 Baud rate generation

A 13-bit modulus counter in the baud rate generator derives the baud rate for both the receiver and the transmitter. The value from 1 to 8191 written to BAUD[SBR] determines the baud clock divisor for the asynchronous LPUART baud clock. The baud rate clock drives the receiver, while the transmitter is driven by a bit clock which is generated from baud rate clock divided by the over sampling ratio (OSR). Depending on the over sampling ratio, the receiver has an acquisition rate of 4 to 32 samples per bit time.

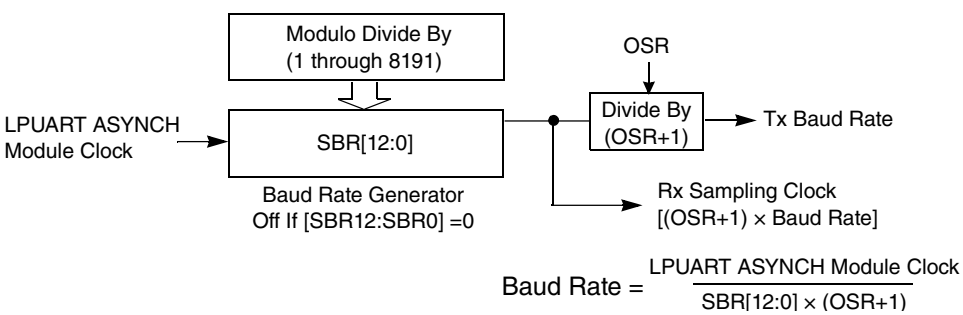


Figure 49-3. LPUART baud rate generation

Baud rate generation is subject to two sources of error:

- Integer division of the asynchronous LPUART baud clock may not give the exact target frequency.
- Synchronization with the asynchronous LPUART baud clock can cause phase shift.

The baud rate generation is a free-running counter that continues whenever the transmitter or receiver is enabled. The transmitter bit clock continues whenever the transmitter is enabled; each transmitted character aligns to the next edge of the transmit bit clock.

In general, configuring OSR for a higher oversampling ratio and/or sampling on both edges of the clock slightly improves the LPUART tolerance to baud rate mismatch between the received data and the LPUART configured baud rate. However, the three data samples in each bit will also be closer together which may impact noise sensitivity.

49.3.4 Transmitter functional description

This section describes the overall block diagram for the LPUART transmitter, as well as specialized functions for sending break and idle characters.

The transmitter output (TXD) idle state defaults to logic high; the transmitter output is inverted by setting CTRL[TXINV]. CTRL[TXINV] is cleared following reset. The transmitter is enabled by setting the CTRL[TE] bit. This queues a preamble character that is one full character frame of the idle state. The transmitter then remains idle until data is available in the transmit FIFO. Programs store data into the transmit FIFO by writing to the DATA register.

The central element of the LPUART transmitter is the transmit shift register that is 9-bit to 13-bit long depending on the setting in the CTRL[M], CTRL[M7], BAUD[M10] and BAUD[SBNS] control bits. For the remainder of this section, assume CTRL[M], CTRL[M7], BAUD[M10] and BAUD[SBNS] are cleared, selecting the normal 8-bit data mode. In 8-bit data mode, the shift register holds a start bit, eight data bits, and a stop bit. When the transmit shift register is available for a new character, the value waiting in the transmit data register is transferred to the transmit shift register, synchronized with the baud rate clock, and the transmit data register empty (STAT[TDRE]) status flag is set to indicate another character may be written to the transmit FIFO at the DATA register.

If no new character is waiting in the transmit FIFO after a stop bit is shifted out of the TXD pin, the transmitter sets the transmit complete flag and enters an idle mode, with TXD high, waiting for more characters to transmit.

Writing 0 to CTRL[TE] does not immediately disable the transmitter. The current transmit activity in progress must first be completed (that could include a data character, idle character or break character), although the transmitter does not start transmitting another character.

49.3.4.1 Send break and queued idle

The CTRL[SBK] bit sends break characters originally used to gain the attention of old teletype receivers. Break characters are a full character time of logic 0, 9-bit to 12-bit times including the start and stop bits. A longer break of 13-bit times can be enabled by setting STAT[BRK13]. Normally, a program waits for STAT[TDRE] to become set to indicate the last character of a message has moved to the transmit shifter, write 1, and then write 0 to the CTRL[SBK] bit. This action queues a break character to be sent as soon as the shifter is available. If CTRL[SBK] remains 1 when the queued break moves into the shifter, synchronized to the baud rate clock, an additional break character is queued. When the LPUART is the receiving device, a break character is received as 0s in all data bits and a framing error (STAT[FE] = 1) is detected.

A break character can also be transmitted by writing to the DATA register with DATA[FRETSC] set and the data bits clear. This supports transmitting the break character as part of the normal data stream and also allows the DMA to transmit a break character.

When idle-line wakeup is used, a full character time of idle (logic 1) is needed between messages to wake up any sleeping receivers. Normally, a program waits for STAT[TDRE] to become set to indicate the last character of a message has moved to the transmit shifter, write 0, and then write 1 to the CTRL[TE] bit. This action queues an idle character to be sent as soon as the shifter is available. As long as the character in the shifter does not finish while CTRL[TE] is cleared, the LPUART transmitter never actually releases control of the TXD pin.

An idle character can also be transmitted by writing to the DATA register with DATA[FRETSC] and DATA[T9] set, and all other bits cleared. This supports transmitting the idle character as part of the normal data stream and also allows the DMA to transmit a idle character.

The length of the break character is affected by the STAT[BRK13], CTRL[M], CTRL[M7], BAUD[M10] and BAUD[SNBS] bits as shown below.

Table 49-2. Break character length

BRK13	M	M10	M7	SBNS	Break character length
0	0	0	0	0	10 bit times
0	0	0	0	1	11 bit times
0	0	0	1	0	9 bit times
0	0	0	1	1	10 bit times
0	1	0	X	0	11 bit times
0	1	0	X	1	12 bit times

Table continues on the next page...

Table 49-2. Break character length (continued)

BRK13	M	M10	M7	SBNS	Break character length
0	X	1	X	0	12 bit times
0	X	1	X	1	13 bit times
1	0	0	0	0	13 bit times
1	0	0	0	1	13 bit times
1	0	0	1	0	12 bit times
1	0	0	1	1	12 bit times
1	1	0	X	0	14 bit times
1	1	0	X	1	14 bit times
1	X	1	X	0	15 bit times
1	X	1	X	1	15 bit times

49.3.4.2 Hardware flow control

The transmitter supports hardware flow control by gating the transmission with the value of CTS_B. If the clear-to-send operation is enabled, the character is transmitted when CTS_B is asserted. If CTS_B is deasserted in the middle of a transmission with characters remaining in the transmitter FIFO, the character in the transmit shift register will be complete; any characters in the FIFO will wait for the CTS_B to assert again and TXD remains in the mark state (idle state) until CTS_B is reasserted. The CTS_B pin must assert for longer than one bit period to guarantee a new transmission is started when the transmitter is idle with data to send.

If the clear-to-send operation is disabled, the transmitter ignores the state of CTS_B.

The transmitter's CTS_B signal can be enabled even if the same LPUART receiver's RTS_B signal is disabled.

49.3.4.3 Transceiver driver enable

The transmitter can use RTS_B as an enable signal for the driver of an external transceiver. See Transceiver driver enable using RTS_B for details. If the request-to-send operation is enabled, when a character is placed into an empty transmit shift register, RTS_B asserts one bit time before the start bit is transmitted. RTS_B remains asserted for the whole time that the transmit shift register has any characters. RTS_B deasserts one bit time after all characters in the transmit FIFO and shift register are completely sent,

including the last stop bit. In other words, when RTS_B is used as a transceiver enable, the RTS_B asserts 1 bit time before the transmitter starts transmitting and negates 1 bit time after the transmitter goes idle.

Transmitting a break character also asserts RTS_B, with the same assertion and deassertion timing as having a character in the transmit shift register.

The transmitter's RTS_B signal asserts only when the transmitter is enabled. However, the transmitter's RTS_B signal is unaffected by its CTS_B signal. RTS_B remains asserted until the transfer is completed, even if the transmitter is disabled mid-way through a data transfer.

49.3.4.4 Transceiver driver enable using RTS_B

RS-485 is a multiple drop communication protocol in which the LPUART transceiver's driver is 3-stated unless the LPUART is driving. The RTS_B signal can be used by the transmitter to enable the driver of a transceiver. The polarity of RTS_B can be matched to the polarity of the transceiver's driver enable signal.

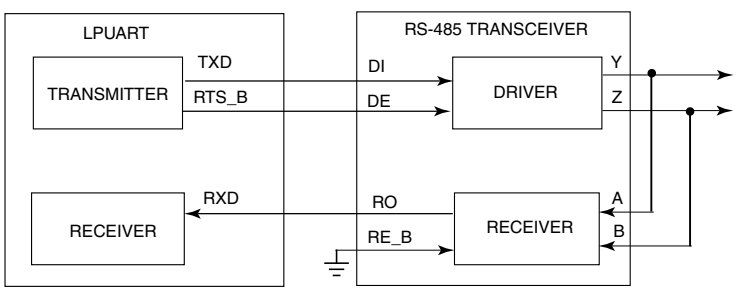


Figure 49-4. Transceiver driver enable using RTS_B

In the figure, the receiver enable signal is asserted. Another option for this connection is to connect RTS_B to both DE and RE_B. The transceiver's receiver is disabled while driving. A pullup can pull RXD to a non-floating value during this time. This option can be refined further by operating the LPUART in single wire mode, freeing the RXD pin for other uses.

49.3.5 Receiver functional description

In this section, the receiver block diagram is a guide for the overall receiver functional description. Next, the data sampling technique used to reconstruct receiver data is described in more detail. Finally, different variations of the receiver wakeup function are explained.

The receiver input is inverted by setting STAT[RXINV]. The receiver is enabled by setting the CTRL[RE] bit. Character frames consist of a start bit of logic 0, seven to ten data bits (msb or lsb first), and one or two stop bits of logic 1. For information about 7-bit, 9-bit or 10-bit data mode, refer to Data Modes. For the remainder of this discussion, assume the LPUART is configured for normal 8-bit data mode.

After receiving the stop bit into the receive shifter, and provided the receive data register is not already full, the data character is transferred to the receive data register and the receive data register full (STAT[RDRF]) status flag is set. If [RDRF] was already set indicating the receive data register (buffer) was already full, the overrun (OR) status flag is set and the new data is lost.

Since the LPUART receiver is separate from the receive FIFO, the receive shift register can be receiving the next word when the receive FIFO is full and it is only at the end of the character that the next data is written into the receive FIFO, potentially triggering the overrun flag if FIFO is full.

When a program detects that the receive data register is full (STAT[RDRF] = 1), it gets the data from the receive data register by reading the DATA register. Refer to Interrupts and status flags for details about flag clearing.

49.3.5.1 Data sampling technique

The LPUART receiver supports a configurable oversampling rate of between 4× and 32× of the baud rate clock for sampling. The receiver starts by taking logic level samples at the oversampling rate times the baud rate to search for a falling edge on the RXD serial data input pin. A falling edge is defined as a logic 0 sample after three consecutive logic 1 samples. The oversampling baud rate clock divides the bit time into 4 to 32 segments from 1 to OSR (where OSR is the configured oversampling ratio). When a falling edge is located, three more samples are taken at (OSR/2), (OSR/2)+1, and (OSR/2)+2 to make sure this was a real start bit and not merely noise. If at least two of these three samples are 0, the receiver assumes it is synchronized to a received character. If another falling edge is detected before the receiver is considered synchronized, the receiver restarts the sampling from the first segment.

The receiver then samples each bit time, including the start and stop bits, at (OSR/2), (OSR/2)+1, and (OSR/2)+2 to determine the logic level for that bit. The logic level is interpreted to be that of the majority of the samples taken during the bit time. If any sample in any bit time, including the start and stop bits, in a character frame fails to agree with the logic level for that bit, the noise flag (STAT[NF]) is set when the received character is transferred to the receive FIFO.

When the LPUART receiver is configured to sample on both edges of the baud rate clock (that is, when BAUD[BOTHEDGE]=1), the number of segments in each received bit is effectively doubled (from 1 to OSR×2). The start and data bits are then sampled at OSR, OSR+1 and OSR+2. Sampling on both edges of the clock must be enabled for oversampling rates of 4× to 7× and is optional for higher oversampling rates.

The "synchronization" is a feature of the LPUART module to synchronize the internal oversampling counter with detected falling edge on Rx signal, and to adjust the data sampling window. The falling edge detection needs three consecutive '1' prior to '1'->'0' transition. After the initial falling edge detection for the start bit, the circuit continuously monitors the next falling edge, and resets the counter once another falling edge is detected. This is called "resynchronization".

- When LPUART_BAUD[RESYNCDIS] = 0, this falling edge detection and resynchronization is performed not only for the start bit but also for the rest of character reception after the start bit.
- When LPUART_BAUD[RESYNCDIS] = 1, the falling edge detection and resynchronization is performed only for the start bit. The use case for disabling the resynchronization is protocols that require this (for example, LIN 2.1 prohibits resynchronization within a byte).

  See the following table and figure.

Table 49-3. LPUART resynchronization

Resynchronization	LPUART_BAUD[RESYNCDIS]=0	LPUART_BAUD[RESYNCDIS]=1
For the starting bit falling edge	Yes	Yes
For every falling edges after the start bit	Yes	No

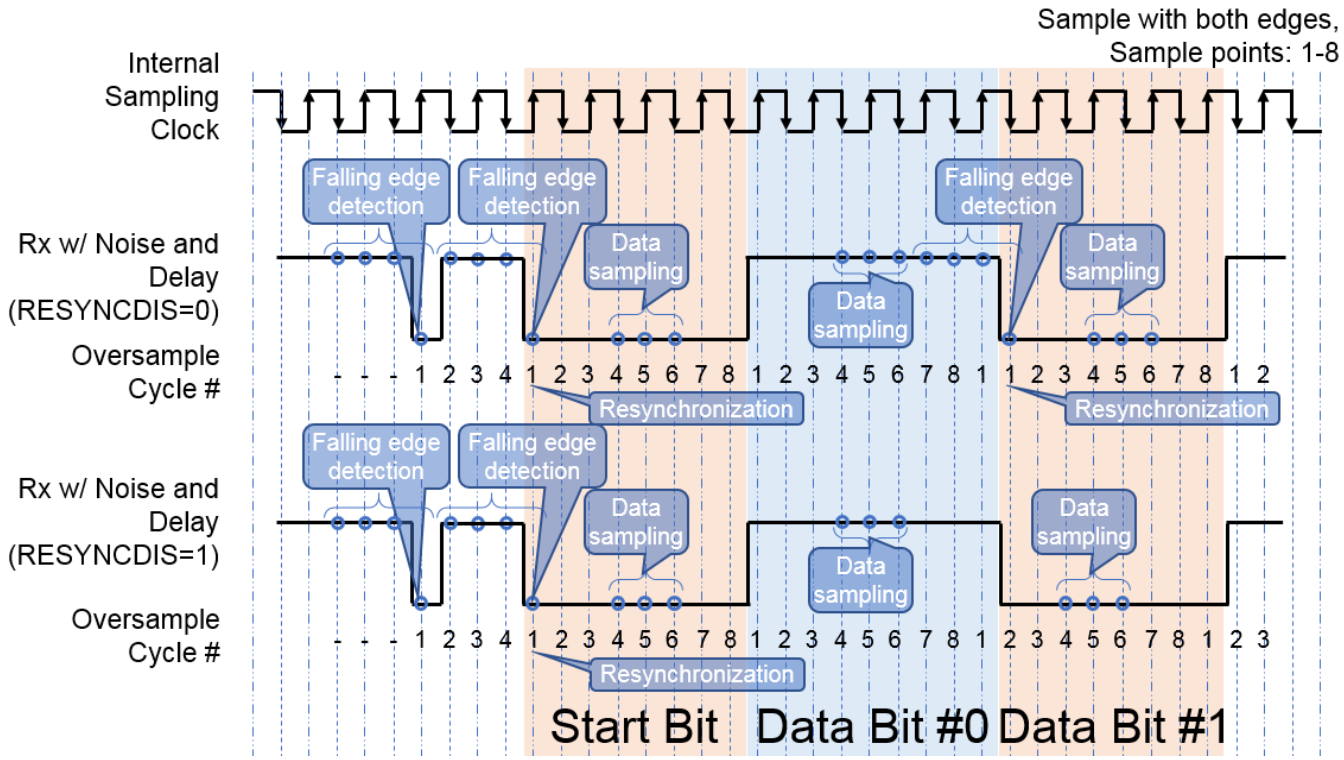


Figure 49-5. LPUART resynchronization

49.3.5.2 Receiver wakeup operation

Receiver wakeup and receiver address matching is a hardware mechanism that allows an LPUART receiver to ignore the characters in a message intended for a different receiver.

During receiver wakeup, all receivers evaluate the first character(s) of each message, and as soon as they determine the message is intended for a different receiver, they write logic 1 to the receiver wake up control bit (CTRL[RWU]). When CTRL[RWU] and STAT[RWUID] bit are set, the status flags associated with the receiver, with the exception of the idle bit, STAT[IDLE], are inhibited from setting, thus eliminating the software overhead for handling the unimportant message characters. At the end of a message, all receivers automatically force CTRL[RWU] to 0 so all receivers wake up in time to look at the first character(s) of the next message.

During receiver address matching, the address matching is performed in hardware and the LPUART receiver ignores all characters that do not meet the address match requirements.

Table 49-4. Receiver Wakeup Options

RWU	MA1 | MA2	MATCFG	WAKE:RWUID	Receiver Wakeup
0	0	X	X	Normal operation

Table continues on the next page...

Table 49-4. Receiver Wakeup Options (continued)

RWU	MA1 | MA2	MATCFG	WAKE:RWUID	Receiver Wakeup
1	0	00	00	Receiver wakeup on idle line, IDLE flag not set
1	0	00	01	Receiver wakeup on idle line, IDLE flag set
1	0	00	10	Receiver wakeup on address mark
1	1	11	10	Receiver wakeup on data match
0	1	00	X0	Address mark address match, IDLE flag not set for discarded characters
0	1	00	X1	Address mark address match, IDLE flag set for discarded characters
0	1	01	X0	Idle line address match
0	1	10	X0	Match On and Match Off, IDLE flag not set for discarded characters
0	1	10	X1	Match On and Match Off, IDLE flag set for discarded characters

49.3.5.2.1 Idle-line wakeup

When CTRL[WAKE] is cleared, the receiver is configured for idle-line wakeup. In this mode, CTRL[RWU] is cleared automatically when the receiver detects a full character time of the idle-line level. The CTRL[M], CTRL[M7], and BAUD[M10] control bits select 7-bit to 10-bit data mode and the BAUD[SBNS] bit selects 1-bit or 2-bit stop bit number that determines how many bit times of idle are needed to constitute a full character time, 9 to 13 bit times because of the start and stop bits.

When CTRL[RWU] is 1 and STAT[RWUID] is 0, the idle condition that wakes up the receiver does not set the STAT[IDLE] flag. The receiver wakes up and waits for the first data character of the next message that sets the STAT[RDRF] flag and generates an interrupt if enabled. When STAT[RWUID] is 1, any idle condition sets the STAT[IDLE] flag and generates an interrupt if enabled, regardless of whether CTRL[RWU] is 0 or 1.

The idle-line type (CTRL[ILT]) control bit selects one of two ways to detect an idle line.

- When CTRL[ILT] is cleared, the idle bit counter starts after the start bit so the stop bit and any logic 1s at the end of a character count toward the full character time of idle.
- When CTRL[ILT] is set, the idle bit counter does not start until after the stop bit time, so the idle detection is not affected by the data in the last character of the previous message.

49.3.5.2.2 Address-mark wakeup

When CTRL[WAKE] is set, the receiver is configured for address-mark wakeup. In this mode, CTRL[RWU] is cleared automatically when the receiver detects a logic 1 in the most significant bit of the received character. When parity is enabled, the second most significant bit is used for address-mark wakeup.

Address-mark wakeup allows messages to contain idle characters, but requires one bit be reserved for use in address frames. The logic 1 in the most significant bit (or second most significant bit when parity is enabled) of an address frame clears the CTRL[RWU] bit and sets the STAT[RDRF] flag. In this case, the character with the address-mark bit is received even though the receiver was sleeping during most of this character time.

49.3.5.2.3 Data match wakeup

When CTRL[RWU] is set, CTRL[WAKE] is set and BAUD[MATCFG] equals 11, the receiver is configured for data match wakeup. In this mode, CTRL[RWU] is cleared automatically when the receiver detects a character that matches MATCH[MA1] field when BAUD[MAEN1] is set, or that matches MATCH[MA2] when BAUD[MAEN2] is set.

49.3.5.2.4 Address Match operation

Address match operation is enabled when the BAUD[MAEN1] or BAUD[MAEN2] bit is set and BAUD[MATCFG] is equal to 00. In this function, a character received by the RXD pin with a logic 1 in the most significant bit (or second most significant bit when parity is enabled) is considered an address and is compared with the associated MATCH[MA1] or MATCH[MA2] field. The character is only transferred to the receive buffer, and STAT[RDRF] is set, if the comparison matches. All subsequent characters received with a logic 0 in the most significant bit (or second most significant bit when parity is enabled) are considered to be data associated with the address and are transferred to the receive FIFO. If no marked address match occurs then no transfer is made to the receive FIFO, and all following characters with logic zero in the most significant bit (or

second most significant bit when parity is enabled) are also discarded. If both the BAUD[MAEN1] and BAUD[MAEN2] bits are negated, the receiver operates normally and all data received is transferred to the receive FIFO.

Address match operation functions in the same way for both MATCH[MA1] and MATCH[MA2] fields.

- If only one of BAUD[MAEN1] and BAUD[MAEN2] is asserted, a marked address is compared only with the associated match register field and data is transferred to the receive FIFO only on a match.
- If BAUD[MAEN1] and BAUD[MAEN2] are asserted, a marked address is compared with both match registers and data is transferred only on a match with either of the MATCH[MA1] or MATCH[MA2] fields.

49.3.5.2.5 Idle Match operation

Idle match operation is enabled when the BAUD[MAEN1] or BAUD[MAEN2] bit is set and BAUD[MATCFG] is equal to 01. In this function, the first character received by the RXD pin after an idle line condition is considered an address and is compared with the associated MATCH[MA1] or MATCH[MA2] field. The character is transferred only to the receive buffer, and STAT[RDRF] is set, if the comparison matches. All subsequent characters are considered to be data associated with the address and are transferred to the receive FIFO until the next idle line condition is detected. If no address match occurs then no transfer is made to the receive FIFO, and all following frames until the next idle condition are also discarded. If both the BAUD[MAEN1] and BAUD[MAEN2] bits are negated, the receiver operates normally and all data received is transferred to the receive FIFO.

Idle match operation functions in the same way for both MATCH[MA1] or MATCH[MA2] fields.

- If only one of BAUD[MAEN1] and BAUD[MAEN2] is asserted, the first character after an idle line is compared only with the associated match register and data is transferred to the receive FIFO only on a match.
- If BAUD[MAEN1] and BAUD[MAEN2] are asserted, the first character after an idle line is compared with both MATCH[MA1] or MATCH[MA2] fields and data is transferred only on a match with either of the fields.

49.3.5.2.6 Match On Match Off operation

Match on, match off operation is enabled when both BAUD[MAEN1] and BAUD[MAEN2] are set and BAUD[MATCFG] is equal to 10. In this function, a character received by the RXD pin that matches MATCH[MA1] is received and transferred to the receive buffer, and STAT[RDRF] is set. All subsequent characters are considered to be data and are also transferred to the receive FIFO, until a character is received that matches MATCH[MA2] field. The character that matches MATCH[MA2] and all following characters are discarded; this continues until another character that matches MATCH[MA1] is received. If both the BAUD[MAEN1] and BAUD[MAEN2] bits are negated, the receiver operates normally and all data received is transferred to the receive FIFO.

NOTE

Match on, match off operation requires both BAUD[MAEN1] and BAUD[MAEN2] to be asserted.

49.3.5.3 Hardware flow control

To support hardware flow control, the receiver can be programmed to automatically deassert and assert RTS_B.

- RTS_B remains asserted until the transfer is complete, even if the transmitter is disabled midway through a data transfer. See Transceiver driver enable using RTS_B for more details.
- If the receiver request-to-send functionality is enabled, the receiver automatically deasserts RTS_B if the number of characters in the receiver data register is full or a start bit is detected that causes the receiver data register to be full.
- The receiver asserts RTS_B when the number of characters in the receiver data register is not full and has not detected a start bit that causes the receiver data register to be full. It is not affected if STAT[RDRF] is asserted.
- Even if RTS_B is deasserted, the receiver continues to receive characters until the receive FIFO is overrun.
- If the receiver request-to-send functionality is disabled, the receiver RTS_B remains deasserted.

49.3.6 Additional LPUART functions

The following sections describe additional LPUART functions.

49.3.6.1 Data Modes

The LPUART transmitter and receiver can be configured to operate in 7-bit data mode by setting CTRL[M7], 9-bit data mode by setting the CTRL[M] or 10-bit data mode by setting BAUD[M10]. In 9-bit mode, there is a ninth data bit and in 10-bit mode, there is a tenth data bit.

When performing 8-bit writes to the transmit FIFO, the 9th and 10th bit are pushed into the FIFO from CTRL[T8] and CTRL[T9]. For coherent 8-bit writes, write to CTRL[T8] and CTRL[T9] before writing to DATA[7:0], but if values in CTRL[T8] or CTRL[T9] do not need to change then it is not necessary to update CTRL[T8] and CTRL[T9] before every 8-bit write to the DATA register.

When performing 16-bit or 32-bit writes to the transmit FIFO, all 10-bits are pushed into the transmit FIFO from the write data.

When performing 8-bit reads of the receive FIFO, the 9th and 10th bit are held in CTRL[R8] and CTRL[R9] but should be read before reading the DATA register. A 16-bit or 32-bit read of the receive FIFO will return all 10-bits in the DATA register.

The 9-bit data mode is typically used with parity to allow eight bits of data plus the parity in the ninth bit, or it is used with address-mark wakeup so the ninth data bit can serve as the wakeup bit. The 10-bit data mode is typically used with parity and address-mark wakeup so the ninth data bit can serve as the wakeup bit and the tenth bit as the parity bit. In custom protocols, the ninth and/or tenth bits can also serve as software-controlled markers.

49.3.6.2 Idle length

An idle character is a character where the start bit, all data bits and stop bits are in the mark position (idle state, generally logic 1). The CTRL[ILT] bit can be configured to start detecting an idle character from the previous start bit (any data bits and stop bits count towards the idle character detection) or from the previous stop bit.

The number of idle characters that must be received before an idle line condition is detected can also be configured using the CTRL[IDLECFG] field. This field configures the number of idle characters that must be received before the STAT[IDLE] flag is set, the STAT[RAF] flag is cleared and the DATA[IDLINE] flag is set with the next received character.

Idle-line wakeup and idle match operation are also affected by the CTRL[IDLECFG] field. When address match or match on/off operation is enabled, setting the STAT[RWUID] bit causes any discarded characters to be treated as if they were idle characters.

49.3.6.3 Loop mode

When CTRL[LOOPS] is set, the CTRL[RSRC] bit chooses between loop mode (CTRL[RSRC] = 0) or single-wire mode (CTRL[RSRC] = 1). Loop mode is sometimes used to check software, independent of connections in the external system, to help isolate system problems. In this mode, the transmitter output is internally connected to the receiver input and the RXD pin is not used by the LPUART.

49.3.6.4 Single-wire operation

When CTRL[LOOPS] is set, the CTRL[RSRC] bit chooses between loop mode (CTRL[RSRC] = 0) or single-wire mode (CTRL[RSRC] = 1). Single-wire mode implements a half-duplex serial connection. The receiver is internally connected to the transmitter output and to the TXD pin (the RXD pin is not used).

In single-wire mode, the CTRL[TXDIR] bit controls the direction of serial data on the TXD pin. When CTRL[TXDIR] is cleared, the TXD pin is an input to the receiver and the transmitter is temporarily disconnected from the TXD pin so an external device can send serial data to the receiver. When CTRL[TXDIR] is set, the TXD pin is an output driven by the transmitter, the internal loop back connection is disabled, and as a result the receiver cannot receive characters that are sent out by the transmitter.

49.3.7 Infrared interface

The LPUART provides the capability of transmitting narrow pulses to an IR LED and receiving narrow pulses and transforming them to serial bits, which are sent to the LPUART. The IrDA physical layer specification defines a half-duplex infrared communication link for exchanging data. The full standard includes data rates up to 16 Mbits/s. The LPUART IrDA support is limited to SIR mode which supports data rates only between 2.4 kbits/s and 115.2 kbits/s.

The LPUART has an infrared transmit encoder and receive decoder. The infrared decoder converts the received character from the IrDA format to the NRZ format used by the receiver. It also has a OSR oversampling baud rate clock counter that filters noise and indicates when a 1 is received. The LPUART transmits serial bits of data that are encoded by the infrared submodule to transmit a narrow pulse for every zero bit. No pulse is transmitted for every one bit. When receiving data, the IR pulses are detected using an IR photo diode (external to LPUART) and transformed to CMOS levels by the IR receive decoder. The narrow pulses are then stretched by the infrared receive decoder to get back

to a serial bit stream to be received by the LPUART. The polarity of transmitted pulses and expected receive pulses can be inverted so that a direct connection can be made to external IrDA transceiver modules that use active high pulses.

The infrared submodule receives its clock sources from the LPUART. One of these two clocks are selected in the infrared submodule to generate either 1/OSR, 2/OSR, 3/OSR, or 4/OSR narrow pulses during transmission.

49.3.7.1 Infrared transmit encoder

The infrared transmit encoder converts serial bits of data from transmit shift register to the TXD signal. A narrow pulse is transmitted for a 0 bit and no pulse for a 1 bit. The narrow pulse is sent at the start of the bit with a duration of 1/OSR, 2/OSR, 3/OSR, or 4/OSR of a bit time. A narrow low pulse is transmitted for a 0 bit when CTRL[TXINV] is cleared, while a narrow high pulse is transmitted for a 0 bit when CTRL[TXINV] is set.

49.3.7.2 Infrared receive decoder

The infrared receive block converts data from the RXD signal to the receive shift register. A narrow pulse is expected for each 0 received and no pulse is expected for each 1 received. A narrow low pulse is expected for a 0 bit when STAT[RXINV] is cleared, while a narrow high pulse is expected for a 0 bit when STAT[RXINV] is set. This receive decoder meets the edge jitter requirement as defined by the IrDA serial infrared physical layer specification.

49.3.7.3 Start bit detection

When STAT[RXINV] is cleared, the first falling edge of the received character corresponds to the start bit. The infrared decoder resets its counter. At this time, the receiver also begins its start bit detection process. After the start bit is detected, the receiver synchronizes its bit times to this start bit time. For the rest of the character reception, the infrared decoder's counter and the receiver's bit time counter count independently from each other.

49.3.7.4 Noise filtering

Any further rising edges detected during the first half of the infrared decoder counter are ignored by the decoder. Any pulses less than one oversampling baud clock can be undetected by it regardless of whether it is seen in the first or second half of the count.

49.3.7.5 Low-bit detection

During the second half of the decoder count, a rising edge is decoded as a 0, which is sent to the receiver. The decoder counter is also reset.

49.3.7.6 High-bit detection

At OSR oversampling baud rate clocks after the previous rising edge, if a rising edge is not seen, then the decoder sends a 1 to the receiver.

If the next bit is a 0, which arrives late, then a low-bit is detected according to Low-bit detection. The value sent to the receiver is changed from 1 to a 0. Then, if a noise pulse occurs outside the receiver's bit time sampling period, then the delay of a 0 is not recorded as noise.

49.3.8 Interrupts and status flags

The LPUART transmitter has two status flags that can optionally generate hardware interrupt requests. Transmit Data Register Empty (STAT[TDRE]) indicates when there is room in the transmit FIFO to write another transmit character to the DATA register. If the Transmit Interrupt Enable (CTRL[TIE]) bit is set, a hardware interrupt is requested when STAT[TDRE] is set. Transmit complete (STAT[TC]) indicates that the transmitter is finished transmitting all data, preamble, and break characters and is idle with TXD at the inactive level. This flag is often used in systems with modems to determine when it is safe to turn off the modem. If the transmit complete interrupt enable (CTRL[TCIE]) bit is set, a hardware interrupt is requested when STAT[TC] is set. Instead of hardware interrupts, software polling may be used to monitor the STAT[TDRE] and STAT[TC] status flags if the corresponding CTRL[TIE] or CTRL[TCIE] local interrupt masks are cleared.

When a program detects that the receive data register is full (STAT[RDRF] = 1), it gets the data from the receive data register by reading the DATA register. The STAT[RDRF] flag is cleared by reading the DATA register.

The IDLE status flag includes logic that prevents it from getting set repeatedly when the RXD line remains idle for an extended period of time. IDLE is cleared by writing 1 to the STAT[IDLE] flag. After STAT[IDLE] has been cleared, it cannot become set again until the receiver has received at least one new character and has set STAT[RDRF].

If the associated error is detected in the received character that caused STAT[RDRF] to be set, the error flags — noise flag (STAT[NF]), framing error (STAT[FE]), and parity error flag (STAT[PF]) — are set at the same time as STAT[RDRF]. These flags are not set in overrun cases.

If STAT[RDRF] is already set when a new character is ready to be transferred from the receive shifter to the receive FIFO, the overrun (STAT[OR]) flag is set instead of the data along with any associated STAT[NF], STAT[FE], or STAT[PF] condition is lost.

If the received character matches the contents of MATCH[MA1] and/or MATCH[MA2] then the STAT[MA1F] and/or STAT[MA2F] flags are set at the same time that STAT[RDRF] is set.

At any time, an active edge on the RXD serial data input pin causes the STAT[RXEDGIF] flag to set. The STAT[RXEDGIF] flag is cleared by writing a 1 to it. This function depends on the receiver being enabled (CTRL[RE] = 1).

49.3.9 Peripheral Triggers

The connection of the LPUART peripheral triggers with other peripherals are device specific.

49.3.9.1 Output Triggers

The LPUART generates the following output triggers that can be connected to other peripherals on the device.

- The transmit word trigger asserts at the end of each transmitted word, it negates after one bit period.
- The receive word trigger asserts at the end of each received word that is written to the receive FIFO, for one oversampling clock period.
- The receive idle trigger asserts when the idle flag would set, for one oversampling clock period.

49.3.9.2 Input Trigger

The LPUART supports one peripheral input trigger, that can be configured in one of the following ways.

- The CTS function must be enabled. The input trigger can be connected in place of the CTS_B pin input. The input trigger must assert for longer than one bit clock period when the transmitter is idle with data to send to guarantee a new transmission.
- The input trigger can modulate the transmit data output (trigger is logically ANDed with the TXD output). The input trigger is expected to be a free running clock (carrier signal) generated from a timer or PWM source with a frequency that is greater than the bit clock frequency. The carrier signal must not toggle faster than the maximum supported bit time.
- The input trigger can be connected in place of the RXD pin input. The input trigger is expected to be generated from a receive data source, such as analog comparator or external pin.

49.4 Clocking and Resets

Table 49-5. Clocks

LPUART Functional clock	The LPUART functional clock is asynchronous to the bus clock and can remain enabled in low power modes to support transmit and/or receive, including low power wakeups.
Bus clock	The bus clock is only used for bus accesses to the control and configuration registers. The bus clock frequency must be sufficient to support the data bandwidth requirements of the LPUART transmit and receive registers, including the FIFOs.

Table 49-6. Resets

Chip reset	The logic and registers for the LPUART transmitter and receiver are reset to their default state on a chip reset.
Software reset	Resets the LPUART logic and registers to their default state, except for the Global Register. The LPUART software reset is in the Global Register GLOBAL[RST].
FIFO reset	The LPUART implements write-only control bits that reset the transmit FIFO (FIFO[TXFLUSH]) and receive FIFO (FIFO[RXFLUSH]). After a FIFO is reset, that FIFO is empty.

49.5 External signals

Signal	Description	I/O
TXD	Transmit data. This pin is normally an output, but is an input (tristated) in single wire mode whenever the transmitter is disabled or transmit direction is configured for receive data.	I/O
RXD	Receive data.	I
CTS_B	Clear to send.	I
RTS_B	Request to send.	O

49.6 Register definition

The LPUART includes registers to control baud rate, select options, report status, and store transmit/receive data. Access to an address outside the valid memory map generates a bus error.

NOTE

Writing a Read-Only (RO) register or reading a Write-Only (WO) register can cause bus errors. This module does not check if programmed values in the registers are correct; the application software must ensure that valid programmed values are being written.

49.6.1 LPUART register descriptions

49.6.1.1 LPUART memory map

LPUART1 base address: 4018_4000h Serial6

LPUART2 base address: 4018_8000h Serial3

LPUART3 base address: 4018_C000h Serial2 (or Serial4 on MicroMod)

LPUART4 base address: 4019_0000h Serial4 (or Serial2 on MicroMod)

LPUART5 base address: 4019_4000h Serial8

LPUART6 base address: 4019_8000h Serial1

LPUART7 base address: 4019_C000h Serial7

LPUART8 base address: 401A_0000h Serial5

Offset	Register	Width (In bits)	Access	Reset value
0h	Version ID Register (VERID)	32	RO	0401_0003h
4h	Parameter Register (PARAM)	32	RO	0000_0202h
8h	LPUART Global Register (GLOBAL)	32	RW	0000_0000h
Ch	LPUART Pin Configuration Register (PINCFG)	32	RW	0000_0000h
10h	LPUART Baud Rate Register (BAUD)	32	RW	0F00_0004h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
14h	LPUART Status Register (STAT)	32	RW	00C0_0000h
18h	LPUART Control Register (CTRL)	32	RW	0000_0000h
1Ch	LPUART Data Register (DATA)	32	RW	0000_1000h
20h	LPUART Match Address Register (MATCH)	32	RW	0000_0000h
24h	LPUART Modem IrDA Register (MODIR)	32	RW	0000_0000h
28h	LPUART FIFO Register (FIFO)	32	RW	00C0_0011h
2Ch	LPUART Watermark Register (WATER)	32	RW	0000_0000h

49.6.1.2 Version ID Register (VERID)

49.6.1.2.1 Offset

Register	Offset
VERID	0h

49.6.1.2.2 Function

The Version ID register indicates the version integrated for this instance on the device and also indicates inclusion/exclusion of several optional features.

49.6.1.2.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	MAJOR								MINOR							
W																
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FEATURE															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1

49.6.1.2.4 Fields

Field	Function
31-24 MAJOR	Major Version Number This read-only field returns the major version number for the module specification.
23-16 MINOR	Minor Version Number This read-only field returns the minor version number for the module specification.
15-0 FEATURE	Feature Identification Number This read-only field returns the feature set number. 0000000000000001b - Standard feature set. 0000000000000011b - Standard feature set with MODEM/IrDA support.

49.6.1.3 Parameter Register (PARAM)

49.6.1.3.1 Offset

Register	Offset
PARAM	4h

49.6.1.3.2 Function

The Parameter register indicates the parameter configuration for this instance on the device.

49.6.1.3.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	RXFIFO								TXFIFO							
W																
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0

49.6.1.3.4 Fields

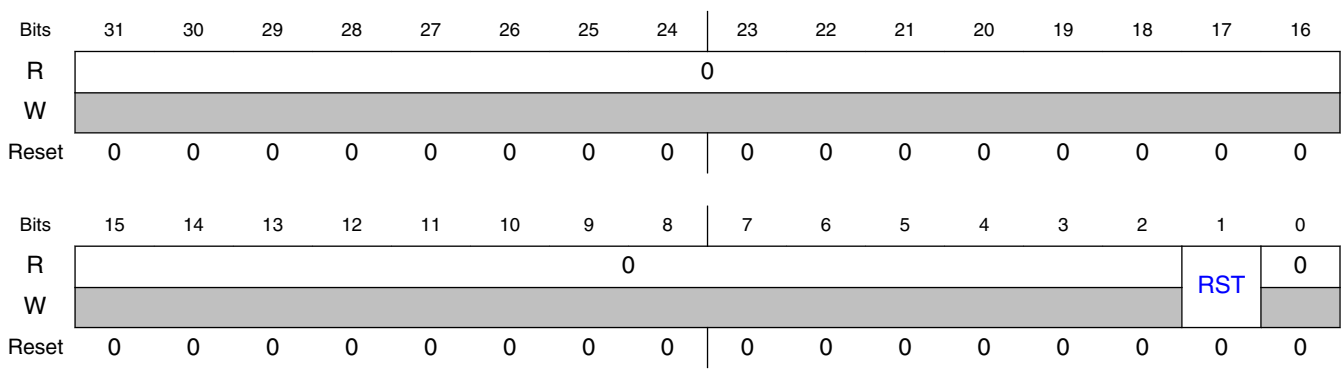
Field	Function
31-16 —	Reserved
15-8 RXFIFO	Receive FIFO Size The number of characters in the receive FIFO is 2^RXFIFO.
7-0 TXFIFO	Transmit FIFO Size The number of characters in the transmit FIFO is 2^TXFIFO.

49.6.1.4 LPUART Global Register (GLOBAL)

49.6.1.4.1 Offset

Register	Offset
GLOBAL	8h

49.6.1.4.2 Diagram



49.6.1.4.3 Fields

Field	Function
31-2 —	Reserved
1 RST	Software Reset

Table continues on the next page...

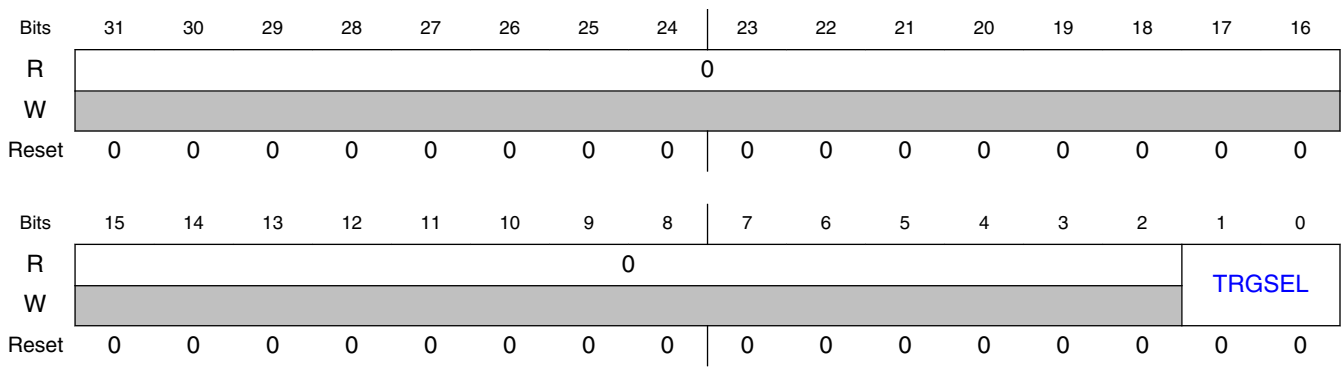
Field	Function
	Resets all internal logic and registers, except the Global Register. The reset takes effect immediately and remains asserted until negated by software. There is no minimum delay required before clearing the software reset. 0b - Module is not reset. 1b - Module is reset.
0 —	Reserved

49.6.1.5 LPUART Pin Configuration Register (PINCFG)

49.6.1.5.1 Offset

Register	Offset
PINCFG	Ch

49.6.1.5.2 Diagram



49.6.1.5.3 Fields

Field	Function
31-2 —	Reserved
1-0 TRGSEL	Trigger Select Configures the input trigger usage. This field should be changed only when the transmitter and receiver are both disabled. 00b - Input trigger is disabled. 01b - Input trigger is used instead of RXD pin input. 10b - Input trigger is used instead of CTS_B pin input.

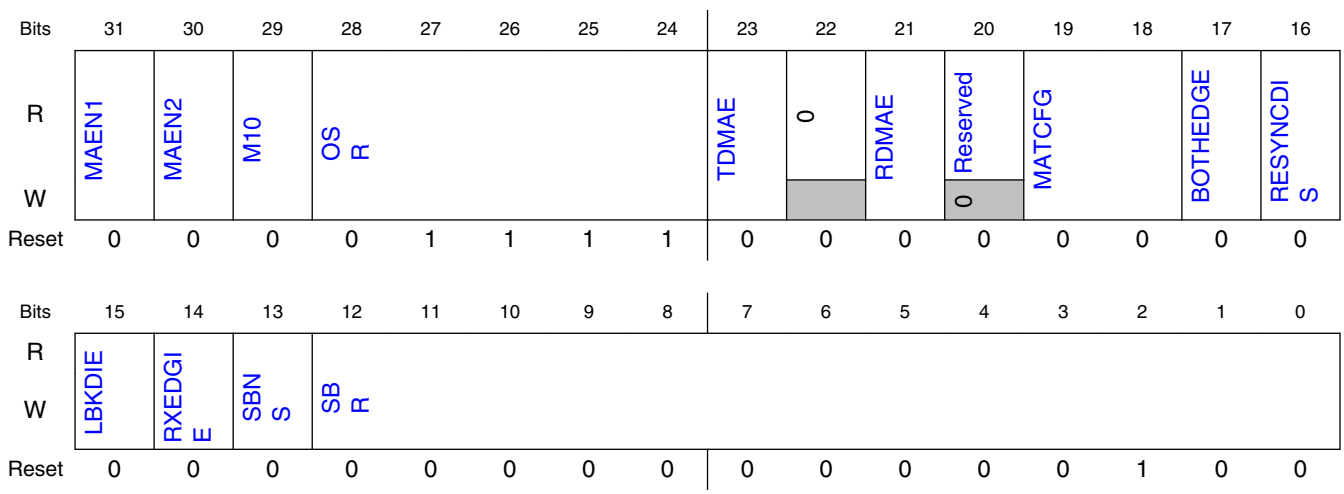
Field	Function
	11b - Input trigger is used to modulate the TXD pin output. The TXD pin output (after TXINV configuration) is internally ANDed with the input trigger.

49.6.1.6 LPUART Baud Rate Register (BAUD)

49.6.1.6.1 Offset

Register	Offset
BAUD	10h

49.6.1.6.2 Diagram



49.6.1.6.3 Fields

Field	Function
31 MAEN1	Match Address Mode Enable 1 0b - Normal operation. 1b - Enables automatic address matching or data matching mode for MATCH[MA1].
30 MAEN2	Match Address Mode Enable 2 0b - Normal operation. 1b - Enables automatic address matching or data matching mode for MATCH[MA2].
29 M10	10-bit Mode select The M10 bit causes a tenth bit to be part of the serial transmission. This bit should be changed only when the transmitter and receiver are both disabled.

Table continues on the next page...

Field	Function
	0b - Receiver and transmitter use 7-bit to 9-bit data characters. 1b - Receiver and transmitter use 10-bit data characters.
28-24 OSR	Oversampling Ratio This field configures the oversampling ratio for the receiver. This field should be changed only when the transmitter and receiver are both disabled. 00000b - Writing 0 to this field results in an oversampling ratio of 16 00001b - Reserved 00010b - Reserved 00011b - Oversampling ratio of 4, requires BOTHEDGE to be set. 00100b - Oversampling ratio of 5, requires BOTHEDGE to be set. 00101b - Oversampling ratio of 6, requires BOTHEDGE to be set. 00110b - Oversampling ratio of 7, requires BOTHEDGE to be set. 00111b - Oversampling ratio of 8. 01000b - Oversampling ratio of 9. 01001b - Oversampling ratio of 10. 01010b - Oversampling ratio of 11. 01011b - Oversampling ratio of 12. 01100b - Oversampling ratio of 13. 01101b - Oversampling ratio of 14. 01110b - Oversampling ratio of 15. 01111b - Oversampling ratio of 16. 10000b - Oversampling ratio of 17. 10001b - Oversampling ratio of 18. 10010b - Oversampling ratio of 19. 10011b - Oversampling ratio of 20. 10100b - Oversampling ratio of 21. 10101b - Oversampling ratio of 22. 10110b - Oversampling ratio of 23. 10111b - Oversampling ratio of 24. 11000b - Oversampling ratio of 25. 11001b - Oversampling ratio of 26. 11010b - Oversampling ratio of 27. 11011b - Oversampling ratio of 28. 11100b - Oversampling ratio of 29. 11101b - Oversampling ratio of 30. 11110b - Oversampling ratio of 31. 11111b - Oversampling ratio of 32.
23 TDMAE	Transmitter DMA Enable TDMAE configures the transmit data register empty flag, STAT[TDRE], to generate a DMA request. 0b - DMA request disabled. 1b - DMA request enabled.
22 —	Reserved
21 RDMAE	Receiver Full DMA Enable RDMAE configures the receiver data register full flag, STAT[RDRF], to generate a DMA request. 0b - DMA request disabled. 1b - DMA request enabled.
20 —	Reserved
19-18 MATCFG	Match Configuration Configures the match addressing mode used. This field should be changed only when the transmitter and receiver are both disabled.

Table continues on the next page...

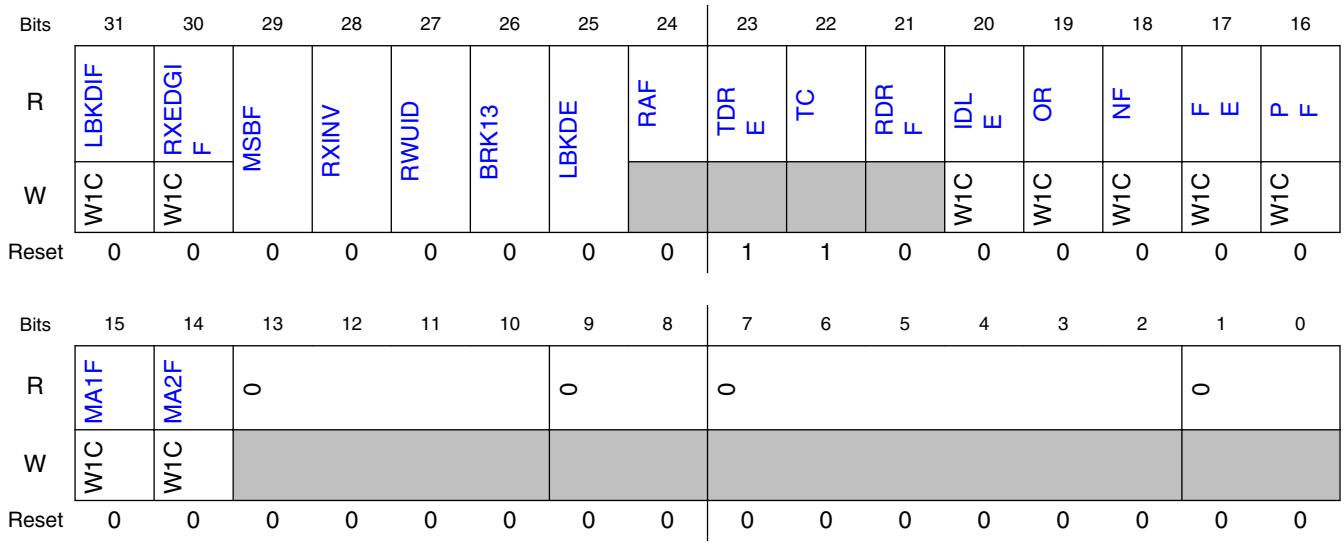
Field	Function
	00b - Address Match Wakeup 01b - Idle Match Wakeup 10b - Match On and Match Off 11b - Enables RWU on Data Match and Match On/Off for transmitter CTS input
17 BOTHEDGE	Both Edge Sampling Enables sampling of the received data on both edges of the baud rate clock, effectively doubling the number of times the receiver samples the input data for a given oversampling ratio. This bit must be set for oversampling ratios between x4 and x7 and is optional for higher oversampling ratios. This bit should be changed only when the receiver is disabled. 0b - Receiver samples input data using the rising edge of the baud rate clock. 1b - Receiver samples input data using the rising and falling edge of the baud rate clock.
16 RESYNCDIS	Resynchronization Disable When set, this field disables the resynchronization of the received data word when a data one followed by data zero transition is detected. This bit should be changed only when the receiver is disabled. 0b - Resynchronization during received data word is supported. 1b - Resynchronization during received data word is disabled.
15 LBKDIE	LIN Break Detect Interrupt Enable LBKDIE enables the LIN break detect flag, STAT[LBKDIF], to generate interrupt requests. 0b - Hardware interrupts from STAT[LBKDIF] flag are disabled (use polling). 1b - Hardware interrupt is requested when STAT[LBKDIF] flag is 1.
14 RXEDGIE	RX Input Active Edge Interrupt Enable Enables the receive input active edge, STAT[RXEDGIF], to generate interrupt requests. Changing CTRL[LOOPS] or CTRL[RSRC] when RXEDGIE is set can cause the STAT[RXEDGIF] flag to set. 0b - Hardware interrupts from STAT[RXEDGIF] are disabled. 1b - Hardware interrupt is requested when STAT[RXEDGIF] flag is 1.
13 SBNS	Stop Bit Number Select SBNS determines whether data characters have one or two stop bits. This bit should be changed only when the transmitter and receiver are both disabled. 0b - One stop bit. 1b - Two stop bits.
12-0 SBR	Baud Rate Modulo Divisor. The 13 bits in SBR[12:0] set the modulo divide rate for the baud rate generator. When SBR is 1 - 8191, the baud rate equals "baud clock / ((OSR+1) * SBR)". The 13-bit baud rate setting [SBR12:SBR0] must be updated only when the transmitter and receiver are both disabled (CTRL[RE] and CTRL[TE] are both 0).

49.6.1.7 LPUART Status Register (STAT)

49.6.1.7.1 Offset

Register	Offset
STAT	14h

49.6.1.7.2 Diagram



49.6.1.7.3 Fields

Field	Function
31 LBKDIF	LIN Break Detect Interrupt Flag LBKDIF is set when the LIN break detect circuitry is enabled and a LIN break character is detected. Writing a 1 clears the LBKDIF field. 0b - No LIN break character has been detected. 1b - LIN break character has been detected.
30 RXEDGIF	RXD Pin Active Edge Interrupt Flag RXEDGIF is set whenever the receiver is enabled and an active edge (falling if RXINV = 0, rising if RXINV=1,) on the RXD pin occurs. To clear RXEDGIF, write a 1 to the RXEDGIF field. 0b - No active edge on the receive pin has occurred. 1b - An active edge on the receive pin has occurred.
29 MSBF	MSB First Setting this bit reverses the order of the bits that are transmitted and received on the wire. This bit does not affect the polarity of the bits, the location of the parity bit, or the location of the start or stop bits. This bit should be changed only when the transmitter and receiver are both disabled. 0b - LSB (bit0) is the first bit that is transmitted following the start bit. Further, the first bit received after the start bit is identified as bit0. 1b - MSB (identified as bit9, bit8, bit7 or bit6) is the first bit that is transmitted following the start bit depending on the setting of CTRL[M], CTRL[PE] and BAUD[M10]. .
28 RXINV	Receive Data Inversion Setting this bit reverses the polarity of the received data input. This bit should be changed only when the receiver is disabled. **NOTE:** Setting RXINV inverts the RXD input for all cases: data bits, start and stop bits, break, and idle. 0b - Receive data not inverted. 1b - Receive data inverted.
27 RWUID	Receive Wake Up Idle Detect

Table continues on the next page...

Field	Function
	For RWU on idle character, RWUID controls whether the idle character that wakes up the receiver sets the IDLE bit. For address match wakeup, RWUID controls if the IDLE bit is set when the address does not match. This bit should be changed only when the receiver is disabled. 0b - During receive standby state (RWU = 1), the IDLE bit does not get set upon detection of an idle character. During address match wakeup, the IDLE bit does not set when an address does not match. 1b - During receive standby state (RWU = 1), the IDLE bit gets set upon detection of an idle character. During address match wakeup, the IDLE bit does set when an address does not match.
26 BRK13	Break Character Generation Length BRK13 selects a longer transmitted break character length. The state of this bit does not affect the detection of a framing error. This bit should be changed only when the transmitter is disabled. A break character can be sent by setting CTRL[SBK] or by writing the transmit FIFO with DATA[FRETSC] set and DATA[R9T9] clear. 0b - Break character is transmitted with length of 9 to 13 bit times. 1b - Break character is transmitted with length of 12 to 15 bit times.
25 LBKDE	LIN Break Detection Enable LBKDE selects a longer break character detection length. While LBKDE is set, receive data is not stored in the receive FIFO. **NOTE:** The LBKDE bit enables the LIN break detect circuit and disables writing receive data to FIFO. So it essentially ignores all characters except a LIN break. 0b - LIN break detect is disabled, normal break character can be detected. 1b - LIN break detect is enabled. LIN break character is detected at length of 11 bit times (if M = 0) or 12 (if M = 1) or 13 (M10 = 1).
24 RAF	Receiver Active Flag RAF is set when the receiver detects the beginning of a valid start bit, and RAF is cleared automatically when the receiver detects an idle line. 0b - LPUART receiver idle waiting for a start bit. 1b - LPUART receiver active (RXD input not idle).
23 TDRE	Transmit Data Register Empty Flag When the transmit FIFO is enabled, TDRE is set when the number of datawords in the transmit FIFO is equal to or less than the number indicated by WATER[TXWATER]. To clear TDRE, write to the DATA register until the number of words in the transmit FIFO is greater than the number indicated by WATER[TXWATER]. When the transmit FIFO is disabled, TDRE is set when the transmit DATA register is empty. To clear TDRE, write to the DATA register. TDRE is not affected by a character that is in the process of being transmitted; it is updated at the start of each transmitted character. 0b - Transmit FIFO level is greater than watermark. 1b - Transmit FIFO level is equal or less than watermark.
22 TC	Transmission Complete Flag TC is cleared when there is a transmission in progress or when a preamble or break character is loaded. TC is set when the transmit buffer is empty and no data, preamble, or break character is being transmitted. When TC is set, the transmit data output signal becomes idle (logic 1). TC is cleared by writing to the DATA register to transmit new data, queuing a preamble by clearing and then setting CTRL[TE], queuing a break character by writing 1 to CTRL[SBK]. 0b - Transmitter active (sending data, a preamble, or a break). 1b - Transmitter idle (transmission activity complete).
21 RDRF	Receive Data Register Full Flag

Table continues on the next page...

Field	Function
	When the receive FIFO is enabled, RDRF is set when the number of datawords in the receive buffer is greater than the number indicated by WATER[RXWATER]. To clear RDRF, read the DATA register until the number of datawords in the receive FIFO is equal to or less than the number indicated by WATER[RXWATER]. When the receive FIFO is disabled, RDRF is set when the receive buffer (the DATA register) is full. To clear RDRF, read the DATA register. A character that is in the process of being received does not cause a change in RDRF until the entire character is received. Even if RDRF is set, the character continues to be received until an overrun condition occurs once the entire character is received. 0b - Receive FIFO level is less than watermark. 1b - Receive FIFO level is equal or greater than watermark.
20 IDLE	Idle Line Flag IDLE is set when the LPUART receive line becomes idle for a full character time after a period of activity. When CTRL[ILT] is cleared, the receiver starts counting idle bit times after the start bit. If the receive character is all 1s, these bit times and the stop bits time count toward the full character time of logic high, 10 to 13 bit times, needed for the receiver to detect an idle line. When CTRL[ILT] is set, the receiver doesn't start counting idle bit times until after the stop bits. The stop bits and any logic high bit times at the end of the previous character do not count toward the full character time of logic high needed for the receiver to detect an idle line. To clear IDLE, write logic 1 to the IDLE flag. After IDLE has been cleared, it cannot be set again until after a new character is stored in the receive buffer or a LIN break character has set the LBKDIF flag . IDLE is set only once even if the receive line remains idle for an extended period. 0b - No idle line detected. 1b - Idle line is detected.
19 OR	Receiver Overrun Flag OR is set when software fails to prevent the receive data register from overflowing with data. The OR bit is set immediately after the stop bit has been completely received for the dataword that overflows the buffer and all the other error flags (FE, NF, and PF) are prevented from setting. The data in the shift register is lost, but the data already in the LPUART data registers is not affected. If LBKDE is enabled and a LIN Break is detected, the OR field asserts if LBKDIF is not cleared before the next data character is received. While the OR flag is set, no additional data is stored in the receive FIFO even if sufficient room exists. To clear OR, write logic 1 to the OR flag. 0b - No overrun. 1b - Receive overrun (new LPUART data lost).
18 NF	Noise Flag The advanced sampling technique used in the receiver takes three samples in each of the received bits. If any of these samples disagrees with the rest of the samples within any bit time in the frame then noise is detected for that character. NF is set whenever the next character to be read from the DATA register is received with noise detected within the character. To clear NF, write logic 1 to the NF field. 0b - No noise detected. 1b - Noise detected in the received character in the DATA register.
17 FE	Framing Error Flag FE is set whenever the next character to be read from the DATA register is received with logic 0 detected where a stop bit was expected. To clear FE, write logic 1 to the FE field. 0b - No framing error detected. This does not guarantee the framing is correct. 1b - Framing error.
16 PF	Parity Error Flag PF is set whenever the next character to be read from the DATA register is received when parity is enabled (PE = 1) and the parity bit in the received character does not agree with the expected parity value. To clear PF, write a logic 1 to the PF field.

Table continues on the next page...

Field	Function
	0b - No parity error. 1b - Parity error.
15 MA1F	Match 1 Flag MA1F is set whenever the next character to be read from the DATA register matches MA1. To clear MA1F, write a logic 1 to the MA1F field. 0b - Received data is not equal to MA1 1b - Received data is equal to MA1
14 MA2F	Match 2 Flag MA2F is set whenever the next character to be read from the DATA register matches MA2. To clear MA2F, write a logic 1 to the MA2F field. 0b - Received data is not equal to MA2 1b - Received data is equal to MA2
13-10 —	Reserved
9-8 —	Reserved
7-2 —	Reserved
1-0 —	Reserved

49.6.1.8 LPUART Control Register (CTRL)

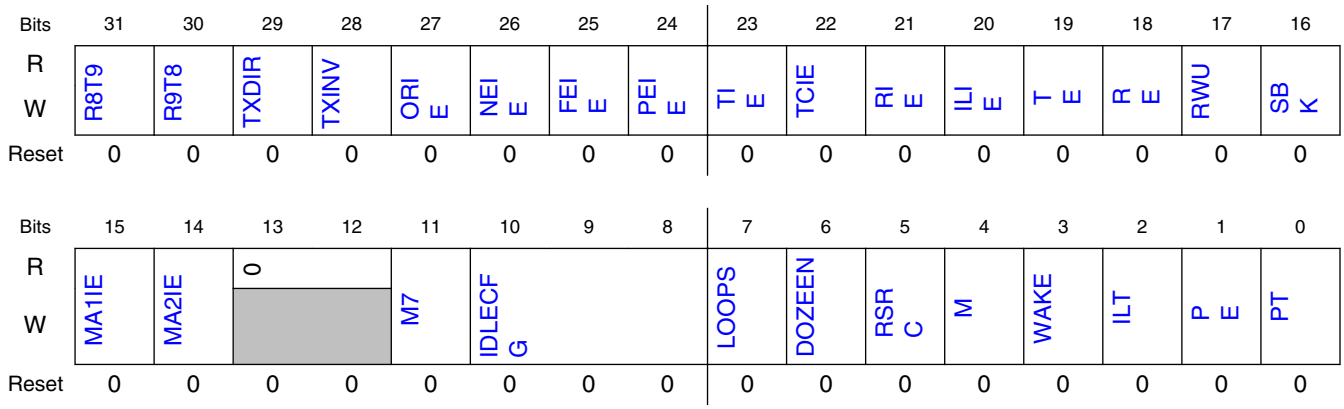
49.6.1.8.1 Offset

Register	Offset
CTRL	18h

49.6.1.8.2 Function

This read/write register controls various optional features of the LPUART system. This register should only be altered when the transmitter and receiver are both disabled.

49.6.1.8.3 Diagram



49.6.1.8.4 Fields

Field	Function
31 R8T9	Receive Bit 8 / Transmit Bit 9 R8 is the ninth data bit received when the LPUART is configured for 9-bit or 10-bit data formats. When reading 9-bit or 10-bit data, read R8 before reading the DATA register. T9 is the tenth data bit transmitted when the LPUART is configured for 10-bit data formats. When writing 10-bit data, write T9 before writing the DATA register. If T9 does not need to change from its previous value, such as when it is used to generate address mark or parity, then it need not be written each time the DATA register is written. **NOTE:** R8 is a read-only bit and T9 is a write-only bit, the value read is different from the value written.
30 R9T8	Receive Bit 9 / Transmit Bit 8 R9 is the tenth data bit received when the LPUART is configured for 10-bit data formats. When reading 10-bit data, read R9 before reading the DATA register T8 is the ninth data bit transmitted when the LPUART is configured for 9-bit or 10-bit data formats. When writing 9-bit or 10-bit data, write T8 before writing the DATA register. If T8 does not need to change from its previous value, such as when it is used to generate address mark or parity, then it need not be written each time the DATA register is written. **NOTE:** R9 is a read-only bit and T8 is a write-only bit, the value read is different from the value written.
29 TXDIR	TXD Pin Direction in Single-Wire Mode When the LPUART is configured for single-wire half-duplex operation (LOOPS = RSRC = 1), this bit determines the direction of data at the TXD pin. When clearing TXDIR, the transmitter finishes receiving the current character (if any) before the receiver starts receiving data from the TXD pin. 0b - TXD pin is an input in single-wire mode. 1b - TXD pin is an output in single-wire mode.
28 TXINV	Transmit Data Inversion Setting this bit reverses the polarity of the transmitted data output. **NOTE:** Setting TXINV inverts the TXD output for all cases: data bits, start and stop bits, break, and idle. 0b - Transmit data not inverted. 1b - Transmit data inverted.
27	Overrun Interrupt Enable

Table continues on the next page...

Field	Function
ORIE	This bit enables the overrun flag (OR) to generate hardware interrupt requests. 0b - OR interrupts disabled; use polling. 1b - Hardware interrupt is requested when OR is set.
26 NEIE	Noise Error Interrupt Enable This bit enables the noise flag (NF) to generate hardware interrupt requests. 0b - NF interrupts disabled; use polling. 1b - Hardware interrupt is requested when NF is set.
25 FEIE	Framing Error Interrupt Enable This bit enables the framing error flag (FE) to generate hardware interrupt requests. 0b - FE interrupts disabled; use polling. 1b - Hardware interrupt is requested when FE is set.
24 PEIE	Parity Error Interrupt Enable This bit enables the parity error flag (PF) to generate hardware interrupt requests. 0b - PF interrupts disabled; use polling). 1b - Hardware interrupt is requested when PF is set.
23 TIE	Transmit Interrupt Enable Enables STAT[TDRE] to generate interrupt requests. 0b - Hardware interrupts from TDRE disabled. 1b - Hardware interrupt is requested when TDRE flag is 1.
22 TCIE	Transmission Complete Interrupt Enable for TCIE enables the transmission complete flag, TC, to generate interrupt requests. 0b - Hardware interrupts from TC disabled. 1b - Hardware interrupt is requested when TC flag is 1.
21 RIE	Receiver Interrupt Enable Enables STAT[RDRF] to generate interrupt requests. 0b - Hardware interrupts from RDRF disabled. 1b - Hardware interrupt is requested when RDRF flag is 1.
20 ILIE	Idle Line Interrupt Enable ILIE enables the idle line flag, STAT[IDLE], to generate interrupt requests. 0b - Hardware interrupts from IDLE disabled; use polling. 1b - Hardware interrupt is requested when IDLE flag is 1.
19 TE	Transmitter Enable Enables the LPUART transmitter. TE can also be used to queue an idle preamble by clearing and then setting TE. When TE is cleared, this register bit reads as 1 until the transmitter has completed the current character and the TXD pin is tristated. A single idle character can also be queued by writing to the transmit FIFO with DATA[FRETSC] set and DATA[R9T9] set. 0b - Transmitter disabled. 1b - Transmitter enabled.
18 RE	Receiver Enable Enables the LPUART receiver. When RE is written to 0, this register bit reads as 1 until the receiver finishes receiving the current character (if any). 0b - Receiver disabled. 1b - Receiver enabled.
17 RWU	Receiver Wakeup Control

Table continues on the next page...

Field	Function
	This field can be set to place the LPUART receiver in a standby state. RWU automatically clears when an RWU event occurs, that is, an IDLE event when CTRL[WAKE] is clear or an address match when CTRL[WAKE] is set and STAT[RWUID] is clear. **NOTE:** RWU must be set only with CTRL[WAKE] = 0 (wakeup on idle) if the channel is currently not idle. This can be determined by STAT[RAF]. If the flag is set to wake up an IDLE event and the channel is already idle, it is possible that the LPUART will discard data. This is because the data must be received or a LIN break detected after an IDLE is detected before IDLE is allowed to be reasserted. 0b - Normal receiver operation. 1b - LPUART receiver in standby waiting for wakeup condition.
16 SBK	Send Break Writing a 1 and then a 0 to SBK queues a break character in the transmit data stream. Additional break characters of 9 to 13 bits, or 12 to 15 bits if STAT[BRK13] is set, bit times of logic 0 are queued as long as SBK is set. Depending on the timing of the set and clear of SBK relative to the character currently being transmitted, a second break character may be queued before software clears SBK. If software takes too long to clear SBK (for example, it is not cleared by the end of the 1st break character), then a 2nd break character is sent. This is compared to queuing a break character through the transmit FIFO which guarantees only one is sent. A single break character can also be queued by writing to the transmit FIFO with DATA[FRETSC] set and DATA[R9T9] clear. 0b - Normal transmitter operation. 1b - Queue break character(s) to be sent.
15 MA1IE	Match 1 Interrupt Enable 0b - MA1F interrupt disabled 1b - MA1F interrupt enabled
14 MA2IE	Match 2 Interrupt Enable 0b - MA2F interrupt disabled 1b - MA2F interrupt enabled
13-12 —	Reserved
11 M7	7-Bit Mode Select This bit should be changed only when the transmitter and receiver are both disabled. 0b - Receiver and transmitter use 8-bit to 10-bit data characters. 1b - Receiver and transmitter use 7-bit data characters.
10-8 IDLECFG	Idle Configuration Configures the number of idle characters that must be received before the IDLE flag is set. 000b - 1 idle character 001b - 2 idle characters 010b - 4 idle characters 011b - 8 idle characters 100b - 16 idle characters 101b - 32 idle characters 110b - 64 idle characters 111b - 128 idle characters
7 LOOPS	Loop Mode Select When LOOPS is set, the RXD pin is disconnected from the LPUART and the transmitter output is internally connected to the receiver input. The transmitter and the receiver must be enabled to use the loop function. 0b - Normal operation - RXD and TXD use separate pins.

Table continues on the next page...

Field	Function
	1b - Loop mode or single-wire mode where transmitter outputs are internally connected to receiver input (see RSRC bit).
6 DOZEEN	Doze Enable 0b - LPUART is enabled in Doze mode. 1b - LPUART is disabled in Doze mode .
5 RSRC	Receiver Source Select This field has no meaning or effect unless the LOOPS field is set. When LOOPS is set, the RSRC field determines the source for the receiver shift register input. 0b - Provided LOOPS is set, RSRC is cleared, selects internal loop back mode and the LPUART does not use the RXD pin. 1b - Single-wire LPUART mode where the TXD pin is connected to the transmitter output and receiver input.
4 M	9-Bit or 8-Bit Mode Select 0b - Receiver and transmitter use 8-bit data characters. 1b - Receiver and transmitter use 9-bit data characters.
3 WAKE	Receiver Wakeup Method Select Determines which condition wakes the LPUART when RWU=1: • Address mark in the bit preceding the stop bit (or bit preceding the parity bit when parity is enabled) of the received data character. • An idle condition on the receive pin input signal. 0b - Configures RWU for idle-line wakeup. 1b - Configures RWU with address-mark wakeup.
2 ILT	Idle Line Type Select Determines when the receiver starts counting logic 1s as idle character bits. The count begins either after a valid start bit or after the stop bit. If the count begins after the start bit, then a string of logic 1s preceding the stop bit can cause false recognition of an idle character. Beginning the count after the stop bit avoids false idle character recognition, but requires properly synchronized transmissions. **NOTE:** In case the LPUART is programmed with ILT = 1, a logic 0 is automatically shifted after a received stop bit, therefore resetting the idle count. 0b - Idle character bit count starts after start bit. 1b - Idle character bit count starts after stop bit.
1 PE	Parity Enable Enables hardware parity generation and checking. When parity is enabled, the bit immediately before the stop bit is treated as the parity bit. 0b - No hardware parity generation or checking. 1b - Parity enabled.
0 PT	Parity Type Provided parity is enabled (PE = 1), this bit selects even or odd parity. Odd parity means the total number of logic 1 bits in the data character, including the parity bit, is odd. Even parity means the total number of 1s in the data character, including the parity bit, is even. 0b - Even parity. 1b - Odd parity.

49.6.1.9 LPUART Data Register (DATA)

49.6.1.9.1 Offset

Register	Offset
DATA	1Ch

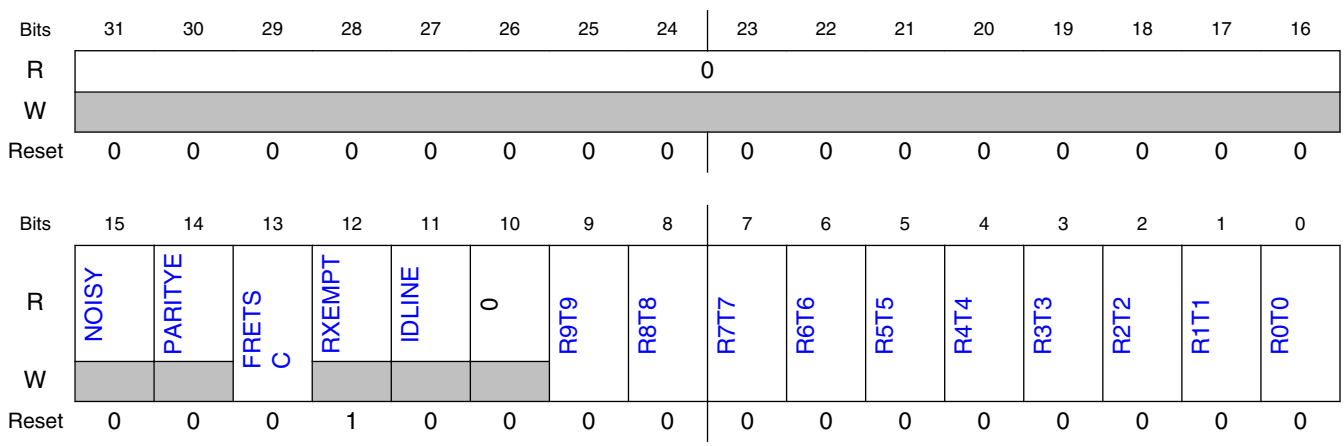
49.6.1.9.2 Function

NOTE

This register is two separate registers. Reads return the contents of the read-only receive FIFO and writes go to the write-only transmit FIFO.

Reads and writes of this register are also involved in the automatic flag clearing mechanisms for some of the LPUART status flags. This register can be written using 8-bit, 16-bit or 32-bit writes. An 8-bit write to DATA[7:0] will push { R8T9, R9T8, DATA[7:0] } the transmit FIFO with TSC clear. A 16-bit or 32-bit write will push the data written into the FIFO and does not update the value in R8T9 or R9T8.

49.6.1.9.3 Diagram



49.6.1.9.4 Fields

Field	Function
31-16 —	Reserved
15 NOISY	Noisy Data Received The current received dataword contained in DATA[R9:R0] is received with noise. 0b - The dataword is received without noise.

Table continues on the next page...

Field	Function
	1b - The data is received with noise.
14 PARITYE	Parity Error The current received dataword contained in DATA[R9:R0] is received with a parity error. 0b - The dataword is received without a parity error. 1b - The dataword is received with a parity error.
13 FRETSC	Frame Error / Transmit Special Character For reads, indicates the current received dataword contained in DATA[R9:R0] was received with a frame error. For writes, indicates a break or idle character is to be transmitted instead of the contents in DATA[T9:T0]. T9 is used to indicate a break character when 0 and a idle character when 1, the contents of DATA[T8:T0] should be zero. 0b - The dataword is received without a frame error on read, or transmit a normal character on write. 1b - The dataword is received with a frame error, or transmit an idle or break character on transmit.
12 RXEMPT	Receive Buffer Empty Asserts when there is no data in the receive buffer. This field does not take into account data that is in the receive shift register. 0b - Receive buffer contains valid data. 1b - Receive buffer is empty, data returned on read is not valid.
11 IDLINE	Idle Line Indicates the receiver line was idle before receiving the character in DATA[9:0]. Unlike the IDLE flag, this bit can set for the first character received when the receiver is first enabled. 0b - Receiver was not idle before receiving this character. 1b - Receiver was idle before receiving this character.
10 —	Reserved
9 R9T9	R9T9 Read receive FIFO bit 9 or write transmit FIFO bit 9.
8 R8T8	R8T8 Read receive FIFO bit 8 or write transmit FIFO bit 8.
7 R7T7	R7T7 Read receive FIFO bit 7 or write transmit FIFO bit 7.
6 R6T6	R6T6 Read receive FIFO bit 6 or write transmit FIFO bit 6.
5 R5T5	R5T5 Read receive FIFO bit 5 or write transmit FIFO bit 5.
4 R4T4	R4T4 Read receive FIFO bit 4 or write transmit FIFO bit 4.
3 R3T3	R3T3 Read receive FIFO bit 3 or write transmit FIFO bit 3.
2 R2T2	R2T2 Read receive FIFO bit 2 or write transmit FIFO bit 2.
1 R1T1	R1T1 Read receive FIFO bit 1 or write transmit FIFO bit 1.
0	R0T0

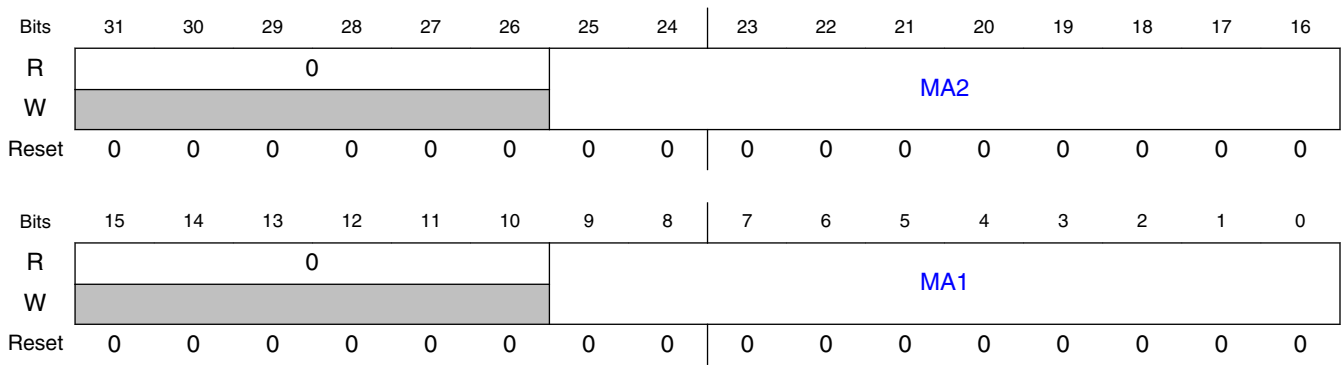
Field	Function
R0T0	Read receive FIFO bit 0 or write transmit FIFO bit 0.

49.6.1.10 LPUART Match Address Register (MATCH)

49.6.1.10.1 Offset

Register	Offset
MATCH	20h

49.6.1.10.2 Diagram



49.6.1.10.3 Fields

Field	Function
31-26 —	Reserved
25-16 MA2	Match Address 2 The MA1 and MA2 fields are compared to input data addresses when the most significant bit is set and the associated BAUD[MAEN] bit is set. If a match occurs, the following data is transferred to the DATA register. If a match fails, the following data is discarded. Software should only write a MAx field when the associated BAUD[MAEN] bit is clear.
15-10 —	Reserved
9-0 MA1	Match Address 1 The MA1 and MA2 fields are compared to input data addresses when the most significant bit is set and the associated BAUD[MAEN] bit is set. If a match occurs, the following data is transferred to the DATA register. If a match fails, the following data is discarded. Software should only write a MAx field when the associated BAUD[MAEN] bit is clear.

49.6.1.11 LPUART Modem IrDA Register (MODIR)

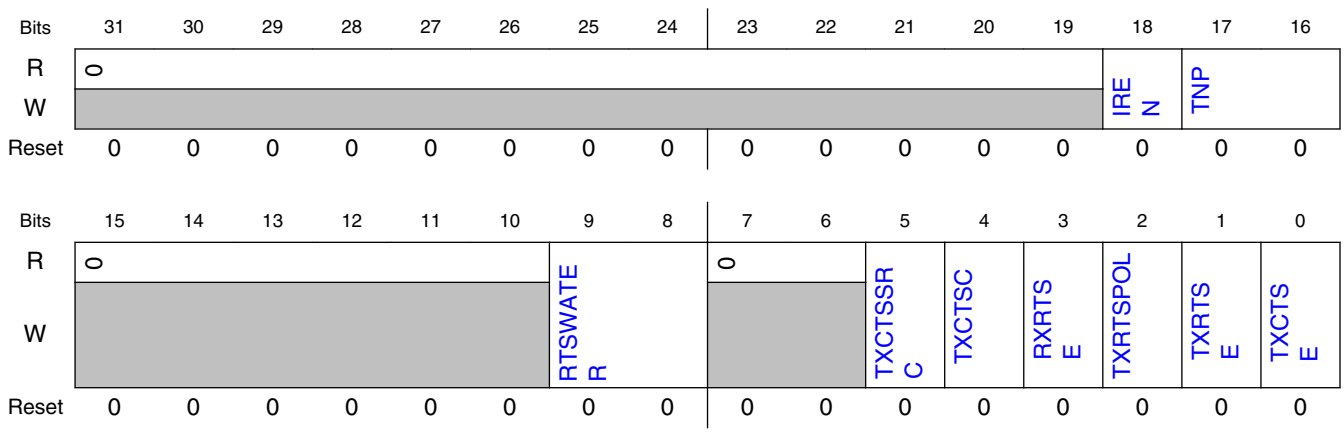
49.6.1.11.1 Offset

Register	Offset
MODIR	24h

49.6.1.11.2 Function

The MODIR register controls options for setting the modem configuration.

49.6.1.11.3 Diagram



49.6.1.11.4 Fields

Field	Function
31-19 —	Reserved
18 IREN	Infrared enable Enables/disables the infrared modulation/demodulation. This bit should be changed only when the transmitter and receiver are both disabled. 0b - IR disabled. 1b - IR enabled.
17-16 TNP	Transmitter narrow pulse Configures whether the LPUART transmits a 1/OSR, 2/OSR, 3/OSR or 4/OSR narrow pulse when IR is enabled. This bit should be changed only when the transmitter and receiver are both disabled.

Table continues on the next page...

Field	Function
	The IR pulse width should be configured to less than half of the oversampling ratio. Common pulse widths are 3/16, 1/16, 1/32 or 1/4 of the bit length. These can be configured by selecting the appropriate oversample ratio and pulse width. 00b - 1/OSR. 01b - 2/OSR. 10b - 3/OSR. 11b - 4/OSR.
15-10 —	Reserved
9-8 RTSWATER	Receive RTS Configuration Configures the assertion and negation of the RX RTS_B output. The RX RTS_B output negates when the number of empty words in the receive FIFO is greater or equal to RTSWATER. If RTSWATER is configured to zero, RTS_B negates when the receive FIFO is full. For the purpose of RX RTS_B generation, the number of words in the receive FIFO updates when a start bit is detected. This supports additional latency between RTS_B negation and the external transmitter ceasing transmission. If both RX RTS_B and address/data matching is enabled, then RTS_B could assert at the end of a character if there is not a match. This field should be changed only when the receiver is disabled.
7-6 —	Reserved
5 TXCTSSRC	Transmit CTS Source Configures the source of the CTS input. 0b - CTS input is the CTS_B pin. 1b - CTS input is an internal connection to the receiver address match result.
4 TXCTSC	Transmit CTS Configuration Configures if the CTS state is checked at the start of each character or only when the transmitter is idle. 0b - CTS input is sampled at the start of each character. 1b - CTS input is sampled when the transmitter is idle.
3 RXRTSE	Receiver request-to-send enable Allows the RTS output to control the CTS input of the transmitting device to prevent receiver overrun. This bit should be changed only when the receiver is disabled. **NOTE:** Do not set both RXRTSE and TXRTSE. 0b - The receiver has no effect on RTS. 1b - RTS is deasserted if the receiver data register is full or a start bit has been detected that would cause the receiver data register to become full. RTS is asserted if the receiver data register is not full and has not detected a start bit that would cause the receiver data register to become full.
2 TXRTSPOL	Transmitter request-to-send polarity Controls the polarity of the transmitter RTS. TXRTSPOL does not affect the polarity of the receiver RTS. RTS remains negated in the active low state unless TXRTSE is set. This bit should be changed only when the transmitter is disabled. 0b - Transmitter RTS is active low. 1b - Transmitter RTS is active high.
1 TXRTSE	Transmitter request-to-send enable Controls RTS before and after a transmission. This bit should be changed only when the transmitter is disabled. 0b - The transmitter has no effect on RTS.

Table continues on the next page...

Field	Function
	1b - When a character is placed into an empty transmit shift register, RTS asserts one bit time before the start bit is transmitted. RTS deasserts one bit time after all characters in the transmitter FIFO and shift register are completely sent, including the last stop bit.
0 TXCTSE	Transmitter clear-to-send enable TXCTSE controls the operation of the transmitter. TXCTSE can be set independently from the state of TXRTSE and RXRTSE. 0b - CTS has no effect on the transmitter. 1b - Enables clear-to-send operation. The transmitter checks the state of CTS each time it is ready to send a character. If CTS is asserted, the character is sent. If CTS is deasserted, the signal TXD remains in the mark state and transmission is delayed until CTS is asserted. Changes in CTS as a character is being sent do not affect its transmission.

49.6.1.12 LPUART FIFO Register (FIFO)

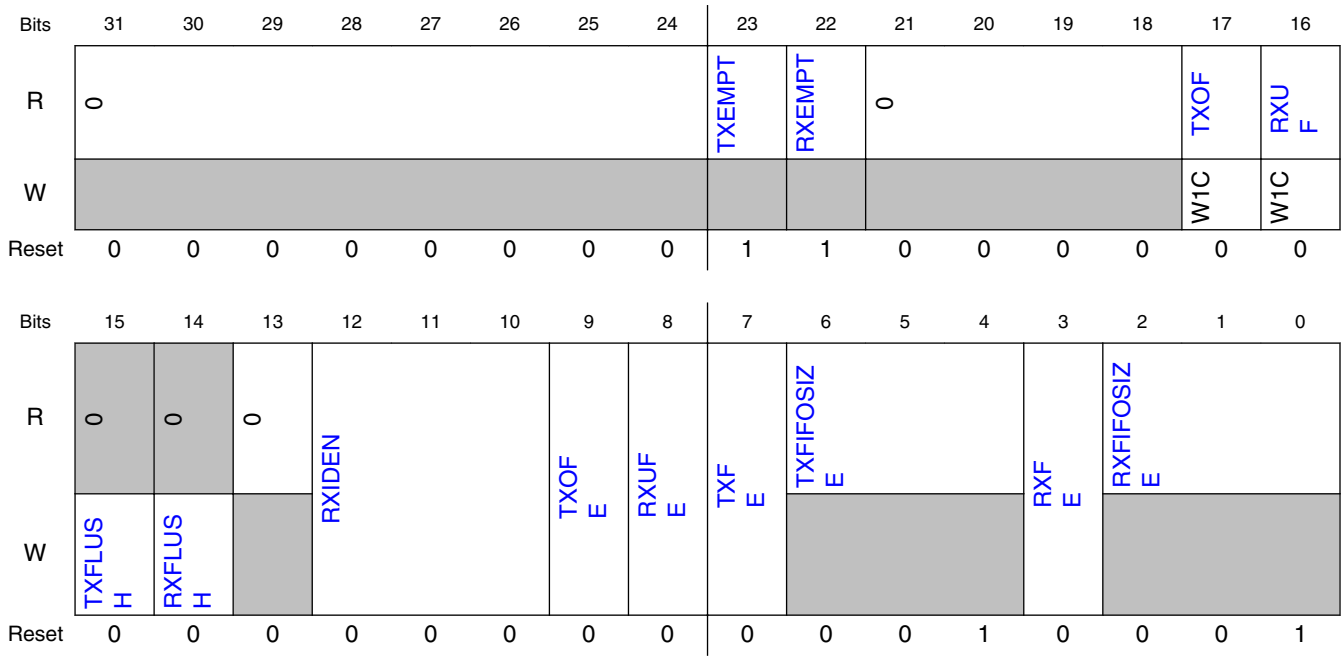
49.6.1.12.1 Offset

Register	Offset
FIFO	28h

49.6.1.12.2 Function

This register provides the ability for the programmer to turn on and off FIFO functionality. It also provides the size of the FIFO that has been implemented. This register may be read at any time. This register must be written only when CTRL[RE] and CTRL[TE] are cleared/not set and when the FIFO is empty.

49.6.1.12.3 Diagram



49.6.1.12.4 Fields

Field	Function
31-24 —	Reserved
23 TXEMPT	Transmit FIFO/Buffer Empty Asserts when there is no data in the Transmit FIFO/Buffer. This field does not take into account data that is in the transmit shift register. 0b - Transmit buffer is not empty. 1b - Transmit buffer is empty.
22 RXEMPT	Receive FIFO/Buffer Empty Asserts when there is no data in the Receive FIFO/Buffer. This field does not take into account data that is in the receive shift register. 0b - Receive buffer is not empty. 1b - Receive buffer is empty.
21-18 —	Reserved
17 TXOF	Transmitter FIFO Overflow Flag Indicates that more data has been written to the transmit FIFO than it can hold. This field asserts regardless of the value of TXOFE. However, an interrupt is issued to the host only if TXOFE is set. Write 1 to clear this flag. 0b - No transmit FIFO overflow has occurred since the last time the flag was cleared. 1b - At least one transmit FIFO overflow has occurred since the last time the flag was cleared.
16	Receiver FIFO Underflow Flag

Table continues on the next page...

Field	Function
RXUF	Indicates that more data has been read from the receive FIFO than was present. This field asserts regardless of the value of RXUFE. However, an interrupt is issued to the host only if RXUFE is set. Write a logic 1 to clear RXUF. 0b - No receive FIFO underflow has occurred since the last time the flag was cleared. 1b - At least one receive FIFO underflow has occurred since the last time the flag was cleared.
15 TXFLUSH	Transmit FIFO Flush Writing to this field causes all data that is stored in the transmit FIFO to be flushed. This does not affect data that is in the transmit shift register. 0b - No flush operation occurs. 1b - All data in the transmit FIFO is cleared out.
14 RXFLUSH	Receive FIFO Flush Writing to this field causes all data that is stored in the receive FIFO to be flushed. This does not affect data that is in the receive shift register. 0b - No flush operation occurs. 1b - All data in the receive FIFO/buffer is cleared out.
13 —	Reserved
12-10 RXIDEN	Receiver Idle Empty Enable When set, enables the assertion of RDRF when the receiver is idle for a number of idle characters and the FIFO is not empty. 000b - Disable RDRF assertion due to partially filled FIFO when receiver is idle. 001b - Enable RDRF assertion due to partially filled FIFO when receiver is idle for 1 character. 010b - Enable RDRF assertion due to partially filled FIFO when receiver is idle for 2 characters. 011b - Enable RDRF assertion due to partially filled FIFO when receiver is idle for 4 characters. 100b - Enable RDRF assertion due to partially filled FIFO when receiver is idle for 8 characters. 101b - Enable RDRF assertion due to partially filled FIFO when receiver is idle for 16 characters. 110b - Enable RDRF assertion due to partially filled FIFO when receiver is idle for 32 characters. 111b - Enable RDRF assertion due to partially filled FIFO when receiver is idle for 64 characters.
9 TXOFE	Transmit FIFO Overflow Interrupt Enable When this field is set, the TXOF flag generates an interrupt to the host. 0b - TXOF flag does not generate an interrupt to the host. 1b - TXOF flag generates an interrupt to the host.
8 RXUFE	Receive FIFO Underflow Interrupt Enable When this field is set, the RXUF flag generates an interrupt to the host. 0b - RXUF flag does not generate an interrupt to the host. 1b - RXUF flag generates an interrupt to the host.
7 TXFE	Transmit FIFO Enable When this field is set, the built-in FIFO structure for the transmit buffer is enabled. The size of the FIFO structure is indicated by TXFIFOSIZE. If this field is not set, the transmit buffer operates as a FIFO of depth one dataword regardless of the value in TXFIFOSIZE. Both CTRL[TE] and CTRL[RE] must be cleared prior to changing this field. 0b - Transmit FIFO is not enabled. Buffer depth is 1. 1b - Transmit FIFO is enabled. Buffer depth is indicated by TXFIFOSIZE.
6-4 TXFIFOSIZE	Transmit FIFO Buffer Depth The maximum number of transmit datawords that can be stored in the transmit buffer. This field is read-only. 000b - Transmit FIFO/Buffer depth = 1 dataword. 001b - Transmit FIFO/Buffer depth = 4 datawords. 010b - Transmit FIFO/Buffer depth = 8 datawords. 011b - Transmit FIFO/Buffer depth = 16 datawords.

Table continues on the next page...

Field	Function
	100b - Transmit FIFO/Buffer depth = 32 datawords. 101b - Transmit FIFO/Buffer depth = 64 datawords. 110b - Transmit FIFO/Buffer depth = 128 datawords. 111b - Transmit FIFO/Buffer depth = 256 datawords
3 RXFE	Receive FIFO Enable When this field is set, the built-in FIFO structure for the receive buffer is enabled. The size of the FIFO structure is indicated by the RXFIFOSIZE field. If this field is not set, the receive buffer operates as a FIFO of depth one dataword regardless of the value in RXFIFOSIZE. Both CTRL[TE] and CTRL[RE] must be cleared prior to changing this field. 0b - Receive FIFO is not enabled. Buffer depth is 1. 1b - Receive FIFO is enabled. Buffer depth is indicted by RXFIFOSIZE.
2-0 RXFIFOSIZE	Receive FIFO Buffer Depth The maximum number of receive datawords that can be stored in the receive buffer before an overrun occurs. This field is read-only. 000b - Receive FIFO/Buffer depth = 1 dataword. 001b - Receive FIFO/Buffer depth = 4 datawords. 010b - Receive FIFO/Buffer depth = 8 datawords. 011b - Receive FIFO/Buffer depth = 16 datawords. 100b - Receive FIFO/Buffer depth = 32 datawords. 101b - Receive FIFO/Buffer depth = 64 datawords. 110b - Receive FIFO/Buffer depth = 128 datawords. 111b - Receive FIFO/Buffer depth = 256 datawords.

49.6.1.13 LPUART Watermark Register (WATER)

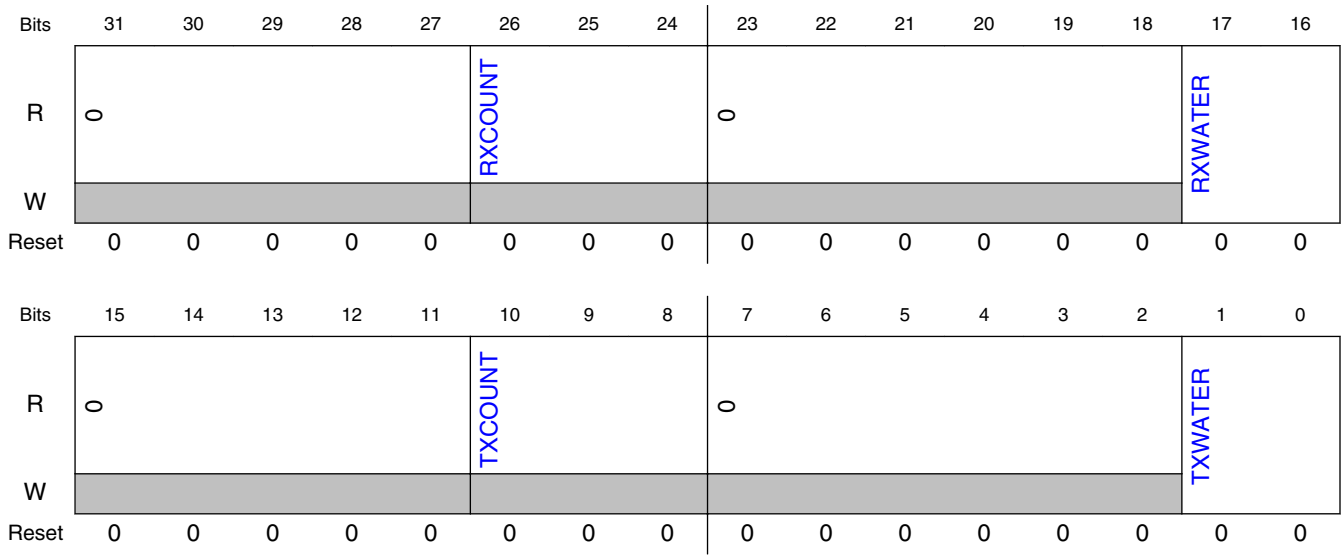
49.6.1.13.1 Offset

Register	Offset
WATER	2Ch

49.6.1.13.2 Function

This register provides the ability to set a programmable threshold for notification of needing additional transmit data. This register may be read at any time but must be written only when CTRL[TE] is not set.

49.6.1.13.3 Diagram



49.6.1.13.4 Fields

Field	Function
31-27 —	Reserved
26-24 RXCOUNT	Receive Counter The value in this register indicates the number of datawords that are in the receive FIFO/buffer. If a dataword is being received, that is, in the receive shift register, it is not included in the count. This value may be used in conjunction with FIFO[RXFIFOSIZE] to calculate how much room is left in the receive FIFO/buffer.
23-18 —	Reserved
17-16 RXWATER	Receive Watermark When the number of datawords in the receive FIFO/buffer is greater than the value in this register field, an interrupt or a DMA request is generated. For proper operation, the value in RXWATER must be set to be less than the receive FIFO/buffer size as indicated by FIFO[RXFIFOSIZE] and FIFO[RXFE].
15-11 —	Reserved
10-8 TXCOUNT	Transmit Counter The value in this register indicates the number of datawords that are in the transmit FIFO/buffer. If a dataword is being transmitted, that is, in the transmit shift register, it is not included in the count. This value may be used in conjunction with FIFO[TXFIFOSIZE] to calculate how much room is left in the transmit FIFO/buffer.
7-2 —	Reserved
1-0	Transmit Watermark

Field	Function
TXWATER	When the number of datawords in the transmit FIFO/buffer is equal to or less than the value in this register field, an interrupt or a DMA request is generated. For proper operation, the value in TXWATER must be set to be less than the size of the transmit buffer/FIFO size as indicated by FIFO[TXFIFOSIZE] and FIFO[TXFE].

Chapter 50
Flexible I/O (FlexIO)

50.1 Chip-specific FlexIO information

Table 50-1. Reference links to related information

Topic	Related module(s)	Reference
System memory map	—	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
Trigger mapping	-	Table 4-8

NOTE

In this device, FLEXIO1 and FLEXIO2/3 parameters are not exactly the same. FLEXIO1 has 16 pins, while FLEXIO2/3 has 32 pins.

NOTE

FlexIO cannot shift and store on the same cycle. For example, when trying to use FlexIO emulate the SSI slave device, will find it fail to shift and store on the same cycle, and the last falling edge of the clock is not used to latch the last data bit. One workaround is to use other timer count the number bit periods and generate the disable signals. However, it has two limitations: 1) it needs to know the transmitting rate, and set this timer with the same baud rate. 2) it requires the transmitting is asynchronous, does not have any clock stretch; otherwise, it will result in fail. So the slave receiving is not synchronous.

50.2 Overview

The FlexIO is a highly configurable module providing a wide range of functionality including:

- Emulation of a variety of serial/parallel communication protocols
- Flexible 16-bit timers with support for a variety of trigger, reset, enable and disable conditions
- Programmable logic blocks allowing the implementation of digital logic functions on-chip and configurable interaction of internal and external modules
- Programmable state machine for offloading basic system control functions from CPU

50.2.1 Block Diagram

The Figure 50-1 gives a high-level overview of the configuration of FlexIO timers and shifters.

The FlexIO uses shifters, timers and external triggers to shift data into or out of the FlexIO. The timing of the data shift is controlled by the timers, as shown in the block diagram. The timers can be configured to use generic timer functions, external triggers, or various other conditions to determine this control.

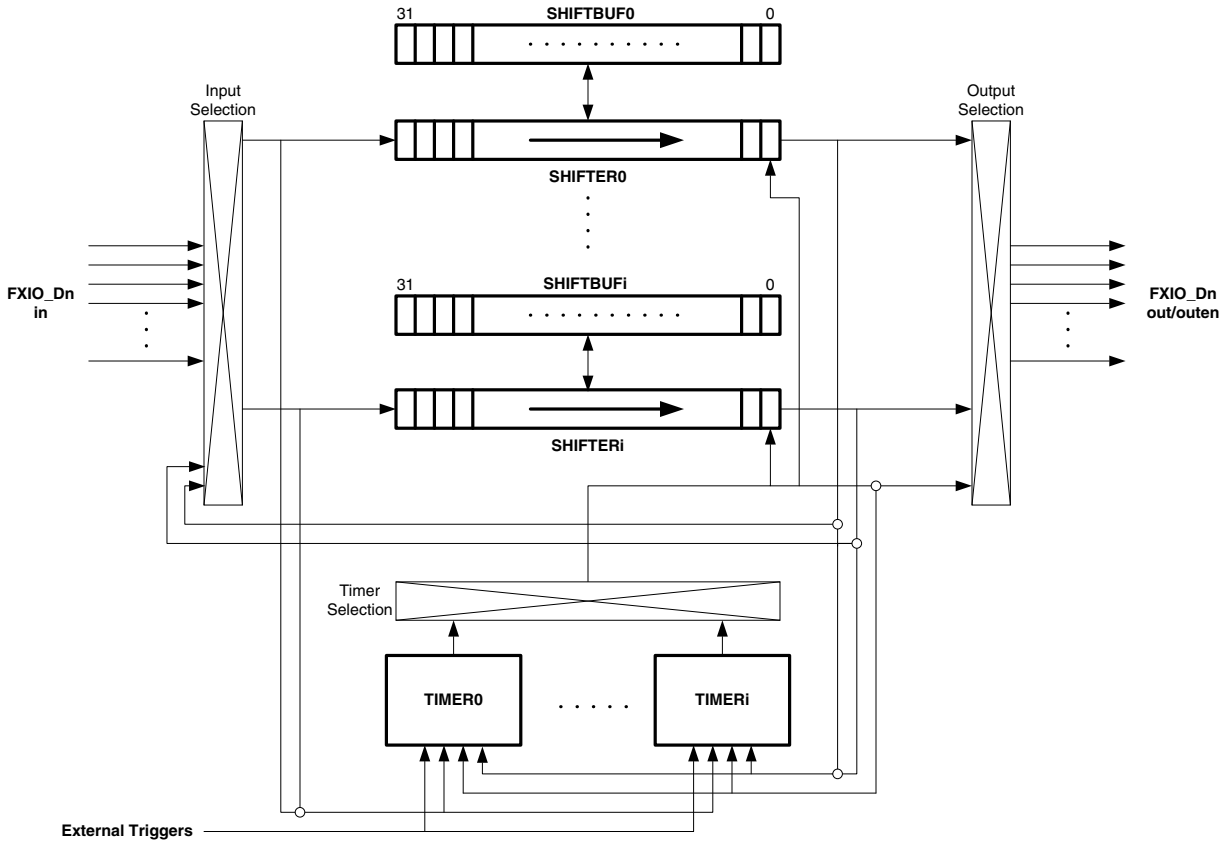


Figure 50-1. FlexIO block diagram

50.2.2 Features

The FlexIO module is capable of supporting a wide range of protocols including, but not limited to:

- UART
- I2C
- SPI
- I2S
- Camera IF
- Motorola 68K/Intel 8080 bus
- PWM/Waveform generation

The following key features are provided:

- Array of 32-bit shift registers with transmit, receive, data match, logic mode, and state modes
- Double buffered shifter operation for continuous data transfer
- Shifter concatenation to support large transfer sizes

- Automatic start/stop bit generation
- 1, 2, 4, 8, 16 or 32 multi-bit shift widths for parallel interface support
- Interrupt, DMA or polled transmit/receive operation
- Programmable baud rates independent of bus clock frequency, with support for asynchronous operation during stop modes
- Highly flexible 16-bit timers with support for a variety of internal or external trigger, reset, enable and disable conditions
- Programmable logic mode for integrating external digital logic functions on-chip or combining pin/shifter/timer functions to generate complex outputs
- Programmable state machine for offloading basic system control functions from CPU with support for up to 8 states, 8 outputs and 3 selectable inputs per state

50.3 Functional description

50.3.1 Shifter operation

Shifters are responsible for buffering and shifting data into or out of the FlexIO. The timing of shift, load and store events are controlled by the Timer assigned to the Shifter via the SHIFTCTL[TIMSEL] register. The Shifters are designed to support either DMA, interrupt or polled operation. The following block diagram provides a detailed view of the Shifter microarchitecture.

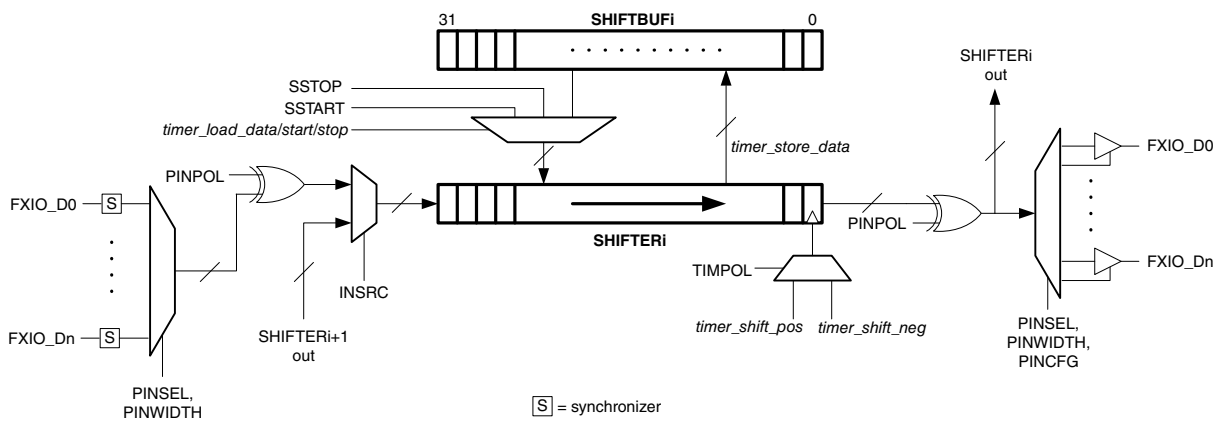


Figure 50-2. Shifter Microarchitecture

50.3.1.1 Transmit Mode

When configured for Transmit mode (SHIFTCTL[SMOD] =Transmit), the shifter loads data from the SHIFTBUF register and shift data out when a load event is signalled by the assigned Timer. An optional start/stop bit can also be automatically loaded before/ afterSHIFTBUF data by configuring the SHIFTCFG[SSTART], TIMCFG[TSTART] or SHIFTCFG[SSTOP], TIMCFG[TSTOP] registers in the Shifter and Timer.

NOTE

The shifter immediately loads a stop bit when the Shifter is initially configured for Transmit mode if a stop bit is enabled.

The Shifter Status Flag (SHIFTSTAT[SSF]) and any enabled interrupts or DMA requests are set when data has been loaded from the SHIFTBUF register into the Shifter or when the Shifter is initially configured into Transmit mode. The flag clears when new data has been written into the SHIFTBUF register or a logic 1 is written to this flag. In transmit mode, any write of the SHIFTBUF register clears the corresponding Shifter Status Flag. It does not matter what is writing or the state of the DMA/interrupt enables, the flag is cleared. How the flag is set/cleared for each mode is documented in the SSF register description.

The Shifter Error Flag (SHIFTERR[SEF]) and any enabled interrupts are set when an attempt to load data from an empty SHIFTBUF register occurs (buffer underrun). The flag can be cleared by writing it with logic 1.

50.3.1.2 Receive Mode

When configured for Receive mode (SHIFTCTL[SMOD] =Receive), the shifter shifts data in and store data into the SHIFTBUF register when a store event is signalled by the assigned Timer. Checking for a start/stop bit can be enabled before/after shifter data is sampled by configuring the SHIFTCFG[SSTART], TIMCFG[TSTART] or SHIFTCFG[SSTOP], TIMCFG[TSTOP] registers in the Shifter and Timer.

The Shifter Status Flag (SHIFTSTAT[SSF]) and any enabled interrupts or DMA requests are set when data has been stored into the SHIFTBUF register from the Shifter. The flag clears when the data has been read from theSHIFTBUF register or a logic 1 is written to this flag. Any read of the SHIFTBUF register clears the corresponding Shifter Status Flag when Shifter is configured in Receive mode. It does not matter what is reading or the state of the DMA/interrupt enables, the flag is cleared. How the flag is set/clear for each mode is documented in the SSF register description.

The Shifter Error Flag (SHIFTERR[SEF]) and any enabled interrupts are set when an attempt to store data into a fullSHIFTBUF register occurs (buffer overrun) or when a mismatch occurs on a start/stop bit check. The flag can be cleared by writing it with logic 1.

50.3.1.3 Match Store Mode

When configured for Match Store mode (SHIFTCTL[SMOD] =Match Store), the shifter shifts data in, check for a match result and store matched data into the SHIFTBUF register when a store event is signalled by the assigned Timer. Checking for a start/stop bit can be enabled before/after shifter data is sampled by configuring the SHIFTCFG[SSTART], TIMCFG[TSTART] or SHIFTCFG[SSTOP], TIMCFG[TSTOP] registers in the Shifter and Timer. Up to 16-bits of data can be compared using SHIFTBUF[31:16] to configure the data to be matched and SHIFTBUF[15:0] to mask the match result.

The Shifter Status Flag (SHIFTSTAT[SSF]) and any enabled interrupts or DMA requests are set when a match occurs and matched data has been stored into the SHIFTBUF register from the Shifter. The flag clears when the matched data has been read from the SHIFTBUF register or a logic 1 is written to this flag. Any read of the SHIFTBUF register clears the corresponding Shifter Status Flag when Shifter is configured in Match Store mode. It does not matter what is reading or the state of the DMA/interrupt enables, the flag is cleared. How the flag is set/clear for each mode is documented in the SSF register description.

The Shifter Error Flag (SHIFTERR[SEF]) and any enabled interrupts are set when an attempt to store matched data into a full SHIFTBUF register occurs (buffer overrun) or when a mismatch occurs on a start/stop bit check. The flag can be cleared by writing it with logic 1.

50.3.1.4 Match Continuous Mode

When configured for Match Continuous mode (SHIFTCTL[SMOD] =Match Continuous), the shifter shifts data in and continuously check for a match result whenever a shift event is signalled by the assigned Timer. Up to 16-bits of data can be compared using SHIFTBUF[31:16] to configure the data to be matched and SHIFTBUF[15:0] to mask the match result.

The Shifter Status Flag (SHIFTSTAT[SSF]) and any enabled interrupts or DMA requests are set when a match occurs. The flag clears automatically as soon as there is no longer a match between Shifter data and SHIFTBUF register. **The flag cannot be cleared by reading the SHIFTBUF register.**

The Shifter Error Flag (SHIFTERR[SEF]) and any enabled interrupts are set when a match occurs. The flag clears when there is a read from the SHIFTBUF register or it written with logic 1.

50.3.1.5 State Mode

Using State mode enables the user to implement any state machine with up to 8 states, 8 outputs and 3 selectable inputs per state. This feature allows basic control functions to be offloaded from the CPU, which could potentially remain in a low power mode.

When configured for State mode (SHIFTCTL[SMOD] =State), the SHIFTBUF register is used to drive the output and compute next state values when the Shifter has been selected by the current state pointer (SHIFTSTATE[STATE]). The following diagram provides a detailed view of Shifter microarchitecture when configured for State mode.

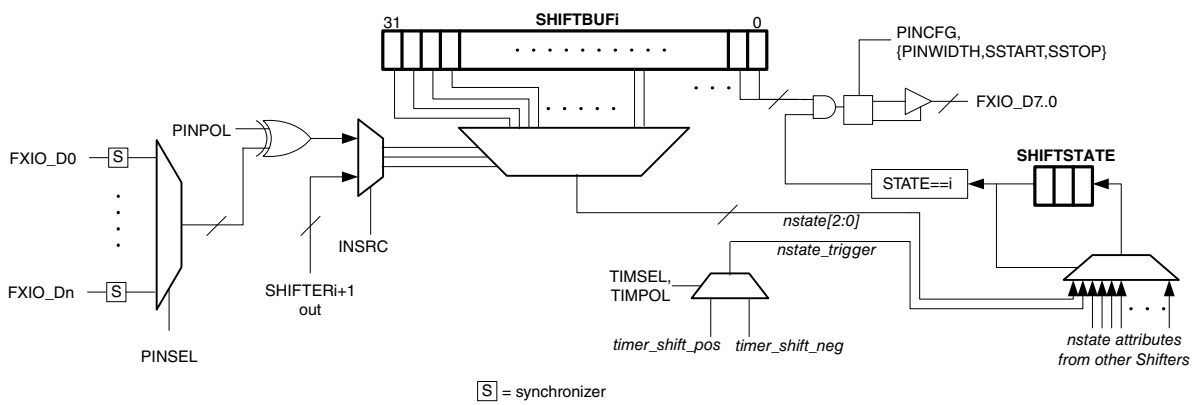


Figure 50-3. State Microarchitecture

When Shifter i has been selected by the current state pointer, output pins FXIO_D[7:0] are driven by SHIFTBUFi[31:24] using the configuration set by SHIFTCTLi[PINCFG]. When set, SHIFTCFGi {PWIDTH[3:0],SSTOP[1:0],SSTART[1:0]} are respectively used to disable the output drive on pins FXIO_D[7:0] for state machine applications which require less than 8 output pins.

The next state value is computed using the 3 input pins selected by SHIFTCTLi[PINSEL] together with SHIFTBUFi[23:0].

NOTE

Each state could potentially use a different set of 3 input pins.

The following table details how the next state value is computed when the current state pointer is pointing to Shifter i.

Table 50-2. Next State computation for SHIFTSTATE[STATE]=i

FXIO_D[PINSEL+2]	FXIO_D[PINSEL+1]	FXIO_D[PINSEL]	Next State Value
0	0	0	SHIFTBUFi[2:0]
0	0	1	SHIFTBUFi[5:3]
0	1	0	SHIFTBUFi[8:6]
0	1	1	SHIFTBUFi[11:9]
...	...	...	...
1	1	1	SHIFTBUFi[23:21]

Note that other shifters/timers could potentially be configured to drive the input pins of a given state, allowing the user to create complex combinations of shifters/timers as desired e.g. the output of a Shifter configured for logic mode could potentially be used to drive a state machine input.

The next state transition is triggered using the Timer output selected by SHIFTCTLi[TIMSEL] with polarity controlled by SHIFTCTLi[TIMPOL]. Note that each state could potentially use a different Timer to trigger each next state transition, allowing a variety of internal/external trigger sources and clocking configurations to be used (see Timer section for more detail).

The current state pointer defaults to Shifter 0 at reset, however it can be written by the user to select a different Shifter for the initial state. If the current state pointer selects a Shifter which is not configured for State mode, then outputs are not driven and a next state transition is never triggered.

The Shifter Status Flag (SHIFTSTAT[SSF]) and any enabled interrupts or DMA requests are set whenever the Shifter has been selected by the current state pointer. The flag clears when the current state pointer is updated to a different Shifter.

50.3.1.6 Logic Mode

Using logic mode enables the user to implement a small amount of programmable digital logic within a FlexIO Shifter.

When configured for Logic mode (SHIFTCTL[SMOD] =Logic), theSHIFTBUF register is used to implement a 5-input, 32-bit programmable logic look-up table. The following diagram provides a detailed view of Shifter microarchitecture when configured for Logic mode.

The SHIFTBUFi configures the lookup table for the 4 pin inputs and the SHIFTERi can optionally be used to configure a feedback or delayed pin source as the 5th input to the look up table.

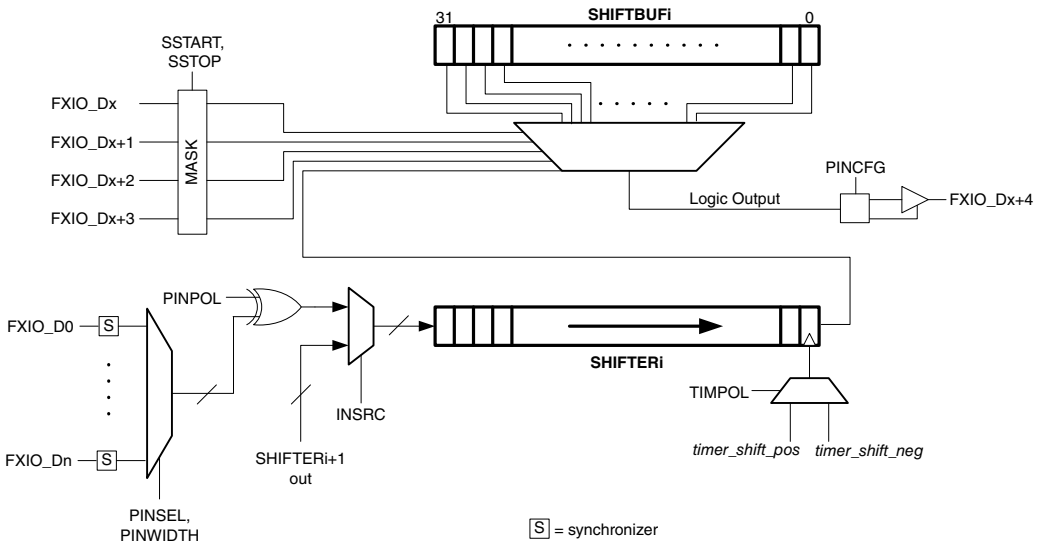


Figure 50-4. Logic Microarchitecture

The look-up table is driven using 4 pin inputs (maskable using SHIFTCFG[SSTOP] and SHIFTCFG[SSTART]) plus 1 input from the internal shifter and can be configured to drive an output pin using the SHIFTCTL[PINCFG] field. Pin inputs and outputs are fixed for each logic look-up table and are not selectable. The following table lists the logic output value selected by the look-up table for Shifter 'i'.

Table 50-3. Logic Look-up table for Shifter 'i'

SHIFTERi[0]	FXIO_D[x+3][1]	FXIO_D[x+2]	FXIO_D[x+1]	FXIO_D[x]	Logic Output to FXIO_D[x+4]
0	0	0	0	0	SHIFTBUFi[0]
0	0	0	0	1	SHIFTBUFi[1]
0	0	0	1	0	SHIFTBUFi[2]
0	0	0	1	1	SHIFTBUFi[3]
...	...	...	...	...	...
1	1	1	1	1	SHIFTBUFi[31]

1. for Shifter i=0...3, x=i

   for Shifter i=4...7, x=i+4

To minimize output glitches, SHIFTCFG[SSTOP] and SHIFTCFG[SSTART] can be used to mask unused input pins. When set, {SSTOP[1:0], SSTART[1:0]} mask FXIO_D[x+3]...FXIO_D[x] inputs respectively, so that any transitions on these pins does not cause the logic output to glitch.

Note that other shifters/timers could potentially be configured to drive the input pins of a given look-up table, allowing the user to concatenate look-up tables or create complex combinations of shifters/timers as desired.

SHIFTCFG[PWIDTH] controls the number of delay stages introduced by the internal shifter input (SHIFTERi[0]). For example, when configured for 1-bit shift (PWIDTH=0), the internal shifter introduces a 32 Shift clock delay before passing its input (selected by SHIFTCTL[PINSEL]) to the look-up table. When configured for 32-bit shift (PWIDTH=16...31), the internal shifter introduces a 1 Shift clock delay to its input.

The Shifter Status Flag (SHIFTSTAT[SSF]) and any enabled interrupts or DMA requests are set whenever the output pin allocated to the logic look-up table has a value of 1 (after being synchronized to the FlexIO clock). The flag clears when the output pin has a value of 0. This also allows the SSF flag to be used as a trigger to a Timer if desired.

The Shifter Error Flag (SHIFTERR[SEF]) and any enabled interrupts are set when the output pin allocated to the logic look-up table has asserted. The flag can be cleared by writing it with logic 1.

50.3.2 Timer Operation

The FlexIO 16-bit timers control the loading, shifting and storing of the shift registers, the counters load the contents of the compare register and decrement down to zero on the FlexIO clock. They can perform generic timer functions such as generating a clock or select output or a PWM waveform. Timers can be configured to enable in response to a trigger, pin or shifter condition; decrement always or only on a trigger or pin edge; reset in response to a trigger or pin condition; and disable on a trigger or pin condition or on a timer compare. Timers can optionally include a start condition and/or stop condition.

Each timer operates independently, although a timer can be configured to enable or disable at the same time as the previous timer (eg: timer1 can enable or disable at the same time as timer 0) and a timer output can be used to trigger any other timer. The trigger used by each timer is configured independently and can be configured to be a timer output, shifter status flag, pin input or an external trigger input (refer to the chip-specific FlexIO information for details on the external trigger connections). The trigger configuration is separate from the pin configuration, which can be configured for input, output data or output enable.

The Timer Configuration Register (TIMCFGn) should be configured before setting the Timer Mode (TIMOD).

50.3.2.1 Timer 8-bit Baud Counter Mode

In 8-bit Baud Counter Mode, the 16-bit counter is divided into two 8-bit counters. The lower 8-bits are used to configure the baud rate of the shift clock and the upper 8-bits are used to configure the number of shift clock edges in the transfer. When the lower 8-bits decrement to zero, the timer output is toggled and the lower 8-bits reload from the compare register. The upper 8-bits decrement when the lower 8-bits equal zero and decrement.

Note that a timer reset event in 8-bit Baud Counter Mode only resets the lower 8-bit counter, the upper 8-bit counter is not affected and can decrement if the timer reset is configured to update the state of the timer output, and the timer output toggles as a result of the timer reset event.

A timer compare event occurs when the upper 8-bits equal zero and decrements. The timer status flag is set on a timer compare event.

50.3.2.2 Timer 8-bit High PWM Mode

In 8-bit High PWM Mode, the 16-bit counter is divided into two 8-bit counters. The lower 8-bits are used to configure the timer output high period and the upper 8-bits are used to configure the timer output low period. The lower 8-bits decrement when the output is high. When the lower 8-bits equal zero and decrement, the timer output is cleared and the lower 8-bits are reloaded from the compare register. The upper 8-bits decrement when the output is low. When the upper 8-bits equal zero and decrement, the timer output is set and the upper 8-bits are reloaded from the compare register.

A timer compare event occurs when the upper 8-bits equal zero and decrements. The timer status flag is set on a timer compare event.

50.3.2.3 Timer 16-bit Counter Mode

In 16-bit Counter Mode, the 16-bit counter can be used to configure either the baud rate of the shift clock (eg: TIMDEC[1:0] != 10 or 11) or the number of shift clock edges in the transfer (eg: TIMDEC[1:0] = 10 or 11). When the 16-bit counter equals zero and decrements, the timer output toggles and the counter reloads from the compare register.

A timer compare event occurs when the 16-bit counter equals zero and decrements. The timer status flag is set on a timer compare event.

50.3.2.4 Timer Enable and Start Bit

When the TIMOD is configured for the desired mode, and the condition configured by timer enable (TIMENA) is detected then the following events occur.

- Timer counter loads the current value of the compare Register and start decrementing as configured by TIMDEC.
- Timer output may update to its initial state depending on the TIMOUT configuration. Shifters that are controlled by this timer do not see this as a rising edge on the timer shift clock.
- Transmit shifters controlled by this timer either output their start bit value, or load the shift register from the shift buffer and output the first bit, as configured by SSTART.

If the Timer start bit is enabled, the timer counter reloads with the compare register on the first rising edge of the shift clock after the timer starts decrementing. If there is no falling edge on the shift clock before the first rising edge (for example, when TIMOUT =1), a shifter that is configured to shift on falling edge and load on the first shift does not load correctly.

50.3.2.5 Timer Decrement and Reset

The Timer then generates the timer output and timer shift clock depending on the TIMOD and TIMDEC fields. The shifter clock is either equal to the timer output (when TIMDEC != 10 or 11) or equal to the decrement clock (when TIMDEC = 10 or 11). When TIMDEC is configured to decrement from a pin or trigger, the timer decrements on both rising and falling edges.

When the Timer is configured to reset as configured in the TIMRST field then the Timer counter loads the current value of the compare register again. The timer output and timer shift clock can be configured to update on timer reset, as configured by TIMOUT. This can result in a timer shift clock edge if the timer output toggles as a result of the timer reset. In 8-bit Baud Counter mode this would also decrement the upper 8-bits of the counter.

In general, when the timer counter decrements to zero a timer compare event is triggered. The timer compare event causes the timer counter to load the contents of the timer compare register, the timer output to toggle, any configured transmit shift registers to load, and any configured receive shift registers to store. Depending on the timer mode, the timer status flag may also be set.

50.3.2.6 Timer Disable and Stop Bit

When the is Timer is configured to add a stop bit on each compare, the following additional events occur.

- Transmit shifters controlled by this timer output their stop bit value (if configured by SSTOP).
- Receive shifters controlled by this timer store the contents of the shift register in their shift buffer, as configured by SSTOP.
- On the first rising edge of the shifter clock after the compare, the timer counter reloads the current value of the Compare Register.

Transmit shifters must be configured to load on the first shift when the timer is configured to insert a stop bit on each compare.

When the condition configured by timer disable (TIMDIS) is detected, the following events occur.

- Timer counter reloads the current value of the Compare Register and start decrementing as configured by TIMDEC.
- Timer output clears. Shifters that are controlled by this timer do not see this as a falling edge on the timer shift clock, but can generate a shift event if the timer shift clock would otherwise generate one.
- Transmit shifters controlled by this timer output their stop bit value (if configured by SSTOP).
- Receive shifters controlled by this timer store the contents of the shift register in their shift buffer, as configured by SSTOP.

If the timer stop bit is enabled, the timer counter continues decrementing until the next rising edge of the shift clock is detected, at which point it finishes. Although the timer output is forced low during the stop bit, the timer shift clock can toggle during the stop bit, but does not generate shift events.

A timer enable condition can be detected in the same cycle as a timer disable condition (if timer stop bit is disabled), or on the first rising edge of the shift clock after the disable condition (if stop bit is enabled). Receive shift registers with stop bit enabled store the contents of the shift register into the shift buffer and verify the state of the input data on

the configured shift edge while the timer is in the stop state condition. If there is no configured edge between the timer disable and the next rising edge of the shift clock then the final store and verify do not occur.

50.3.3 Pin operation

The pin configuration for each timer and shifter can be configured to use any FlexIO pin with either polarity. Each timer and shifter can be configured as an input, output data, output enable or bidirectional output. A pin configured for output enable can be used as an open drain (with inverted polarity, since the output enable assertion would cause logic zero to be output on the pin) or to control the enable on the bidirectional output. Any timer or shifter could be configured to control the output enable for a pin where the bidirectional output data is driven by another timer or shifter.

50.3.3.1 Parallel Interface

Shifters can be configured to use multiple FlexIO pins in parallel using the SHIFTCFG[PWIDTH] field. PWIDTH is used to configure the following settings of a shifter:

1. Number of bits shifted per Shift clock.
2. Number of pins driven by the shifter per Shift clock (only on shifters supporting parallel transmit i.e. SHIFTER0, SHIFTER4).
3. Number of pins sampled by the shifter per Shift clock (only on Shifter supporting parallel receive i.e. SHIFTER3, SHIFTER7).

When configured for parallel shift, either 4, 8, 16 or 32-bits can be shifted on every Shift clock. If an adjacent shifter is selected as the input source (SHIFTCFG[INSRC] =1), the least significant 4, 8, 16 or 32-bits from the adjacent shifter is sampled on each Shift clock.

For shifters supporting parallel receive (SHIFTER3, SHIFTER7), the shifter can be configured to sample multiple pins (SHIFTCFG[INSRC] =0), with PWIDTH and PINSEL selecting the pins as follows: FXIO_D[PINSEL+PWIDTH]:FXIO_D[PINSEL]. Note that if PWIDTH is less than the number of bits being shifted on each Shift clock, then the most significant bits are masked with 0 e.g. if PINSEL=7 and PWIDTH=6, then SHIFTER[31:24] samples {0,0,FXIO_D[12:7]} on each Shift clock.

For shifters supporting parallel transmit (SHIFTER0, SHIFTER4), the shifter can be configured to drive multiple pins using SHIFTCTL[PINCFG], with PWIDTH and PINSEL selecting the pins as follows: FXIO_D[PINSEL+PWIDTH]:FXIO_D[PINSEL].

Note that if PWIDTH is less than the number of bits being shifted on each Shift clock, then the most significant pins are not driven e.g. if PINSEL=7 and PWIDTH=6, then SHIFTER[5:0] drives only FXIO_D[12:7] on each Shift clock.

50.3.3.2 Pin Synchronization

When configuring a pin as an input (this includes a timer trigger configured as a pin input), the input signal is first synchronized to the FlexIO clock before the signal is used by a timer or shifter. This introduces a small latency of between 0.5 to 1.5 FlexIO clock cycles when using an external pin input to generate an output or control a shifter. This sets the maximum setup time at 1.5 FlexIO clock cycles.

If an input is used by more than one timer or shifter then the synchronization occurs once to ensure any edge is seen on the same cycle by all timers and shifters using that input.

Note that FlexIO pins are also connected internally, configuring a FlexIO shifter or timer to output data on an unused pin makes an internal connection that allows other shifters and timer to use this pin as an input. This allows a shifter output to be used to trigger a timer or a timer output to be shifted into a shifter. This path is also synchronized to the FlexIO clock and therefore incurs a 1 cycle latency.

So when using a Pin input as a Timer Trigger, Timer Clock or Shifter Data Input, the following synchronization delays occur:

1. 0.5 to 1.5 FlexIO clock cycles for external pin
2. 1 FlexIO clock cycle for an internally driven pin

For timing considerations such as output valid time and input setup time for specific applications (SPI Master, SPI Slave, I2C Master, I2S Master, I2S Slave) please refer to the FlexIO Application Information Section.

50.3.4 Low Power Modes

The FlexIO remains functional during low power modes, provided the Doze Enable bit (CTRL[DOZEN]) is clear and the FlexIO functional clock remains enabled.

The exception to this is in LLS mode. In this case, the Doze Enable (CTRL[DOZEN]) bit is ignored and the FlexIO waits for all Timers to complete any pending operation before acknowledging LLS mode entry.

50.3.5 Debug Mode

The FlexIO remains functional in debug mode provided the Debug Enable bit (CTRL[DBGE]) is set.

50.3.6 Clocks

50.3.6.1 Functional clock

The FlexIO functional clock is asynchronous to the bus clock and can remain enabled in low power modes. The FlexIO functional clock must be enabled before accessing any FlexIO registers. Provided the FlexIO functional clock is at least two times faster than the bus clock, the CTRL[FASTACC] bit can be set to support fast register accesses.

50.3.6.2 Bus clock

The bus clock is only used for bus accesses to the control and configuration registers.

50.3.7 Reset

50.3.7.1 Chip reset

The logic and registers for the FlexIO are reset to their default state on a chip reset.

50.3.7.2 Software reset

The FlexIO implements a software reset bit in its Control Register. The CTRL[SWRST] resets all logic and registers to their default state, except for the CTRL itself.

50.3.8 Interrupts and DMA Requests

The following table illustrates the status flags that can generate the FlexIO interrupt and DMA requests.

Table 50-4. FlexIO Interrupts and DMA Requests

Flag	Description	Interrupt	DMA Request	Low Power Wakeup
SSF	Shifter Status Flag.	Y	Y	Y
SEF	Shifter Error Flag.	Y	N	Y
TSF	Timer Status Flag.	Y	N	Y

50.3.9 Peripheral Triggers

The connection of the FlexIO peripheral triggers with other peripherals are device specific.

50.3.9.1 Output Triggers

Each FlexIO Timer generates an output trigger equal to the timer output. The output trigger is not affected by the timer pin polarity configuration.

50.3.9.2 Input Trigger

FlexIO supports multiple external trigger inputs that can be used to trigger one or more FlexIO timers. The external triggers are synchronized to the FlexIO functional clock and must assert for at least two cycles of the FlexIO functional clock to be sampled correctly.

50.4 Application Information

This section provides examples for a variety of FlexIO module applications.

50.4.1 UART Transmit

UART transmit can be supported using one Timer, one Shifter and one Pin (two Pins if supporting CTS). The start and stop bit insertion is handled automatically and multiple transfers can be supported using DMA controller. The timer status flag can be used to indicate when the stop bit of each word is transmitted.

Break and idle characters require software intervention, before transmitting a break or idle character the SSTART and SSTOP fields should be altered to transmit the required state and the data to transmit must equal 0xFF or 0x00. Supporting a second stop bit

requires the stop bit to be inserted into the data stream using software (and increasing the number of bits to transmit). Note that when performing byte writes to SHIFTBUFn (or SHIFTBUFBIS for transmitting MSB first), the rest of the register remains unaltered allowing an address mark bit or additional stop bit to remain undisturbed.

FlexIO does not support automatic insertion of parity bits.

Table 50-5. UART Transmit Configuration

Register	Value	Comments
SHIFTCFGn	0x0000_0032	Configure start bit of 0 and stop bit of 1.
SHIFTCTLn	0x0003_0002	Configure transmit using Timer 0 on posedge of clock with output data on Pin 0. Can invert output data by setting PINPOL, or can support open drain by setting PINPOL=0x1 and PINCFG =0x1.
TIMCMPn	0x0000_0F01	Configure 8-bit transfer with baud rate of divide by 4 of the FlexIO clock. Set TIMCMP[15:8] = (number of bits x 2) - 1. Set TIMCMP[7:0] = (baud rate divider / 2) - 1.
TIMCFGn	0x0000_2222	Configure start bit, stop bit, enable on trigger asserted and disable on compare. Can support CTS by configuring TIMEN=0x3.
TIMCTLn	0x01C0_0001	Configure dual 8-bit counter using Shifter 0 status flag as inverted internal trigger source. Can support CTS by configuring PINSEL=0x1 (for Pin 1) and PINPOL=0x1.
SHIFTBUFn	Data to transmit	Transmit data can be written to SHIFTBUF[7:0] to initiate an 8-bit transfer, use the Shifter Status Flag to indicate when data can be written using interrupt or DMA request. Can support MSB first transfer by writing to SHIFTBUFBBS[7:0] register instead.

An alternative configuration is possible that supports slower baud rates but requires two Timers.

Table 50-6. UART Transmit Configuration for Slow Baud Rate

Register	Value	Comments
SHIFTCFGn	0x0000_0032	Configure start bit of 0 and stop bit of 1.
SHIFTCTLn	0x0003_0002	Configure transmit using Timer 0 on posedge of clock with output data on Pin 0. Can invert output data by setting PINPOL, or can support open drain by setting PINPOL=0x1 and PINCFG =0x1.

Table continues on the next page...

Table 50-6. UART Transmit Configuration for Slow Baud Rate (continued)

Register	Value	Comments
TIMCMPn	0x0000_000F	Configure for 8-bit transfer. Set TIMCMP[15:0] = (number of bits x 2) - 1.
TIMCFGn	0x0030_2622	Configure start bit, stop bit, enable on trigger rising edge, decrement on trigger and disable on compare.
TIMCTLn	0x0740_0003	Configure 16-bit counter using Timer 1 output as internal trigger source.
TIMCMP(n+1)	0x0000_0001	Configure baud rate of divide by 4 of the FlexIO clock. Set TIMCMP[15:0] = (baud rate divider / 2) - 1.
TIMCFG(n+1)	0x0000_1200	Configure enable on trigger asserted and disable on Timer 0 disable. Can support CTS by configuring TIMEN=0x3.
TIMCTL(n+1)	0x01C0_0003	Configure 16-bit counter using Shifter 0 status flag as inverted internal trigger source. Can support CTS by configuring PINSEL =0x1 (for Pin 1) and PINPOL =0x1.
SHIFTBUFn	Data to transmit	Transmit data can be written to SHIFTBUF[7:0] to initiate an 8-bit transfer, use the Shifter Status Flag to indicate when data can be written using interrupt or DMA request. Can support MSB first transfer by writing to SHIFTBUFBBS[7:0] register instead.

50.4.2 UART Receive

UART receive can be supported using one Timer, one Shifter and one Pin (two Timers and two Pins if supporting RTS). The start and stop bit verification is handled automatically and multiple transfers can be supported using the DMA controller. The timer status flag can be used to indicate when the stop bit of each word is received.

Triple voting of the received data is not supported by FlexIO, data is sampled only once in the middle of each bit. Another timer can be used to implement a glitch filter on the incoming data, another Timer can also be used to detect an idle line of programmable length. Break characters cause the error flag to set and the shifter buffer register returns 0x00.

FlexIO does not support automatic verification of parity bits.

Table 50-7. UART Receiver Configuration

Register	Value	Comments
SHIFTCFGn	0x0000_0032	Configure start bit of 0 and stop bit of 1.
SHIFTCTLn	0x0080_0001	Configure receive using Timer 0 on negedge of clock with input data on Pin 0. Can invert input data by setting PINPOL.
TIMCMPn	0x0000_0F01	Configure 8-bit transfer with baud rate of divide by 4 of the FlexIO clock. Set TIMCMP[15:8] = (number of bits x 2) - 1. Set TIMCMP[7:0] = (baud rate divider / 2) - 1.
TIMCFGn	0x0204_2422	Configure start bit, stop bit, enable on pin posedge and disable on compare. Enable resynchronization to received data with TIMOUT=0x2 and TIMRST=0x4.
TIMCTLn	0x0000_0081	Configure dual 8-bit counter using inverted Pin 0 input.
SHIFTBUFn	Data to receive	Received data can be read from SHIFTBUFBYS[7:0], use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request. Can support MSB first transfer by reading from SHIFTBUFBIS[7:0] register instead.

The UART Receiver with RTS configuration uses a 2nd Timer to generate the RTS output. The RTS asserts when the start bit is detected and negate when the data is read from the shifter buffer register. No start bit is detected while the RTS is asserted, the received data is simply ignored.

Table 50-8. UART Receiver with RTS Configuration

Register	Value	Comments
SHIFTCFGn	0x0000_0032	Configure start bit of 0 and stop bit of 1.
SHIFTCTLn	0x0080_0001	Configure receive using Timer 0 on negedge of clock with input data on Pin 0. Can invert input data by setting PINPOL.
TIMCMPn	0x0000_0F01	Configure 8-bit transfer with baud rate of divide by 4 of the FlexIO clock. Set TIMCMP[15:8] = (number of bits x 2) - 1. Set TIMCMP[7:0] = (baud rate divider / 2) - 1.
TIMCFGn	0x0204_2522	Configure start bit, stop bit, enable on pin posedge with trigger asserted and disable on compare. Enable resynchronization to received data with TIMOUT=0x2 and TIMRST=0x4.

Table continues on the next page...

Table 50-8. UART Receiver with RTS Configuration (continued)

Register	Value	Comments
TIMCTLn	0x02C0_0081	Configure dual 8-bit counter using inverted Pin 0 input. Trigger is internal using inverted Pin 1 input.
TIMCMP(n+1)	0x0000_FFFF	Never compare.
TIMCFG(n+1)	0x0030_6100	Enable on Timer N enable and disable on trigger falling edge. Decrement on trigger to ensure no compare.
TIMCTL(n+1)	0x0143_0003	Configure 16-bit counter and output on Pin 1. Trigger is internal using Shifter 0 flag.
SHIFTBUFn	Data to receive	Received data can be read from SHIFTBUFBYS[7:0], use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request. Can support MSB first transfer by reading from SHIFTBUFBIS[7:0] register instead.

50.4.3 SPI Master

SPI master mode can be supported using two Timers, two Shifters and four Pins. Either CPHA=0 or CPHA=1 can be supported and transfers can be supported using the DMA controller. For CPHA=1, the select can remain asserted for multiple transfers and the timer status flag can be used to indicate the end of the transfer.

The stop bit is used to guarantee a minimum of 1 clock cycle between the slave select negating and before the next transfer. Writing to the transmit buffer by either core or DMA is used to initiate each transfer.

NOTE

Due to synchronization delays, the setup time for the serial input data is 1.5 FlexIO clock cycles, so the maximum baud rate is divide by 4 of the FlexIO clock frequency.

Table 50-9. SPI Master (CPHA=0) Configuration

Register	Value	Comments
SHIFTCFGn	0x0000_0000	Start and stop bit disabled.
SHIFTCTLn	0x0083_0002	Configure transmit using Timer 0 on negedge of clock with output data on Pin 0.
SHIFTCFG(n+1)	0x0000_0000	Start and stop bit disabled.

Table continues on the next page...

Table 50-9. SPI Master (CPHA=0) Configuration (continued)

Register	Value	Comments
SHIFTCTL(n+1)	0x0000_0101	Configure receive using Timer 0 on posedge of clock with input data on Pin 1.
TIMCMPn	0x0000_3F01	Configure 32-bit transfer with baud rate of divide by 4 of the FlexIO clock. Set TIMCMP[15:8] = (number of bits x 2) - 1. Set TIMCMP[7:0] = (baud rate divider / 2) - 1.
TIMCFGn	0x0100_2222	Configure start bit, stop bit, enable on trigger high and disable on compare, initial clock state is logic 0.
TIMCTLn	0x01C3_0201	Configure dual 8-bit counter using Pin 2 output (shift clock), with Shifter 0 flag as the inverted trigger. Set PINPOL to invert the output shift clock.
TIMCMP(n+1)	0x0000_FFFF	Never compare.
TIMCFG(n+1)	0x0000_1100	Enable when Timer 0 is enabled and disable when Timer 0 is disabled.
TIMCTL(n+1)	0x0003_0383	Configure 16-bit counter (never compare) using inverted Pin 3 output (as slave select).
SHIFTBUFn	Data to transmit	Transmit data can be written to SHIFTBUF, use the Shifter Status Flag to indicate when data can be written using interrupt or DMA request. Can support MSB first transfer by writing to SHIFTBUFBBS register instead.
SHIFTBUF(n+1)	Data to receive	Received data can be read from SHIFTBUFBYS, use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request. Can support MSB first transfer by reading from SHIFTBUFBIS register instead.

Table 50-10. SPI Master (CPHA=1) Configuration

Register	Value	Comments
SHIFTCFGn	0x0000_0021	Start bit loads data on first shift.
SHIFTCTLn	0x0003_0002	Configure transmit using Timer 0 on posedge of clock with output data on Pin 0.
SHIFTCFG(n+1)	0x0000_0000	Start and stop bit disabled.
SHIFTCTL(n+1)	0x0080_0101	Configure receive using Timer 0 on negedge of clock with input data on Pin 1.
TIMCMPn	0x0000_3F01	Configure 32-bit transfer with baud rate of divide by 4 of the FlexIO clock. Set TIMCMP[15:8] = (number of bits x 2) - 1.

Table continues on the next page...

Table 50-10. SPI Master (CPHA=1) Configuration (continued)

Register	Value	Comments
		Set TIMCMP[7:0] = (baud rate divider / 2) - 1.
TIMCFGn	0x0100_2222	Configure start bit, stop bit, enable on trigger high and disable on compare, initial clock state is logic 0.
TIMCTLn	0x01C3_0201	Configure dual 8-bit counter using Pin 2 output (shift clock), with Shifter 0 flag as the inverted trigger. Set PINPOL to invert the output shift clock. Set TIMDIS=3 to keep slave select asserted for as long as there is data in the transmit buffer.
TIMCMP(n+1)	0x0000_FFFF	Never compare.
TIMCFG(n+1)	0x0000_1100	Enable when Timer 0 is enabled and disable when Timer 0 is disabled.
TIMCTL(n+1)	0x0003_0383	Configure 16-bit counter (never compare) using inverted Pin 3 output (as slave select).
SHIFTBUFn	Data to transmit	Transmit data can be written to SHIFTBUF, use the Shifter Status Flag to indicate when data can be written using interrupt or DMA request. Can support MSB first transfer by writing to SHIFTBUFBBS register instead.
SHIFTBUF(n+1)	Data to receive	Received data can be read from SHIFTBUFBYS, use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request. Can support MSB first transfer by reading from SHIFTBUFBIS register instead.

50.4.4 SPI Slave

SPI slave mode can be supported using one Timer, two Shifters and four Pins. Either CPHA=0 or CPHA=1 can be supported and transfers can be supported using the DMA controller. For CPHA=1, the select can remain asserted for multiple transfers and the timer status flag can be used to indicate the end of the transfer.

The transmit data must be written to the transmit buffer register before the external slave select asserts, otherwise the shifter error flag is set.

NOTE

Due to synchronization delays, the output valid time for the serial output data is 2.5 FlexIO clock cycles, so the maximum baud rate is divide by 6 of the FlexIO clock frequency.

Table 50-11. SPI Slave (CPHA=0) Configuration

Register	Value	Comments
SHIFTCFGn	0x0000_0000	Start and stop bit disabled.
SHIFTCTLn	0x0083_0002	Configure transmit using Timer 0 on falling edge of shift clock with output data on Pin 0.
SHIFTCFG(n+1)	0x0000_0000	Start and stop bit disabled.
SHIFTCTL(n+1)	0x0000_0101	Configure receive using Timer 0 on rising edge of shift clock with input data on Pin 1.
TIMCMPn	0x0000_003F	Configure 32-bit transfer. Set TIMCMP[15:0] = (number of bits x 2) - 1.
TIMCFGn	0x0120_0600	Configure enable on trigger rising edge, initial clock state is logic 0 and decrement on pin input.
TIMCTLn	0x06C0_0203	Configure 16-bit counter using Pin 2 input (shift clock), with Pin 3 input (slave select) as the inverted trigger.
SHIFTBUFn	Data to transmit	Transmit data can be written to SHIFTBUF, use the Shifter Status Flag to indicate when data can be written using interrupt or DMA request. Can support MSB first transfer by writing to SHIFTBUFBBS register instead.
SHIFTBUF(n+1)	Data to receive	Received data can be read from SHIFTBUFBYS, use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request. Can support MSB first transfer by reading from SHIFTBUFBIS register instead.

Table 50-12. SPI Slave (CPHA=1) Configuration

Register	Value	Comments
SHIFTCFGn	0x0000_0001	Shifter configured to load on first shift and stop bit disabled.
SHIFTCTLn	0x0003_0002	Configure transmit using Timer 0 on rising edge of shift clock with output data on Pin 0.
SHIFTCFG(n+1)	0x0000_0000	Start and stop bit disabled.
SHIFTCTL(n+1)	0x0080_0101	Configure receive using Timer 0 on falling edge of shift clock with input data on Pin 1.
TIMCMPn	0x0000_003F	Configure 32-bit transfer. Set TIMCMP[15:0] = (number of bits x 2) - 1.
TIMCFGn	0x0120_6602	Configure start bit, enable on trigger rising edge, disable on trigger falling edge, initial clock state is logic 0 and decrement on pin input.

Table continues on the next page...

Table 50-12. SPI Slave (CPHA=1) Configuration (continued)

Register	Value	Comments
TIMCTLn	0x06C0_0203	Configure 16-bit counter using Pin 2 input (shift clock), with Pin 3 input (slave select) as the inverted trigger.
SHIFTBUFn	Data to transmit	Transmit data can be written to SHIFTBUF, use the Shifter Status Flag to indicate when data can be written using interrupt or DMA request. Can support MSB first transfer by writing to SHIFTBUFBBS register instead.
SHIFTBUF(n+1)	Data to receive	Received data can be read from SHIFTBUFBYS, use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request. Can support MSB first transfer by reading from SHIFTBUFBIS register instead.

50.4.5 I2C Master

I2C master mode can be supported using two Timers, two Shifters and two Pins. One timer is used to generate the SCL output and one timer is used to control the shifters. The two shifters are used to transmit and receive for every word, when receiving the transmitter must transmit 0xFF to tristate the output. FlexIO inserts a stop bit after every word to generate/verify the ACK/NACK. FlexIO waits for the first write to the transmit data buffer before enabling SCL generation. Data transfers can be supported using the DMA controller and the shifter error flag sets on transmit underrun or receive overflow.

The first timer generates the bit clock for the entire packet (START to Repeated START/ STOP), so the compare register needs to be programmed with the total number of clock edges in the packet (minus one). The timer supports clock stretching using the reset counter when pin equal to output (although this increases both the clock high and clock low periods by at least 1 FlexIO clock cycle each). The second timer uses the SCL input pin to control the transmit/receive shift registers, this enforces an SDA data hold time by an extra 2 FlexIO clock cycles.

Both the transmit and receive shifters need to be serviced for each word in the transfer, the transmit shifter must transmit 0xFF when receiving and the receive shifter returns the data actually present on the SDA pin. The transmit shifter loads 1 additional word on the last falling edge of SCL pin, this word should be 0x00 if generating a STOP condition or 0xFF if generating a repeated START condition. During the last word of a master-receiver transfer, the transmit SSTOP bit should be set by software to generate a NACK.

The receive shift register asserts an error interrupt if a NACK is detected, but software is responsible for generating the STOP or repeated START condition. If a NACK is detected during master-transmit, the interrupt routine should immediately write the transmit shifter register with 0x00 (if generating STOP) or 0xFF (if generating repeated START). Software should then wait for the next rising edge on SCL and then disable both timers. The transmit shifter should then be disabled after waiting the setup delay for a repeated START or STOP condition.

NOTE

Due to synchronization delays, the data valid time for the transmit output is 2 FlexIO clock cycles, so the maximum baud rate is divide by 6 of the FlexIO clock frequency.

The I2C master data valid is delayed 2 cycles because the clock output is passed through a synchronizer before clocking the transmit/receive shifter (to guarantee some SDA hold time). Since the SCL output is synchronous with FlexIO clock, the synchronization delay is 1 cycle and then 1 cycle to generate the output.

Table 50-13. I2C Master Configuration

Register	Value	Comments
SHIFTCFGn	0x0000_0032	Start bit enabled (logic 0) and stop bit enabled (logic 1).
SHIFTCTLn	0x0101_0082	Configure transmit using Timer 1 on rising edge of clock with inverted output enable (open drain output) on Pin 0.
SHIFTCFG(n+1)	0x0000_0020	Start bit disabled and stop bit enabled (logic 0) for ACK/NACK detection.
SHIFTCTL(n+1)	0x0180_0001	Configure receive using Timer 1 on falling edge of clock with input data on Pin 0.
TIMCMPn	0x0000_2501	Configure 2 word transfer with baud rate of divide by 4 of the FlexIO clock. Set TIMCMP[15:8] = (number of words x 18) + 1. Set TIMCMP[7:0] = (baud rate divider / 2) - 1.
TIMCFGn	0x0102_2222	Configure start bit, stop bit, enable on trigger high, disable on compare, reset if output equals pin. Initial clock state is logic 0 and is not affected by reset.
TIMCTLn	0x01C1_0101	Configure dual 8-bit counter using Pin 1 output enable (SCL open drain), with Shifter 0 flag as the inverted trigger.
TIMCMP(n+1)	0x0000_000F	Configure 8-bit transfer. Set TIMCMP[15:0] = (number of bits x 2) - 1.
TIMCFG(n+1)	0x0020_1112	Enable when Timer 0 is enabled, disable when Timer 0 is disabled, enable start bit and stop bit at end of each word, decrement on pin input.

Table continues on the next page...

Table 50-13. I2C Master Configuration (continued)

Register	Value	Comments
TIMCTL(n+1)	0x01C0_0183	Configure 16-bit counter using inverted Pin 1 input (SCL).
SHIFTBUFn	Data to transmit	Transmit data can be written to SHIFTBUFBBS[7:0], use the Shifter Status Flag to indicate when data can be written using interrupt or DMA request.
SHIFTBUF(n+1)	Data to receive	Received data can be read from SHIFTBUFBIS[7:0], use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request.

50.4.6 I2S Master

I2S master mode can be supported using two Timers, two Shifters and four Pins. One timer is used to generate the bit clock and control the shifters and one timer is used to generate the frame sync. FlexIO waits for the first write to the transmit data buffer before enabling bit clock and frame sync generation. Data transfers can be supported using the DMA controller and the shifter error flag sets on transmit underrun or receive overflow.

The bit clock frequency is an even integer divide of the FlexIO clock frequency, and the initial frame sync assertion occurs at the same time as the first bit clock edge. The timer uses the start bit to ensure the frame sync is generated one clock cycle before the first output data.

NOTE

Due to synchronization delays, the setup time for the receiver input is 1.5 FlexIO clock cycles, so the maximum baud rate is divide by 4 of the FlexIO clock frequency.

Table 50-14. I2S Master Configuration

Register	Value	Comments
SHIFTCFGn	0x0000_0001	Load transmit data on first shift and stop bit disabled.
SHIFTCTLn	0x0003_0002	Configure transmit using Timer 0 on rising edge of clock with output data on Pin 0.
SHIFTCFG(n+1)	0x0000_0000	Start and stop bit disabled.
SHIFTCTL(n+1)	0x0080_0101	Configure receive using Timer 0 on falling edge of clock with input data on Pin 1.
TIMCMPn	0x0000_3F01	Configure 32-bit transfer with baud rate of divide by 4 of the FlexIO clock. Set

Table continues on the next page...

Table 50-14. I2S Master Configuration (continued)

Register	Value	Comments
		TIMCMP[15:8] = (number of bits x 2) - 1. Set TIMCMP[7:0] = (baud rate divider / 2) - 1.
TIMCFGn	0x0000_0202	Configure start bit, enable on trigger high and never disable. Initial clock state is logic 1.
TIMCTLn	0x01C3_0281	Configure dual 8-bit counter using inverted Pin 2 output (bit clock), with Shifter 0 flag as the inverted trigger. Clear PINPOL to invert the polarity of the output shift clock.
TIMCMP(n+1)	0x0000_007F	Configure 32-bit transfer with baud rate of divide by 4 of the FlexIO clock. Set TIMCMP[15:0] = (number of bits x baud rate divider) - 1.
TIMCFG(n+1)	0x0000_0100	Enable when Timer 0 is enabled and never disable.
TIMCTL(n+1)	0x0003_0383	Configure 16-bit counter using inverted Pin 3 output (as frame sync). Clear PINPOL to invert the polarity of the output frame sync
SHIFTBUFn	Data to transmit	Transmit data can be written to SHIFTBUFBIS, use the Shifter Status Flag to indicate when data can be written using interrupt or DMA request. Can support LSB first transfer by writing to SHIFTBUF register instead.
SHIFTBUF(n+1)	Data to receive	Received data can be read from SHIFTBUFBIS, use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request. Can support LSB first transfer by reading from SHIFTBUF register instead.

50.4.7 I2S Slave

I2S slave mode can be supported using three Timers, two Shifters and four Pins (for single transmit and single receive, other combinations of transmit and receive are possible).

The transmit data must be written to the transmit buffer register before the external frame sync asserts, otherwise the shifter error flag is set.

NOTE

Due to synchronization delays, the output valid time for the serial output data is 2.5 FlexIO clock cycles, so the maximum baud rate is divide by 6 of the FlexIO clock frequency.

The output valid time of I2S slave is max 2.5 cycles because there is a maximum 1.5 cycle delay on the clock synchronization plus 1 cycle to output the data

Timer 2 detects the falling edge of frame sync (start of new frame) and asserts output until rising edge of bit clock (when frame sync is normally sampled). Timer 0 detects rising edge of bit clock with Timer 2 output asserted and asserts output for length of frame. Timer 1 detects falling edge of bit clock with Timer 0 output asserted and controls shift registers for 32-bit transfers.

Table 50-15. I2S Slave Configuration

Register	Value	Comments
SHIFTCFGn	0x0000_0000	Start and stop bit disabled.
SHIFTCTLn	0x0103_0002	Configure transmit using Timer 1 on rising edge of shift clock with output data on Pin 0.
SHIFTCFG(n+1)	0x0000_0000	Start and stop bit disabled.
SHIFTCTL(n+1)	0x0180_0101	Configure receive using Timer 1 on falling edge of shift clock with input data on Pin 1.
TIMCMPn	0x0000_007F	Configure two 32-bit transfers per frame. Set TIMCMP[15:0] = (number of bits x 4) - 1.
TIMCFGn	0x0020_2500	Configure enable on pin rising edge (inverted bit clock) with trigger high (Timer 2) and disable on compare, initial clock state is logic 1 and decrement on pin input (bit clock).
TIMCTLn	0x0B40_0203	Configure 16-bit counter using Pin 2 input (bit clock), with Timer 2 output as the trigger.
TIMCMP(n+1)	0x0000_003F	Configure 32-bit transfers. Set TIMCMP[15:0] = (number of bits x 2) - 1.
TIMCFG(n+1)	0x0020_2500	Configure enable on pin (bit clock) rising edge with trigger (Timer 0) high and disable on compare, initial clock state is logic 1 and decrement on pin input (bit clock).
TIMCTL(n+1)	0x0340_0283	Configure 16-bit counter using inverted Pin 2 input (bit clock), with Timer 0 output as the trigger.
TIMCMP(n+2)	0x0000_0000	Compare on zero (first edge).
TIMCFG(n+2)	0x0020_6400	Configure enable on inverted pin (frame sync) rising edge and disable on trigger

Table continues on the next page...

Table 50-15. I2S Slave Configuration (continued)

Register	Value	Comments
		falling edge (bit clock), initial clock state is logic 1 and decrement on inverted pin input (frame sync).
TIMCTL(n+2)	0x04C0_0383	Configure 16-bit counter using inverted Pin 3 input (frame sync), with Pin 2 inverted input (bit clock) as the trigger.
SHIFTBUFn	Data to transmit	Transmit data can be written to SHIFTBUFBIS, use the Shifter Status Flag to indicate when data can be written using interrupt or DMA request. Can support LSB first transfer by writing to SHIFTBUF register instead.
SHIFTBUF(n+1)	Data to receive	Received data can be read from SHIFTBUFBIS, use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request. Can support LSB first transfer by reading from SHIFTBUF register instead.

50.4.8 Camera Interface

Camera Interface can be supported using one Timer, one or more Shifters and multiple Pins. Multiple transfers can be supported using DMA controller.

The example below describes FlexIO configuration for interfacing to an 8-bit CMOS sensor with PCLK, VSYNC, HREF and D[7:0] outputs. The example uses a 128-bit buffer to capture 16-pixels of image data before interrupt or DMA transfer, however a bigger or smaller buffer may be used depending on system DMA performance and FlexIO resource usage by other applications. Note that additional timers may be used to track number of pixels per row and number of rows per frame or HREF/VSYNC may be assigned as GPIO interrupts for software tracking.

Table 50-16. Camera Interface Configuration for 8-bit CMOS sensor

Register	Value	Comments
SHIFTCFGn...n+2[1]	0x0007_0100	Configure 8-bit parallel shift in from adjacent shifter.
SHIFTCFGn+3	0x0007_0000	Configure 8-bit parallel shift in from pins FXIO_D[7:0] (D[7:0]).
SHIFTCTLn...n+3	0x0080_0001	Configure receive using Timer 0 on negedge of clock.
TIMCMPn	0x0000_001F	Configure 16-pixel (8 bits/pixel x 16 pixels = 128-bits) transfer. Set

Table continues on the next page...

Table 50-16. Camera Interface Configuration for 8-bit CMOS sensor (continued)

Register	Value	Comments
		TIMCMP[15:0] = (number of pixels x 2) - 1.
TIMCFGn	0x0120_6600	Configure enable on trigger (HREF) rising edge and disable on trigger falling edge, initial Shift clock state is logic 0 and decrement on PCLK input.
TIMCTLn	0x12C0_0803	Configure 16-bit counter using FXIO_D[8] input (PCLK), with FXIO_D[9] input (HREF) as the inverted trigger.
SHIFTBUFn...n+3	Data to receive	Received data can be read from SHIFTBUFn...n+3, use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request.

1. n=0 or 4

50.4.9 Motorola 68K/Intel 8080 Bus Interface

The Motorola 68K and Intel 8080 bus are two parallel interfaces commonly used by Smart/Asynchronous LCD controllers. In conjunction with GPIO, FlexIO is able to drive these interfaces using one Timer and one Shifter, although additional Shifters could be used to support large transfers via the DMA controller.

The configuration below provides an example of how to drive a 16-bit 68K or 8080 bus. For a 8080 bus, two GPIO are used to drive the nCS and RS pins. For a 68K bus, an additional GPIO is required to drive the RDWR pin.

Table 50-17. Motorola 68K/Intel 8080 Write Configuration

Register	Value	Comments
SHIFTCFG0...7	0x000F_0100	Configure 16-bit parallel shift in from adjacent shifter.
SHIFTCTL0	0x0003_0002	Configure transmit using Timer 0 on posedge of clock with data output to FXIO_D[15:0].
SHIFTCTL1...7	0x0000_0002	Configure transmit using Timer 0 on posedge of clock.
TIMCMP0	0x0000_0101 (1-beat) 0x0000_1F01 (16-beats)	Configure 1 or 16-beat transfer with baud rate of divide by 4 of the FlexIO clock. Set TIMCMP[15:8] = (number of beats x 2) - 1. Set TIMCMP[7:0] = (baud rate divider / 2) - 1.
TIMCFG0	0x0000_2200	Configure enable on trigger high and disable on compare. Timer output high on enable.

Table continues on the next page...

Table 50-17. Motorola 68K/Intel 8080 Write Configuration (continued)

Register	Value	Comments
TIMCTL0	0x01C3_1001 (Motorola 68K, 1-beat) 0x1DC3_1001 (Motorola 68K, 16-beats) 0x01C3_1081 (Intel 8080, 1-beat) 0x1DC3_1081 (Intel 8080, 16-beats)	Configure dual 8-bit counter using FXIO_D[16] output (EN pin for 68K, nWR pin for 8080), with Shifter 0 (1-beat) or Shifter 7 (16-beats) flag as the inverted trigger.
SHIFTBUF0...7	Data to transmit	Transmit data can be written to SHIFTBUF0 (1-beat) or SHIFTBUF0...7 (16-beats) to initiate a transfer, use the Shifter Status Flag to indicate when data can be written using interrupt or DMA request.

Table 50-18. Motorola 68K/Intel 8080 Read Configuration

Register	Value	Comments
SHIFTCFG0...6	0x000F_0100	Configure 16-bit parallel shift in from adjacent shifter.
SHIFTCFG7	0x000F_0000	Configure 16-bit parallel shift in from pin.
SHIFTCTL0...7	0x0080_0001	Configure receive using Timer 0 on negedge of clock with data input from FXIO_D[15:0].
TIMCMP0	0x0000_0101 (1-beat) 0x0000_1F01 (16-beats)	Configure 1 or 16-beat transfer with baud rate of divide by 4 of the FlexIO clock. Set TIMCMP[15:8] = (number of beats x 2) - 1. Set TIMCMP[7:0] = (baud rate divider / 2) - 1.
TIMCFG0	0x0000_2220	Configure stop_bit, enable on trigger high and disable on compare. Timer output high on enable.
TIMCTL0	0x1DC3_1001 (Motorola 68K, 1 beat) 0x01C3_1001 (Motorola 68K, 16 beats) 0x1DC3_1181 (Intel 8080, 1 beat) 0x01C3_1181 (Intel 8080, 16 beats)	Configure dual 8-bit counter using either FXIO_D[16] output (EN pin for 68K) or FXIO_D[17] output (nRD pin for 8080), with Shifter 7 flag (1-beat) or Shifter 0 flag (16-beats) as the inverted trigger.
SHIFTBUF0...7	Data received	Received data can be read from SHIFTBUF0 (1-beat) or SHIFTBUF0...7 (16-beats), use the Shifter Status Flag to indicate when data can be read using interrupt or DMA request.

In general, any operation to a 68K/8080 bus slave begins with a register write cycle followed by one or more data read or write cycles. To accomplish this, the following program flow should be used:

1. Configure FlexIO with 1-beat write configuration
2. Configure GPIO to assert nCS, RS pins (and deassert RDWR pin for 68K)
3. Write register index data to SHIFTBUF0[15:0]

4. Configure GPIO to deassert RS pin (and assert RDWR pin for 68K data read)
5. Configure FlexIO with desired read or write configuration (e.g. 1 or 16-beats)
6. Use the Shifter Status Flag to trigger interrupt or DMA driven data transfers to/from SHIFTBUF registers
7. Configure GPIO to deassert nCS pin

50.4.10 Low Power State Machine

The configuration below details a hypothetical state machine example to illustrate the flexibility allowed when using Shifter state mode.

In this example, FlexIO waits for the FXIO_D[2] pin to assert and then drives a complementary square wave output at a frequency of FLEXIO_CLK/131072 on the FXIO_D[1:0] pins while the comparator output is asserted (This assumes comparator is connected to external trigger 15. See the chip-specific FlexIO information for actual FlexIO trigger mappings). Throughout this operation, the CPU can be kept in a STOP/ VLPS mode by clearing the CTRL[DOZEN] bit and ensuring the FLEXIO_CLK is enabled. The state diagram below shows the states and transitions implemented by this example.

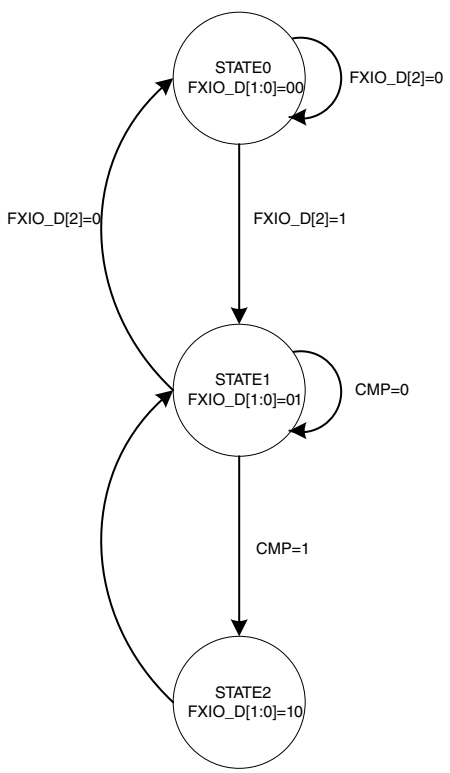


Figure 50-5. State Diagram

Table 50-19. State Machine Configuration

Register	Value	Comments
SHIFTCFG0...2	0x0000_0003	Enable FXIO_D[1:0] as outputs.
SHIFTCTL0	0x0080_0206	Configure for State mode using FXIO_D[4:2] as inputs to select next state and Timer0 output low to trigger next state.
SHIFTBUF0	0x0020_8208	State0: Drive FXIO_D[1:0]=00, transition to State0 if FXIO_D[2]=0, State1 if FXIO_D[2]=1.
SHIFTCTL1	0x0000_0206	Configure for State mode using FXIO_D[4:2] as inputs to select next state and Timer0 output high to trigger next state.
SHIFTBUF1	0x0140_8408	State1: Drive FXIO_D[1:0]=01, transition to State0 if FXIO_D[2]=0, State1 if CMP=0, State2 if CMP=1 (FXIO_D[3]=1)
SHIFTCTL2	0x0080_0206	Configure for State mode using FXIO_D[4:2] as inputs to select next state and Timer0 output low to trigger next state.
SHIFTBUF2	0x0224_9249	State2: Drive FXIO_D[1:0]=10, transition to State1 when Timer0 output low
TIMCMP0	0x0000_FFFF	Configure baud rate of divide by 131072 of the FlexIO clock. Set TIMCMP[7:0] = (baud rate divider / 2) - 1.
TIMCFG0	0x0000_0000	Configure timer always enabled.
TIMCTL0	0x0000_0003	Configure single 16-bit counter.
TIMCFG1	0x0010_7600	Configure timer enabled on trigger rising edge, disabled on trigger falling edge, decrement on trigger edge.
TIMCTL1	0x0F03_0303	Configure timer output enabled on FXIO_D[3], external trigger 15 (comparator output) selected.

50.4.11 Keypad Interface

The Keypad Interface can support a 3x4 keypad matrix using 3 timers and 3 shifters, although a larger matrix could be supported using additional shifters. The configuration is designed for 4 columns which are configured as active low open drain pins, and 3 rows which are configured as input pins with pullup resistors enabled.

One shifter is configured in logic mode and is configured to assert its output when any of the row inputs are low, indicating a key is pressed. A timer is used to filter the shifter output to ensure the key is pressed for a minimum amount of time before performing the column scan.

Another shifter is configured for parallel transmit, this shifter is used to scan each column when a key press is detected. Whenever not scanning, the shifter output is configured to assert all active low open drain column outputs to detect any keypress. A dedicated timer is used to control the transmit shifter.

The last shifter is configured for parallel receive, this shifter is used to capture the result of the column scanning for software to decode which key or keys was pressed. This configuration captures the state of both row and colum pins for each scan, although the row state could also be deduced by the shift order. A dedicated timer is used to control the receive shifter, it shifts at half the frequency of the transmit shifter.

Once the result of the key scan is available in the receive shifter register, FlexIO continues to monitor the row inputs and can trigger multiple scans from the one key press. To support deboucing, software can decide how many consecutive scans should be considered a single key press.

Since the pins used in logic mode are fixed per shifter, and the shifters that support parallel shifts are limited, this configuration is restricted in what pins and shifters it can use. Increasing the matrix beyond 4 row inputs requires multiple shifters in logic mode and increasing beyond 4 column outputs requires concatenating transmit shifters to create a larger shift register.

Table 50-20. Keypad Interface Configuration

Register	Value	Comments
SHIFTCFG0	0x0003_0032	Start bit enabled (logic 0) and stop bit enabled (logic 1), configired for 4-bit shift.
SHIFTCTL0	0x0101_0402	Configure transmit using Timer 1 on rising edge of clock generating open drain output on Pins [7:4] (column outputs).
SHIFTBUF0	0x0804_0201	Static data containing the column scan pattern, each column is scanned one-hot with deadtime inbetween.
SHIFTCFG1	0x0000_0020	In logic mode, mask input 3.
SHIFTCTL1	0x0000_0007	Configure logic mode.
SHIFTBUF1	0x07ff_07ff	Static data configuring logic mode LUT. Output asserts if pins [3:1] are logic 0.
SHIFTCFG3	0x0007_0000	Configured for 8-bit shift.
SHIFTCTL3	0x0280_0001	Configure receive using Timer 2 on falling edge of clock with input data on Pins [7:0] (both rows and columns, pin 0 is don't care).
TIMCMP0	0x0000_00ff	Configures pre-scanning glitch filter to 256 FlexIO clock cycles. For different filter cycles, set TIMCMP[15:0] = (filter cycles) - 1.

Table continues on the next page...

Table 50-20. Keypad Interface Configuration (continued)

Register	Value	Comments
TIMCFG0	0x0103_6600	Configure enable on trigger rising edge and disable on trigger falling edge. Initial clock state is logic 0 and is not affected by reset.
TIMCTL0	0x0540_0003	Configure 16-bit counter using Shifter 1 flag (logic state) as trigger.
TIMCMP1	0x0000_0f3f	Configure 8 shifts (twice for each column) at column scan rate of divide by 128. For different scan frequency, set TIMCMP[7:0] = (scan divider / 2) - 1.
TIMCFG1	0x0020_2622	Enable on trigger rising edge, disable on compare, start and stop bits are enabled.
TIMCTL1	0x0340_0001	Configure dual 8-bit counter using Timer 0 output as the trigger.
TIMCMP2	0x0000_0801	Configure 4 shifts at half the frequency of Timer 1 trigger.
TIMCFG2	0x0110_2100	Enable on Timer 1 enable, disable on compare, decrement on trigger input with output initially negated and not affected by reset
TIMCTL2	0x0740_0001	Configure dual 8-bit counter using Timer 1 output as the trigger.
SHIFTBUF3	Keypad Scan Result	Keypad scan result can be read from SHIFTBUF3, each byte is the result of one scan with both row and column pin state from the scan (note that pin 0 is not used). Use the Shifter Status Flag to indicate when data can be written using interrupt or DMA request.

50.5 FlexIO Signal Descriptions

Signal	Description	I/O
FXIO_Dn (n=0...15)	Bidirectional FlexIO Shifter and Timer pin inputs/outputs	I/O

50.6 Memory Map and Registers

50.6.1 FLEXIO register descriptions

NOTE

An invalid register access results in a bus error. This includes reading a write-only register, writing a read-only register or accessing an invalid address.

50.6.1.1 FLEXIO memory map

FLEXIO1 base address: 401A_C000h

FLEXIO2 base address: 401B_0000h

FLEXIO3 base address: 4202_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Version ID Register (VERID)	32	RO	0101_0001h
4h	Parameter Register (PARAM)	32	RO	Table 50-21
8h	FlexIO Control Register (CTRL)	32	RW	0000_0000h
Ch	Pin State Register (PIN)	32	RO	0000_0000h
10h	Shifter Status Register (SHIFTSTAT)	32	W1C	0000_0000h
14h	Shifter Error Register (SHIFTERR)	32	W1C	0000_0000h
18h	Timer Status Register (TIMSTAT)	32	W1C	0000_0000h
20h	Shifter Status Interrupt Enable (SHIFTSIEN)	32	RW	0000_0000h
24h	Shifter Error Interrupt Enable (SHIFTEIEN)	32	RW	0000_0000h
28h	Timer Interrupt Enable Register (TIMIEN)	32	RW	0000_0000h
30h	Shifter Status DMA Enable (SHIFTSDEN)	32	RW	0000_0000h
40h	Shifter State Register (SHIFTSTATE)	32	RW	0000_0000h
80h - 9Ch	Shifter Control N Register (SHIFTCTL0 - SHIFTCTL7)	32	RW	0000_0000h
100h - 11Ch	Shifter Configuration N Register (SHIFTCFG0 - SHIFTCFG7)	32	RW	0000_0000h
200h - 21Ch	Shifter Buffer N Register (SHIFTBUF0 - SHIFTBUF7)	32	RW	0000_0000h
280h - 29Ch	Shifter Buffer N Bit Swapped Register (SHIFTBUFBIS0 - SHIFTBUFBIS7)	32	RW	0000_0000h
300h - 31Ch	Shifter Buffer N Byte Swapped Register (SHIFTBUFBYS0 - SHIFTBUFBYS7)	32	RW	0000_0000h
380h - 39Ch	Shifter Buffer N Bit Byte Swapped Register (SHIFTBUFBBS0 - SHIFTBUFBBS7)	32	RW	0000_0000h
400h - 41Ch	Timer Control N Register (TIMCTL0 - TIMCTL7)	32	RW	0000_0000h
480h - 49Ch	Timer Configuration N Register (TIMCFG0 - TIMCFG7)	32	RW	0000_0000h
500h - 51Ch	Timer Compare N Register (TIMCMP0 - TIMCMP7)	32	RW	0000_0000h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
680h - 69Ch	Shifter Buffer N Nibble Byte Swapped Register (SHIFTBUFNBS0 - SHIFTBUFNBS7)	32	RW	0000_0000h
700h - 71Ch	Shifter Buffer N Half Word Swapped Register (SHIFTBUFHWS0 - SHIFTBUFHWS7)	32	RW	0000_0000h
780h - 79Ch	Shifter Buffer N Nibble Swapped Register (SHIFTBUFNIS0 - SHIFTBUFNIS7)	32	RW	0000_0000h

50.6.1.2 Version ID Register (VERID)

50.6.1.2.1 Offset

Register	Offset
VERID	0h

50.6.1.2.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	MAJOR								MINOR							
W																
Reset	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FEATURE															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

50.6.1.2.3 Fields

Field	Function
31-24	Major Version Number
MAJOR	This read only field returns the major version number for the module specification.
23-16	Minor Version Number
MINOR	This read only field returns the minor version number for the module specification.
15-0	Feature Specification Number
FEATURE	This read only field returns the feature set number.

Field	Function
	0000000000000000b - Standard features implemented. 0000000000000001b - Supports state, logic and parallel modes.

50.6.1.3 Parameter Register (PARAM)

50.6.1.3.1 Offset

Register	Offset
PARAM	4h

50.6.1.3.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TRIGGER								PIN							
W																
Reset	See Register reset values.															

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TIMER								SHIFTER							
W																
Reset	See Register reset values.															

50.6.1.3.3 Register reset values

Register	Reset value
PARAM	FLEXIO1: 0210_0808h FLEXIO2,FLEXIO3: 0220_0808h

50.6.1.3.4 Fields

Field	Function
31-24 TRIGGER	Trigger Number Number of external triggers implemented.

Table continues on the next page...

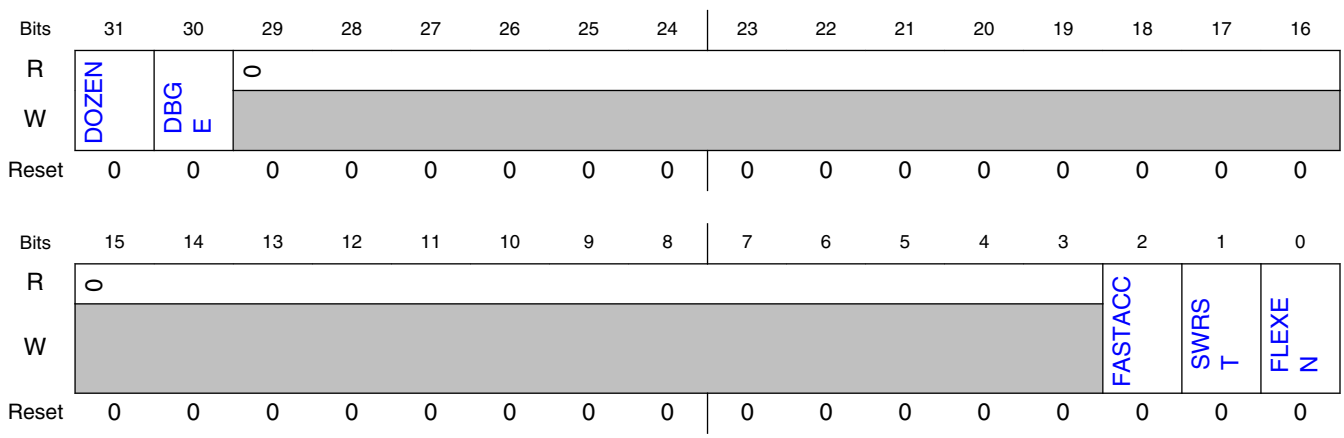
Field	Function
23-16 PIN	Pin Number Number of Pins implemented.
15-8 TIMER	Timer Number Number of Timers implemented.
7-0 SHIFTER	Shifter Number Number of Shifters implemented.

50.6.1.4 FlexIO Control Register (CTRL)

50.6.1.4.1 Offset

Register	Offset
CTRL	8h

50.6.1.4.2 Diagram



50.6.1.4.3 Fields

Field	Function
31 DOZEN	Doze Enable Disables FlexIO operation in Doze modes. 0b - FlexIO enabled in Doze modes. 1b - FlexIO disabled in Doze modes.
30	Debug Enable

Table continues on the next page...

Field	Function
DBGE	Enables FlexIO operation in Debug mode. 0b - FlexIO is disabled in debug modes. 1b - FlexIO is enabled in debug modes
29-3 —	Reserved
2 FASTACC	Fast Access Enables fast register accesses to FlexIO registers, but requires the FlexIO functional clock to be at least two times faster than the frequency of the bus clock. 0b - Configures for normal register accesses to FlexIO 1b - Configures for fast register accesses to FlexIO
1 SWRST	Software Reset The FlexIO Control Register is not affected by the software reset, all other logic in the FlexIO is affected by the software reset and all other register accesses are ignored until this bit is cleared. This register bit remains set until cleared by software, and the reset has cleared in the FlexIO clock domain. 0b - Software reset is disabled 1b - Software reset is enabled, all FlexIO registers except the Control Register are reset.
0 FLEXEN	FlexIO Enable 0b - FlexIO module is disabled. 1b - FlexIO module is enabled.

50.6.1.5 Pin State Register (PIN)

50.6.1.5.1 Offset

Register	Offset
PIN	Ch

50.6.1.5.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PDI															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PDI															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

50.6.1.5.3 Fields

Field	Function
31-0 PDI	Pin Data Input Returns the input data on each of the FlexIO pins. **NOTE:** This field is not supported in every instance. The following table includes only supported registers.

Instance	Field supported in	Field not supported in
FLEXIO1	PIN[15–0]	PIN[31–16]
FLEXIO2	PIN	—
FLEXIO3	PIN	—

50.6.1.6 Shifter Status Register (SHIFTSTAT)

50.6.1.6.1 Offset

Register	Offset
SHIFTSTAT	10h

50.6.1.6.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								SSF							
W									W1C							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

50.6.1.6.3 Fields

Field	Function
31-8	Reserved

Table continues on the next page...

Field	Function
—	
7-0 SSF	Shifter Status Flag The shifter status flag is updated when one of the following events occurs: For SMOD =Receive, the status flag is set when SHIFTBUF has been loaded with data from Shifter (SHIFTBUF is full), and the status flag is cleared when SHIFTBUF register is read. For SMOD =Transmit, the status flag is set when SHIFTBUF data has been transferred to the Shifter (SHIFTBUF is empty) or when initially configured for SMOD=Transmit, and the status flag is cleared when the SHIFTBUF register is written. For SMOD =Match Store, the status flag is set when a match has occured between SHIFTBUF and Shifter, and the status flag is cleared when the SHIFTBUF register is read. For SMOD =Match Continuous, returns the current match result between the SHIFTBUF and Shifter. The status flag cannot be cleared by reading SHIFTBUF. For SMOD =State, the status flag for a shifter sets when it is selected by the current state pointer. For SMOD =Logic, returns the current value of the programmable logic block output. The status flag can also be cleared by writing a logic one to the flag for all modes except Match Continuous/State/Logic. 0b - Status flag is clear. 1b - Status flag is set.

50.6.1.7 Shifter Error Register (SHIFTERR)

50.6.1.7.1 Offset

Register	Offset
SHIFTERR	14h

50.6.1.7.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								SEF							
W									W1C							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

50.6.1.7.3 Fields

Field	Function
31-8 —	Reserved
7-0 SEF	Shifter Error Flags The shifter error flag is set when one of the following events occurs: For SMOD =Receive, indicates Shifter was ready to store new data into SHIFTBUF before the previous data was read from SHIFTBUF (SHIFTBUF Overrun), or indicates that the received start or stop bit does not match the expected value. For SMOD =Transmit, indicates Shifter was ready to load new data from SHIFTBUF before new data had been written into SHIFTBUF (SHIFTBUF Underrun). For SMOD =Match Store, indicates a match event occured before the previous match data was read from SHIFTBUF (SHIFTBUF Overrun). For SMOD =Match Continuous, the error flag is set when a match has occured between SHIFTBUF and Shifter. For SMOD =Logic, the error flag is set when the output of the programmable logic block has asserted. Can be cleared by writing logic one to the flag. For SMOD=Match Continuous, can also be cleared when the SHIFTBUF register is read. 0b - Shifter Error Flag is clear. 1b - Shifter Error Flag is set.

50.6.1.8 Timer Status Register (TIMSTAT)

50.6.1.8.1 Offset

Register	Offset
TIMSTAT	18h

50.6.1.8.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								TSF							
W									W1C							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

50.6.1.8.3 Fields

Field	Function
31-8 —	Reserved
7-0 TSF	Timer Status Flags The timer status flag sets depending on the timer mode, and can be cleared by writing logic one to the flag. In 8-bit baud counter mode, the timer status flag is set when the upper 8-bit counter equals zero and decrements. In 8-bit high PWM mode, the timer status flag is set when the upper 8-bit counter equals zero and decrements. In 16-bit counter mode, the timer status flag is set when the 16-bit counter equals zero and decrements. 0b - Timer Status Flag is clear. 1b - Timer Status Flag is set.

50.6.1.9 Shifter Status Interrupt Enable (SHIFTSIEN)

50.6.1.9.1 Offset

Register	Offset
SHIFTSIEN	20h

50.6.1.9.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								SSIE							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

50.6.1.9.3 Fields

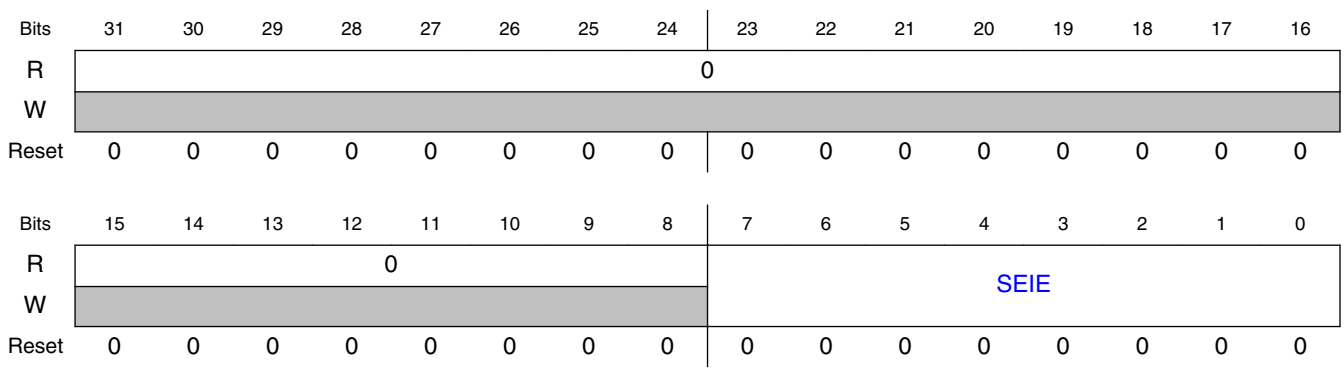
Field	Function
31-8 —	Reserved
7-0 SSIE	Shifter Status Interrupt Enable Enables interrupt generation when corresponding SSF is set. 0b - Shifter Status Flag interrupt disabled. 1b - Shifter Status Flag interrupt enabled.

50.6.1.10 Shifter Error Interrupt Enable (SHIFTEIEN)

50.6.1.10.1 Offset

Register	Offset
SHIFTEIEN	24h

50.6.1.10.2 Diagram



50.6.1.10.3 Fields

Field	Function
31-8 —	Reserved
7-0 SEIE	Shifter Error Interrupt Enable Enables interrupt generation when corresponding SEF is set.

Field	Function
	0b - Shifter Error Flag interrupt disabled. 1b - Shifter Error Flag interrupt enabled.

50.6.1.11 Timer Interrupt Enable Register (TIMIEN)

50.6.1.11.1 Offset

Register	Offset
TIMIEN	28h

50.6.1.11.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								TEIE							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

50.6.1.11.3 Fields

Field	Function
31-8 —	Reserved
7-0 TEIE	Timer Status Interrupt Enable Enables interrupt generation when corresponding TSF is set. 0b - Timer Status Flag interrupt is disabled. 1b - Timer Status Flag interrupt is enabled.

50.6.1.12 Shifter Status DMA Enable (SHIFTSDEN)

50.6.1.12.1 Offset

Register	Offset
SHIFTSDEN	30h

50.6.1.12.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								SSDE							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

50.6.1.12.3 Fields

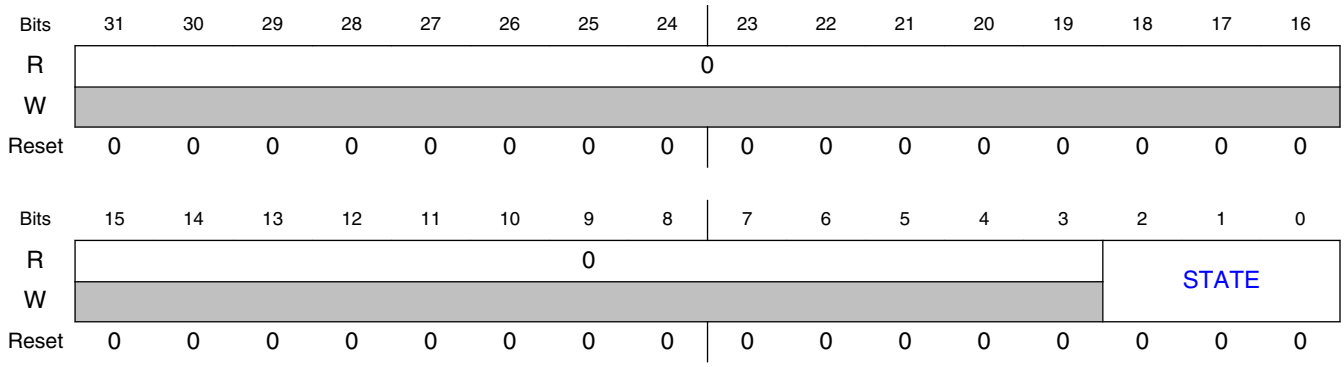
Field	Function
31-8 —	Reserved
7-0 SSDE	Shifter Status DMA Enable Enables DMA request generation when corresponding SSF is set. 0b - Shifter Status Flag DMA request is disabled. 1b - Shifter Status Flag DMA request is enabled.

50.6.1.13 Shifter State Register (SHIFTSTATE)

50.6.1.13.1 Offset

Register	Offset
SHIFTSTATE	40h

50.6.1.13.2 Diagram



50.6.1.13.3 Fields

Field	Function
31-3 —	Reserved
2-0 STATE	Current State Pointer The current state field maintains a pointer to keep track of the current Shifter (configured for State mode) enabled to drive outputs and compute the next state. Reading this register when the state pointer is updating can result in the incorrect state returned.

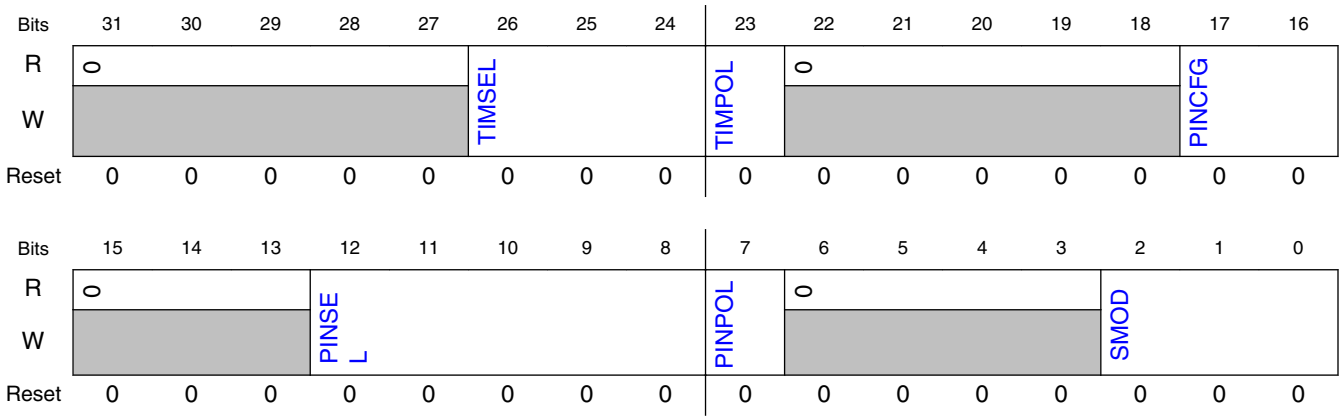
50.6.1.14 Shifter Control N Register (SHIFTCTL0 - SHIFTCTL7)

50.6.1.14.1 Offset

For a = 0 to 7:

Register	Offset
SHIFTCTLa	80h + (a × 4h)

50.6.1.14.2 Diagram



50.6.1.14.3 Fields

Field	Function
31-27 —	Reserved
26-24 TIMSEL	Timer Select Selects which Timer is used for controlling the logic/shift register and generating the Shift clock. TIMSEL=i selects TIMERi.
23 TIMPOL	Timer Polarity 0b - Shift on posedge of Shift clock 1b - Shift on negedge of Shift clock
22-18 —	Reserved
17-16 PINCFG	Shifter Pin Configuration For pins configured as an output (PINCFG=11), this field takes effect when the register is written. 00b - Shifter pin output disabled 01b - Shifter pin open drain or bidirectional output enable 10b - Shifter pin bidirectional output data 11b - Shifter pin output
15-13 —	Reserved
12-8 PINSEL	Shifter Pin Select Selects which pin is used by the Shifter input or output. PINSEL=i selects the FXIO_Di pin. For pins configured as an output (PINCFG=11), this field takes effect when the register is written. **NOTE:** This field is not supported in every instance. The following table includes only supported registers.

Instance	Field supported in	Field not supported in
FLEXIO1	SHIFTCTL0–SHIFTCTL7[11–8]	SHIFTCTL0–SHIFTCTL7[12]
FLEXIO2	SHIFTCTL0–SHIFTCTL7	—

Table continues on the next page...

Field	Function		
	Instance	**Field supported in**	**Field not supported in**
	FLEXIO3	SHIFTCTL0–SHIFTCTL7	—
7 PINPOL	Shifter Pin Polarity For pins configured as an output (PINCFG=11), this field takes effect when the register is written. 0b - Pin is active high 1b - Pin is active low		
6-3 —	Reserved		
2-0 SMOD	Shifter Mode Configures the mode of the Shifter. 000b - Disabled. 001b - Receive mode. Captures the current Shifter content into the SHIFTBUF on expiration of the Timer. 010b - Transmit mode. Load SHIFTBUF contents into the Shifter on expiration of the Timer. 011b - Reserved. 100b - Match Store mode. Shifter data is compared to SHIFTBUF content on expiration of the Timer. 101b - Match Continuous mode. Shifter data is continuously compared to SHIFTBUF contents. 110b - State mode. SHIFTBUF contents are used for storing programmable state attributes. 111b - Logic mode. SHIFTBUF contents are used for implementing programmable logic look up table.		

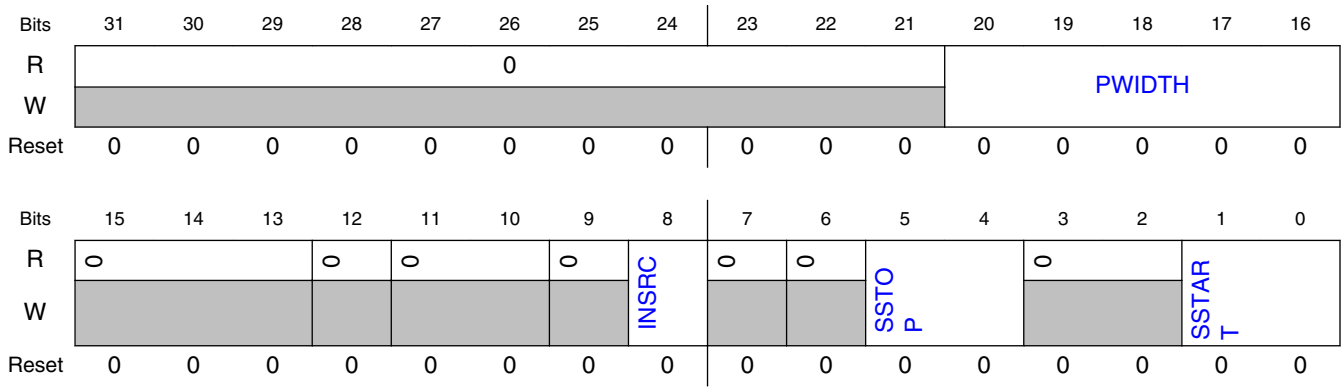
50.6.1.15 Shifter Configuration N Register (SHIFTCFG0 - SHIFTCFG7)

50.6.1.15.1 Offset

For a = 0 to 7:

Register	Offset
SHIFTCFGa	100h + (a × 4h)

50.6.1.15.2 Diagram



50.6.1.15.3 Fields

Field	Function
31-21 —	Reserved
20-16 PWIDTH	Parallel Width For all Shifters, this register field configures the number of bits to be shifted on each Shift clock as follows: 1-bit shift for PWIDTH=0 4-bit shift for PWIDTH=1...3 8-bit shift for PWIDTH=4...7 16-bit shift for PWIDTH=8...15 For Shifters which support parallel transmit (SHIFTER0, SHIFTER4, ...) or parallel receive (SHIFTER3, SHIFTER7, ...), this register field, together with SHIFTCTL[PINSEL], also selects the pins to be driven or sampled on each Shift clock as follows: • FXIO_D[PINSEL+PWIDTH]:FXIO_D[PINSEL] Shifters which do not support parallel transmit or parallel receive only support parallel shift when INSRC =1. For SMOD =State, this field is used to disable state outputs. See State Mode section for more detail. **NOTE:** This field is not supported in every instance. The following table includes only supported registers.
15-13 —	Reserved

Instance	Field supported in	Field not supported in
FLEXIO1	SHIFTCFG0–SHIFTCFG7[19–16]	SHIFTCFG0–SHIFTCFG7[20]
FLEXIO2	SHIFTCFG0–SHIFTCFG7	—
FLEXIO3	SHIFTCFG0–SHIFTCFG7	—

Table continues on the next page...

Field	Function
12 —	Reserved
11-10 —	Reserved
9 —	Reserved
8 INSRC	Input Source Selects the input source for the shifter. Configuring INSRC=1 is not supported for the last shifter. 0b - Pin 1b - Shifter N+1 Output
7 —	Reserved
6 —	Reserved
5-4 SSTOP	Shifter Stop bit For SMOD =Transmit, this field allows automatic stop bit insertion if the selected timer has also enabled a stop bit. For SMOD =Receive or Match Store, this field allows automatic stop bit checking if the selected timer has also enabled a stop bit. For SMOD =State, this field is used to disable state outputs. See 'State Mode' section for more detail. For SMOD =Logic, this field is used to mask logic pin inputs. See 'Logic Mode' section for more detail. 00b - Stop bit disabled for transmitter/receiver/match store 01b - Reserved for transmitter/receiver/match store 10b - Transmitter outputs stop bit value 0 on store, receiver/match store sets error flag if stop bit is not 0 11b - Transmitter outputs stop bit value 1 on store, receiver/match store sets error flag if stop bit is not 1
3-2 —	Reserved
1-0 SSTART	Shifter Start bit For SMOD =Transmit, this field allows automatic start bit insertion if the selected timer has also enabled a start bit. For SMOD =Receive or Match Store, this field allows automatic start bit checking if the selected timer has also enabled a start bit. For SMOD =State, this field is used to disable state outputs. See 'State Mode' section for more detail. For SMOD =Logic, this field is used to mask logic pin inputs. See 'Logic Mode' section for more detail. 00b - Start bit disabled for transmitter/receiver/match store, transmitter loads data on enable 01b - Start bit disabled for transmitter/receiver/match store, transmitter loads data on first shift 10b - Transmitter outputs start bit value 0 before loading data on first shift, receiver/match store sets error flag if start bit is not 0 11b - Transmitter outputs start bit value 1 before loading data on first shift, receiver/match store sets error flag if start bit is not 1

50.6.1.16 Shifter Buffer N Register (SHIFTBUF0 - SHIFTBUF7)

50.6.1.16.1 Offset

For a = 0 to 7:

Register	Offset
SHIFTBUFa	200h + (a × 4h)

50.6.1.16.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SHIFTBUF															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SHIFTBUF															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

50.6.1.16.3 Fields

Field	Function
31-0 SHIFTBUF	Shift Buffer Shift buffer data is used for a variety of functions depending on the SMOD setting: For SMOD=Receive, Shifter data is transferred into SHIFTBUF at the expiration of Timer. The SHIFTBUF register should only be read when the corresponding shifter status flag is set, indicating new Shifter data is available. For SMOD=Transmit, SHIFTBUF data is transferred into the Shifter before the Timer begins. For SMOD=Match Store, SHIFTBUF[31:16] contains the data to be matched with the Shifter contents and SHIFTBUF[15:0] can be used to mask the match result (1=mask, 0=no mask). The Match is checked when the Timer expires and Shifter data [31:16] is written to SHIFTBUF[31:16] whenever a match event occurs. The SHIFTBUF register should only be read when the corresponding shifter status flag is set, indicating new Shifter data is available. For SMOD=Match Continuous, SHIFTBUF[31:16] contains the data to be matched with the Shifter contents and SHIFTBUF[15:0] can be used to mask the match result (1=mask, 0=no mask). For SMOD=Logic, SHIFTBUF[31:0] is used to implement a 5-input, 32-bit programmable logic look-up table. See Logic Mode section for more detail. For SMOD=State, SHIFTBUF[31:24] is used to drive the output value when this shifter is selected by the current state pointer and SHIFTBUF[23:0] is used to configure the value of the next state transition. See State Mode section for more detail.

50.6.1.17 Shifter Buffer N Bit Swapped Register (SHIFTBUFBIS0 - SHIFTBUFBIS7)

50.6.1.17.1 Offset

For a = 0 to 7:

Register	Offset
SHIFTBUFBISa	280h + (a × 4h)

50.6.1.17.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SHIFTBUFBIS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SHIFTBUFBIS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

50.6.1.17.3 Fields

Field	Function
31-0 SHIFTBUFBIS	Shift Buffer Alias to SHIFTBUF register, except reads/writes to this register are bit swapped. Reads return SHIFTBUF[0:31].

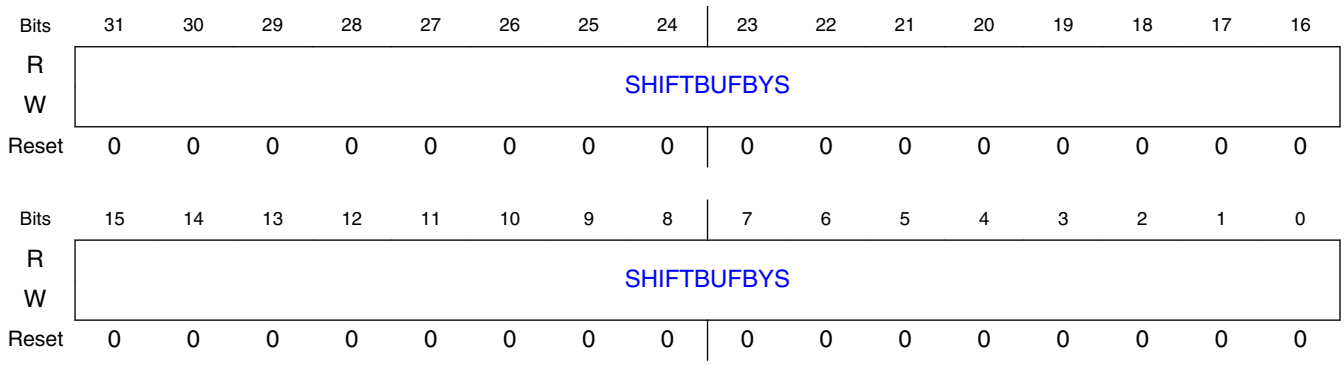
50.6.1.18 Shifter Buffer N Byte Swapped Register (SHIFTBUFBYS0 - SHIFTBUFBYS7)

50.6.1.18.1 Offset

For a = 0 to 7:

Register	Offset
SHIFTBUFBYSa	300h + (a × 4h)

50.6.1.18.2 Diagram



50.6.1.18.3 Fields

Field	Function
31-0 SHIFTBUFBYS	Shift Buffer Alias to SHIFTBUF register, except reads/writes to this register are byte swapped. Reads return { SHIFTBUF[7:0], SHIFTBUF[15:8], SHIFTBUF[23:16], SHIFTBUF[31:24] }.

50.6.1.19 Shifter Buffer N Bit Byte Swapped Register (SHIFTBUFBBS0 - SHIFTBUFBBS7)

50.6.1.19.1 Offset

For a = 0 to 7:

Register	Offset
SHIFTBUFBBSa	380h + (a × 4h)

50.6.1.19.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SHIFTBUFBBS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SHIFTBUFBBS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

50.6.1.19.3 Fields

Field	Function
31-0 SHIFTBUFBBS	Shift Buffer Alias to SHIFTBUF register, except reads/writes to this register are bit swapped within each byte. Reads return { SHIFTBUF[24:31], SHIFTBUF[16:23], SHIFTBUF[8:15], SHIFTBUF[0:7] }.

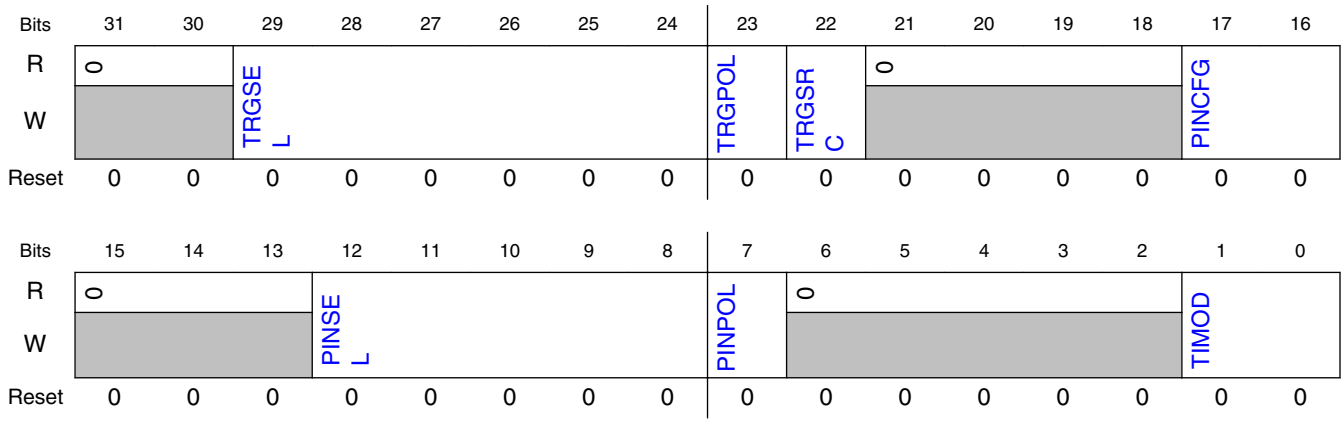
50.6.1.20 Timer Control N Register (TIMCTL0 - TIMCTL7)

50.6.1.20.1 Offset

For a = 0 to 7:

Register	Offset
TIMCTLa	400h + (a × 4h)

50.6.1.20.2 Diagram



50.6.1.20.3 Fields

Field	Function
31-30 —	Reserved
29-24 TRGSEL	Trigger Select The valid values for TRGSEL depends on the FLEXIO_PARAM register. • When TRGSRC = 1, the valid values for N depends on PIN, TIMER, SHIFTER fields in the FLEXIO_PARAM register. • When TRGSRC = 0, the valid values for N depends on TRIGGER field in FLEXIO_PARAM register. Refer to the chip-specific FlexIO information for external trigger selection. **NOTE:** For a pin, N=0 to 15. For a Shifter, N=0 to 7. For a Timer, N=0 to 7. When TRGSRC = 0 the trigger selection is configured as follows: • N - External trigger N input When TRGSRC = 1 the internal trigger can be configured to select an input pin as follows: • 2*N - Pin N input When TRGSRC = 1 the internal trigger can be configured to select a shifter/timer signal as follows: • 4*N+1 - Shifter N status flag • 4*N+3 - Timer N trigger output In other words, the expanded internal trigger selection (TRGSRC = 1) is as follows: • 0000 = Pin 0 • 0001 = Shifter 0 Flag • 0010 = Pin 1 • 0011 = Timer 0 Trigger • 0100 = Pin 2 • 0101 = Shifter 1 Flag • 0110 = Pin 3 • 0111 = Timer 1 Trigger • ... • This continues up to the Pin 15, Shifter 7 and Timer 7.

Table continues on the next page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>
NOTE: This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>FLEXIO1</td><td>TIMCTL0–TIMCTL7[28–24]</td><td>TIMCTL0–TIMCTL7[29]</td></tr>
<tr><td>FLEXIO2</td><td>TIMCTL0–TIMCTL7</td><td>—</td></tr>
<tr><td>FLEXIO3</td><td>TIMCTL0–TIMCTL7</td><td>—</td></tr>
</table>
</td></tr>
<tr><td>23
TRGPOL</td><td>Trigger Polarity
0b - Trigger active high
1b - Trigger active low</td></tr>
<tr><td>22
TRGSRC</td><td>Trigger Source
0b - External trigger selected
1b - Internal trigger selected</td></tr>
<tr><td>21-18
—</td><td>Reserved</td></tr>
<tr><td>17-16
PINCFG</td><td>Timer Pin Configuration
Configures the direction of the Timer pin. For pins configured as an output (PINCFG=11), this field takes effect when the register is written.
00b - Timer pin output disabled
01b - Timer pin open drain or bidirectional output enable
10b - Timer pin bidirectional output data
11b - Timer pin output</td></tr>
<tr><td>15-13
—</td><td>Reserved</td></tr>
<tr><td>12-8
PINSEL</td><td>Timer Pin Select
Selects which pin is used by the Timer input or output. PINSEL=i selects the FXIO_Di pin. For pins configured as an output (PINCFG=11), this field takes effect when the register is written.
NOTE: This field is not supported in every instance. The following table includes only supported registers.
<table>
<tr><th>Instance</th><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>FLEXIO1</td><td>TIMCTL0–TIMCTL7[11–8]</td><td>TIMCTL0–TIMCTL7[12]</td></tr>
<tr><td>FLEXIO2</td><td>TIMCTL0–TIMCTL7</td><td>—</td></tr>
<tr><td>FLEXIO3</td><td>TIMCTL0–TIMCTL7</td><td>—</td></tr>
</table>
</td></tr>
<tr><td>7
PINPOL</td><td>Timer Pin Polarity
For pins configured as an output (PINCFG=11), this field takes effect when the register is written.
0b - Pin is active high
1b - Pin is active low</td></tr>
<tr><td>6-2
—</td><td>Reserved</td></tr>
<tr><td>1-0
TIMOD</td><td>Timer Mode
In 8-bit baud counter mode, the lower 8-bits of the counter and compare register are used to configure the baud rate of the timer shift clock, and the upper 8-bits are used to configure the shifter bit count.</td></tr>
</table>

Field	Function
	In 8-bit PWM high mode, the lower 8-bits of the counter and compare register are used to configure the high period of the timer shift clock, and the upper 8-bits are used to configure the low period of the timer shift clock. The shifter bit count is configured using another timer or external signal. In 16-bit counter mode, the full 16-bits of the counter and compare register are used to configure either the baud rate of the shift clock or the shifter bit count. 00b - Timer Disabled. 01b - Dual 8-bit counters baud mode. 10b - Dual 8-bit counters PWM high mode. 11b - Single 16-bit counter mode.

50.6.1.21 Timer Configuration N Register (TIMCFG0 - TIMCFG7)

50.6.1.21.1 Offset

For a = 0 to 7:

Register	Offset
TIMCFGa	480h + (a × 4h)

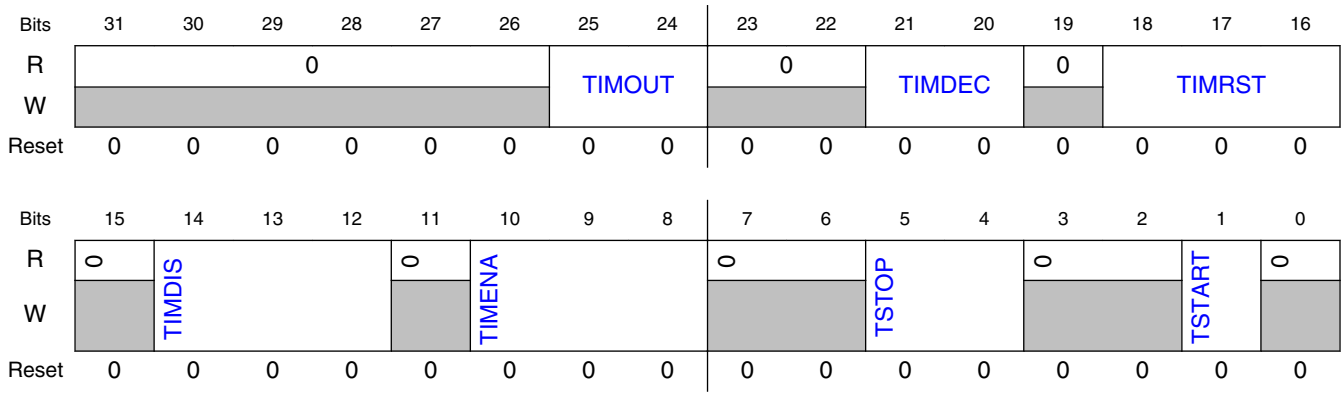
50.6.1.21.2 Function

The options to enable or disable the timer using the Timer N-1 enable or disable are reserved when N is evenly divisible by 4 (eg: Timer 0).

NOTE

The pin and trigger level/edges specified below refer to the signal state after being modified by the PINPOL or TRGPOL setting in the TIMCTL register. For example, "Trigger low" means a trigger is actually at logic level 1 if the TRGPOL is set to 1 (active low).

50.6.1.21.3 Diagram



50.6.1.21.4 Fields

Field	Function
31-26 —	Reserved
25-24 TIMOUT	Timer Output Configures the initial state of the Timer Output and whether it is affected by the Timer reset. 00b - Timer output is logic one when enabled and is not affected by timer reset 01b - Timer output is logic zero when enabled and is not affected by timer reset 10b - Timer output is logic one when enabled and on timer reset 11b - Timer output is logic zero when enabled and on timer reset
23-22 —	Reserved
21-20 TIMDEC	Timer Decrement Configures the source of the Timer decrement and the source of the Shift clock. 00b - Decrement counter on FlexIO clock, Shift clock equals Timer output. 01b - Decrement counter on Trigger input (both edges), Shift clock equals Timer output. 10b - Decrement counter on Pin input (both edges), Shift clock equals Pin input. 11b - Decrement counter on Trigger input (both edges), Shift clock equals Trigger input.
19 —	Reserved
18-16 TIMRST	Timer Reset Configures the condition that causes the timer counter (and optionally the timer output) to be reset. In 8-bit counter mode, the timer reset only resets the lower 8-bits that configure the baud rate. In all other modes, the timer reset resets the full 16-bits of the counter. 000b - Timer never reset 001b - Reserved 010b - Timer reset on Timer Pin equal to Timer Output 011b - Timer reset on Timer Trigger equal to Timer Output 100b - Timer reset on Timer Pin rising edge 101b - Reserved 110b - Timer reset on Trigger rising edge 111b - Timer reset on Trigger rising or falling edge

Table continues on the next page...

Field	Function
15 —	Reserved
14-12 TIMDIS	Timer Disable Configures the condition that causes the Timer to be disabled and stop decrementing. 000b - Timer never disabled 001b - Timer disabled on Timer N-1 disable 010b - Timer disabled on Timer compare (upper 8-bits match and decrement) 011b - Timer disabled on Timer compare (upper 8-bits match and decrement) and Trigger Low 100b - Timer disabled on Pin rising or falling edge 101b - Timer disabled on Pin rising or falling edge provided Trigger is high 110b - Timer disabled on Trigger falling edge 111b - Reserved
11 —	Reserved
10-8 TIMENA	Timer Enable Configures the condition that causes the Timer to be enabled and start decrementing. 000b - Timer always enabled 001b - Timer enabled on Timer N-1 enable 010b - Timer enabled on Trigger high 011b - Timer enabled on Trigger high and Pin high 100b - Timer enabled on Pin rising edge 101b - Timer enabled on Pin rising edge and Trigger high 110b - Timer enabled on Trigger rising edge 111b - Timer enabled on Trigger rising or falling edge
7-6 —	Reserved
5-4 TSTOP	Timer Stop Bit The stop bit can be added on a timer compare (between each word) or on a timer disable. When stop bit is enabled, configured shifters output the contents of the stop bit when the timer is disabled. When stop bit is enabled on timer disable, the timer remains disabled until the next rising edge of the shift clock. If configured for both timer compare and timer disable, only one stop bit is inserted on timer disable. 00b - Stop bit disabled 01b - Stop bit is enabled on timer compare 10b - Stop bit is enabled on timer disable 11b - Stop bit is enabled on timer compare and timer disable
3-2 —	Reserved
1 TSTART	Timer Start Bit When start bit is enabled, configured shifters outputs the contents of the start bit when the timer is enabled and the timer counter reloads from the compare register on the first rising edge of the shift clock. 0b - Start bit disabled 1b - Start bit enabled
0 —	Reserved

50.6.1.22 Timer Compare N Register (TIMCMP0 - TIMCMP7)

50.6.1.22.1 Offset

For a = 0 to 7:

Register	Offset
TIMCMPa	500h + (a × 4h)

50.6.1.22.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CMP															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

50.6.1.22.3 Fields

Field	Function
31-16 —	Reserved
15-0 CMP	Timer Compare Value The timer compare value is loaded into the timer counter when the timer is first enabled, when the timer is reset and when the timer decrements down to zero. In 8-bit baud counter mode, the lower 8-bits configure the baud rate divider equal to (CMP[7:0] + 1) * 2. The upper 8-bits configure the number of bits in each word equal to (CMP[15:8] + 1) / 2. In 8-bit PWM high mode, the lower 8-bits configure the high period of the output to (CMP[7:0] + 1) and the upper 8-bits configure the low period of the output to (CMP[15:8] + 1). In 16-bit counter mode, the compare value can be used to generate the baud rate divider (if shift clock source is timer output) to equal (CMP[15:0] + 1) * 2. When the shift clock source is a pin or trigger input, the compare register is used to set the number of bits in each word equal to (CMP[15:0] + 1) / 2.

50.6.1.23 Shifter Buffer N Nibble Byte Swapped Register (SHIFTBUFNBS0 - SHIFTBUFNBS7)

50.6.1.23.1 Offset

For a = 0 to 7:

Register	Offset
SHIFTBUFNBSa	680h + (a × 4h)

50.6.1.23.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SHIFTBUFNBS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SHIFTBUFNBS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

50.6.1.23.3 Fields

Field	Function
31-0 SHIFTBUFNBS	Shift Buffer Alias to SHIFTBUF register, except reads/writes to this register are nibble swapped within each byte. Reads return { SHIFTBUF[27:24], SHIFTBUF[31:28], SHIFTBUF[19:16], SHIFTBUF[23:20], SHIFTBUF[11:8], SHIFTBUF[15:12], SHIFTBUF[3:0], SHIFTBUF[7:4] }.

50.6.1.24 Shifter Buffer N Half Word Swapped Register (SHIFTBUFHWS0 - SHIFTBUFHWS7)

50.6.1.24.1 Offset

For a = 0 to 7:

Register	Offset
SHIFTBUFHWSa	700h + (a × 4h)

50.6.1.24.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SHIFTBUFHWS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SHIFTBUFHWS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

50.6.1.24.3 Fields

Field	Function
31-0 SHIFTBUFHWS	Shift Buffer Alias to SHIFTBUF register, except reads/writes to this register are half word swapped. Reads return { SHIFTBUF[15:0], SHIFTBUF[31:16] }.

50.6.1.25 Shifter Buffer N Nibble Swapped Register (SHIFTBUFNIS0 - SHIFTBUFNIS7)

50.6.1.25.1 Offset

For a = 0 to 7:

Register	Offset
SHIFTBUFNISa	780h + (a × 4h)

50.6.1.25.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SHIFTBUFNIS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SHIFTBUFNIS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

50.6.1.25.3 Fields

Field	Function
31-0 SHIFTBUFNIS	Shift Buffer Alias to SHIFTBUF register, except reads/writes to this register are nibble swapped. Reads return { SHIFTBUF[3:0], SHIFTBUF[7:4], SHIFTBUF[11:8], SHIFTBUF[15:12], SHIFTBUF[19:16], SHIFTBUF[23:20], SHIFTBUF[27:24], SHIFTBUF[31:28] }.

Chapter 51
Timers Overview

51.1 Overview

The following timers are supported in this chip:

- General Purpose Timer (GPT): A 32-bit up-counter with 12-bit pre-scaler
- Periodic Interrupt Timer (PIT): A 32-bit counter timer that features programmable count modulus, clock division features etc.
- Quad Timer (TMR): It provides four timer channels with variety of controls for individual and multi-channel features
- Quadrature Decoder (QDC): It provides interfacing capability to position/speed sensors
- Enhanced FlexPWM: It contains PWM submodules each of which is set up to control a single half bridge power stage
- Watchdog Timer (WDOG1,2): The WDOG1 and WDOG2 protect against system failures by providing a method by which to escape from unexpected events or programming errors
- Watchdog timer (WDOG3): It is a high reliability independent timer that is available for system use
- External Watchdog Monitor (EWM): It is designed to monitor external circuits, as well as the MCU software flow

51.1.1 General Purpose Timer (GPT)

The GPT has a 32-bit up-counter. The timer counter value can be captured in a register using an event on an external pin. The capture trigger can be programmed to be a rising or/and falling edge. The GPT also features output compare event generation, when the timer matches a preset value, GPT can generate an interrupt for software and a compare event signal on output pins for external hardware.

The GPT has a 12-bit pre-scaler, which provides a programmable clock frequency derived from multiple clock sources.

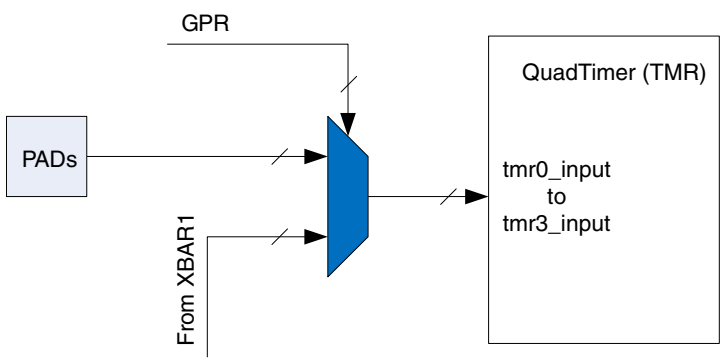
51.1.2 Periodic Interrupt Timer (PIT)

The PIT is a basic 32 bit counter timer. It features 32-bit counter timer, programmable count modulus, clock division features, interrupt generation, and ability to chain adjacent timers to achieve longer interval

51.1.3 Quad Timer (TMR)

The quad-timer provides four timer channels. Specific features include up/down count, cascading of counters, programmable modulo, count once/repeated, counter preload, compare registers with preload, shared use of input signals, prescaler controls, independent capture/compare, fault input control, programmable input filters, and multi-channel synchronization.

As shown in diagram below, input of QTimer can be sourced from PADs or XBAR1:



51.1.4 Enhanced Flex Pulse Width Modulator (eFlexPWM)

This module can generate various switching patterns, including highly sophisticated waveforms. Each module shall be configured with 4 channels supporting A, B, and X PWM output. All channels are configured for standard edge placement and have their capture function enabled.

51.1.5 Quadrature Decoder (QDC)

The enhanced Quadrature Decoder module provides interfacing capability to position/ speed sensors used in industrial motor control applications. It has five input signals: PHASEA, PHASEB, INDEX, TRIGGER, and HOME. This module is used to decode shaft position, revolution count and speed.

51.1.6 Watchdog Timer (WDOG1,2)

The WDOG1 and WDOG2 protect against system failures by providing a method by which to escape from unexpected events or programming errors.

After WDOG1 and WDOG2 are activated, it must be serviced by the software on a periodic basis. If servicing does not take place, the timer times out. Upon timeout, the WDOG1 asserts the internal system reset signal, WDOG_RESET_B_DEB to the System Reset Controller (SRC); and WDOG2 asserts interrupt to SNVS to report the security violation condition.

Main features of the WDOG1 and WDOG2 module are:

- Configurable timeout counter with timeout periods from 0.5 to 128 seconds which, after timeout expiration, result in the assertion of WDOG_RESET_B_DEB reset signal
- Time resolution of 0.5 seconds
- Configurable timeout counter that can be programmed to run or stop during low power modes
- Configurable timeout counter that can be programmed to run or stop during DEBUG mode
- Programmable interrupt generation prior to timeout

51.1.7 Watchdog Timer (WDOG3)

The WDOG module is an high reliability independent timer that is available for system use. It provides a safety feature to ensure that software is executing as planned and that the CPU is not stuck in an infinite loop or executing unintended code. If the WDOG module is not serviced (refreshed) within a certain period, it resets the CPU.

Main features of the WDOG module are:

- Configurable independent clock source input from bus clock
- Programmable time-out period
- Robust write sequence for counter refresh

- Windowed refresh option
- Optional timeout interrupt to allow post-processing diagnostics
- Configuration bits are write-once-after-reset to ensure watchdog configuration cannot be mistakenly altered
- Robust write sequence for unlocking write-once control/configuration bits

The WDOG3 clock can be sourced from:

- 1MHz RC OSC
- 32KHz clock generated by 32 KHZ XTALOSC which could be automatically switched to 32KHz RC OSC upon XTAL clock loss
- IP Bus Clock

51.1.8 External Watchdog Monitor (EWM)

The External Watchdog Monitor provides a back-up mechanism to the internal WDOG that can reset the system. The EWM differs from the internal WDOG in that it does not reset the system.

The EWM, if allowed to time-out, provides an independent trigger pin that when asserted resets or places an external circuit into a safe mode.

Chapter 52
General Purpose Timer (GPT)

52.1 Chip-specific GPT information

Table 52-1. Reference links to related information

Topic	Related module(s)	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
Interrupts, DMA Events and XBAR Assignments	-	Interrupts, DMA Events and XBAR Assignments

52.2 Overview

The General Purpose Timer (GPT) is a 32-bit up-counter with clock source and frequency scaling options to provide a wide range of timing rates for various system applications. External hardware can send the GPT a "capture event" signal on an input pin to load the current timer value into a register for software to read. This capture signal can be configured to trigger the GPT on a rising or/and falling edge. Also, when the timer matches a preset value, the GPT can generate an interrupt for software and a "compare event" signal on output pins for external hardware.

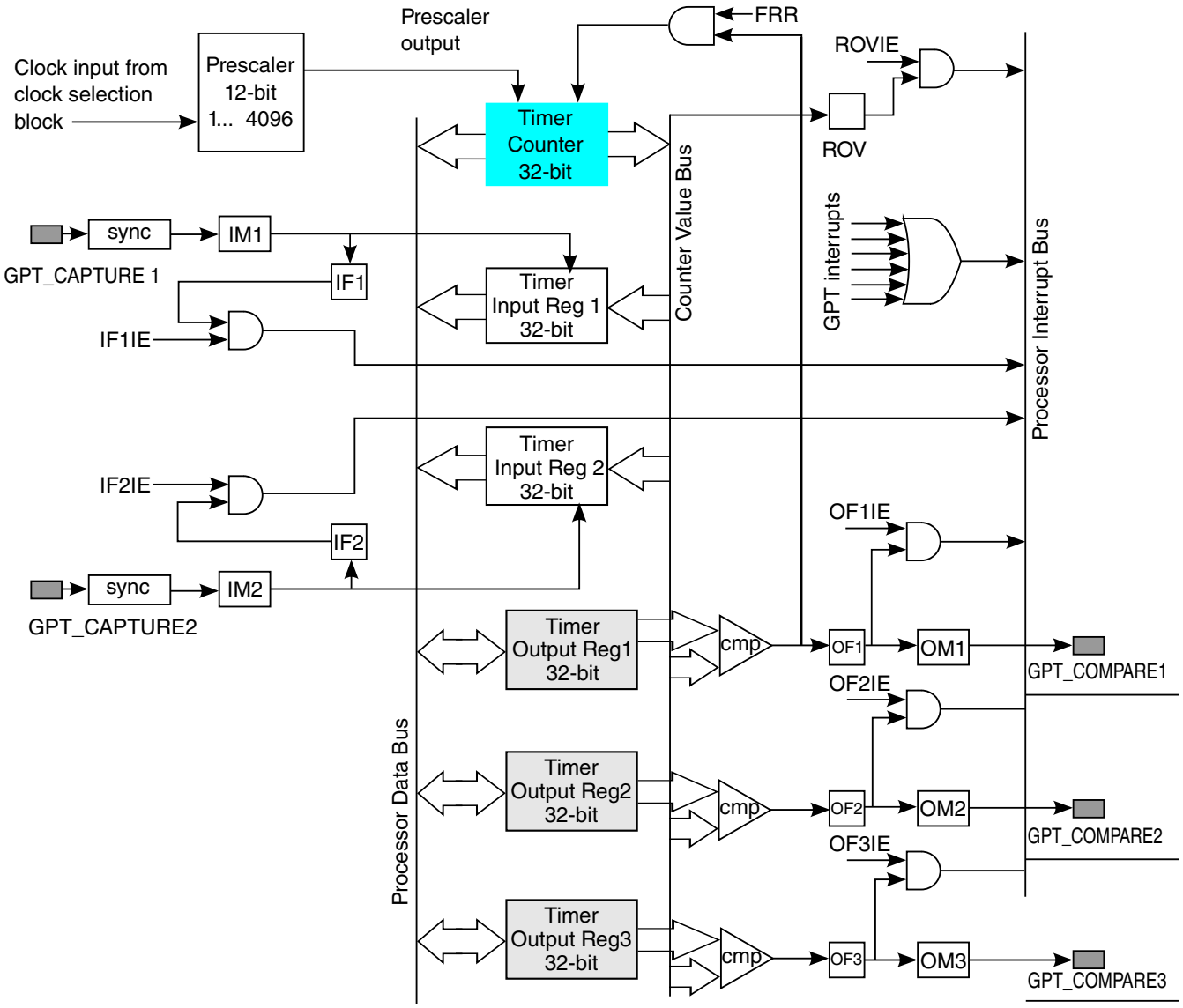


Figure 52-1. GPT Block Diagram

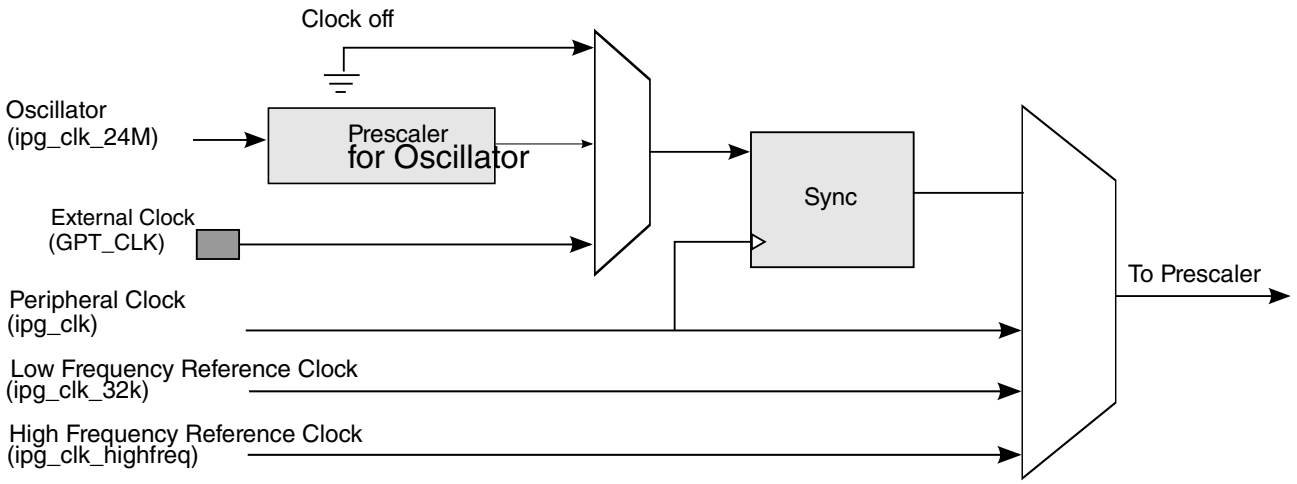


Figure 52-2. GPT Counter Clocks Diagram

52.2.1 Features

- One 32-bit up-counter with clock source selection, including external clock.
- Two input capture channels with a programmable trigger edge.
- Three output compare channels with a programmable output mode. A "forced compare" feature is also available.
- Can be programmed to be *active* in low power and debug modes.
- Interrupt generation at capture, compare, and rollover events.
- Restart or free-run modes for counter operations.

52.3 External Signals

The GPT follows the IP Bus protocol for interfacing with the processor core. The GPT does not have *any interface signals with any other module inside the chip,* except for the clock and reset inputs (from the clock and reset controller module) and for the interrupt signals *to* the processor interrupt handler. There are functional and clock inputs, and functional output signals going outside the chip boundary.

The following table describes all block signals that connect off-chip. Please see IOMUX Controller for GPT pin mux assignments.

Table 52-2. Off-Chip Module Signals

Name	Direction	Function	Reset State	Pull-Up
GPT_CLK	I	Input pin for an external clock that the counter can be operated at.	-	Passive Hysteresis
GPT_CAPTURE1	I	Input pin for a capture event for Input Capture Channel 1.	-	Passive
GPT_CAPTURE2	I	Input pin for a capture event for Input Capture Channel 2.	-	Passive
GPT_COMPARE1	O	Output pin that indicates a "compare event" occurrence in Output Compare Channel 1.	0	Passive
GPT_COMPARE2	O	Output pin that indicates a "compare event" occurrence in Output Compare Channel 2.	0	Passive
GPT_COMPARE3	O	Output pin that indicates a "compare event" occurrence in Output Compare Channel 3.	0	Passive

52.3.1 External Clock Input

The GPT counter can be operated using an external clock from outside the device, and this is the input pin used for that purpose. The external clock input (GPT_CLK) is treated as asynchronous to the peripheral clock (ipg_clk). To ensure proper operations of GPT, the external clock input frequency should be less than 1/4 of frequency of the peripheral clock (ipg_clk). Hysteresis characteristics on this pad will be required because this is a clock input.

52.3.2 Input Capture Trigger Signals

The GPT counter value can be stored in a register, triggered by an event from *outside the device*. A positive / negative edge on these GPT_CAPTURE*n* signals can trigger this capture event. These signals are treated as asynchronous to the peripheral clock (ipg_clk). Only those transitions which occur *at least a single clock cycle* (the clock selected to run the counter) *after the previous recorded transition* are guaranteed to trigger a capture event.

52.3.3 Output Compare Signals

The GPT_COMPARE*n* signals indicate that an output compare event has occurred on the corresponding Output Compare Channel *n*; see Output Compare.

52.4 Clocks

The clock that is input to the prescaler can be selected from multiple clock sources. The following table describes the clock sources for GPT. See Clock Controller Module (CCM) for clock source connection, configuration and gating information.

Table 52-3. GPT Clocks

Clock name	Description
Peripheral Clock (ipg_clk)	This is the internal clock used to access the module registers. If the Peripheral Clock or the External Clock (GPT_CLK) is selected as the Clock Source (CLKSRC=001 or 011), then the Peripheral Clock will be ON in normal GPT operations. In Low Power modes, if the GPT is programmed to be disabled (STOPEN or WAITEN or DOZEN=0), then the Peripheral Clock (ipg_clk) can be switched OFF.
Low Reference Clock (ipg_clk_32k)	This low-frequency 32-kHz reference clock (provided by the CCM) is intended to be ON in Low Power mode when the Peripheral Clock (ipg_clk) is turned OFF, thereby enabling the GPT to be operated using the Low Reference Clock in Low Power mode.
High-Frequency Clock (ipg_clk_highfreq)	Provided by the Clock Controller Module (CCM), this high-frequency reference clock is intended to be ON in Normal Power mode when the Peripheral Clock (ipg_clk) is turned OFF, thereby enabling the GPT to be operated using the High Frequency Clock *in Normal Power mode*.
Oscillator Clock (ipg_clk_24M)	This 24-MHz oscillator reference clock is intended to be used against frequency changes of the Peripheral Clock (ipg_clk) to provide a more accurate timer clock for system operation. The CCM provides the 24 MHz Oscillator Clock *without* synchronizing it to the System Bus Clock (ahb_clk) in Normal functional mode. Synchronization is done in GPT module. Before synchronization, the 24 MHz Oscillator Clock is divided by a dedicated prescaler to make sure the clock frequency less than half of System Bus Clock (ahb_clk).
External Clock (GPT_CLK)	The External Clock comes from *outside* the device and can be selected to run the GPT counter. The External Clock frequency is limited to < 1/4 frequency of the Peripheral Clock (ipg_clk) for proper GPT operations. Note that in Low Power modes, *if* the Peripheral Clock (ipg_clk) is not available, then the External Clock *cannot be used* to run the counter.

The clock input source is configured using the clock source field (CLKSRC, in the GPT_CR control register). The clock input to the prescaler can be disabled by programming the CLKSRC bits (of the GPT_CR control register) to 000. **The CLKSRC field value should be changed** *only after disabling the GPT* (by setting the EN bit in the GPT_CR to 0).

The PRESCALER field selects the divide ratio of the input clock that drives the main counter. The prescaler can divide the input clock by a value (from 1 to 4096) and can be changed *at any time*. A change in the value of the PRESCALER field *immediately affects* the output clock frequency.

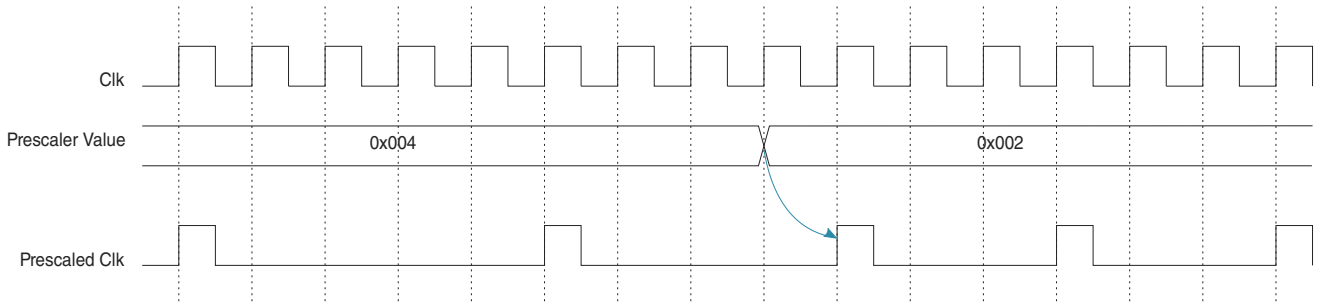


Figure 52-3. Prescaler Value Change Timing Diagram

52.5 Functional Description

This section provides a complete functional description of the GPT.

52.5.1 Operating Modes

The GPT counter can be programmed to work in either of two modes: Restart mode or Free-Run mode.

52.5.1.1 Restart Mode

In Restart mode (selectable through the GPT Control Register GPT_CR), when the counter reaches the compared value, the counter resets and starts again from 0x00000000. The Restart feature is associated only with Compare Channel 1.

Any write access to the Compare register of Channel 1 will reset the GPT counter. This is done to avoid possibly missing a compare event when compare value is changed from a higher value to lower value while counting is proceeding.

For the other two compare channels, when the compare event occurs the counter is *not reset*.

52.5.1.2 Free-Run Mode

In Free-Run mode, when compare events occur for all 3 channels, the counter is *not reset*; instead the counter continues to count until 0xffffffff, and then rolls over (to 0x00000000).

52.5.2 Operation

The General Purpose Timer (GPT) has a single counter (GPT_CNT) that is a 32-bit free-running *up-counter,* which starts counting *after it is enabled by software* (EN=1). The counter's clock source is the output of the prescaler labelled "Prescaler output" in Figure 52-1.

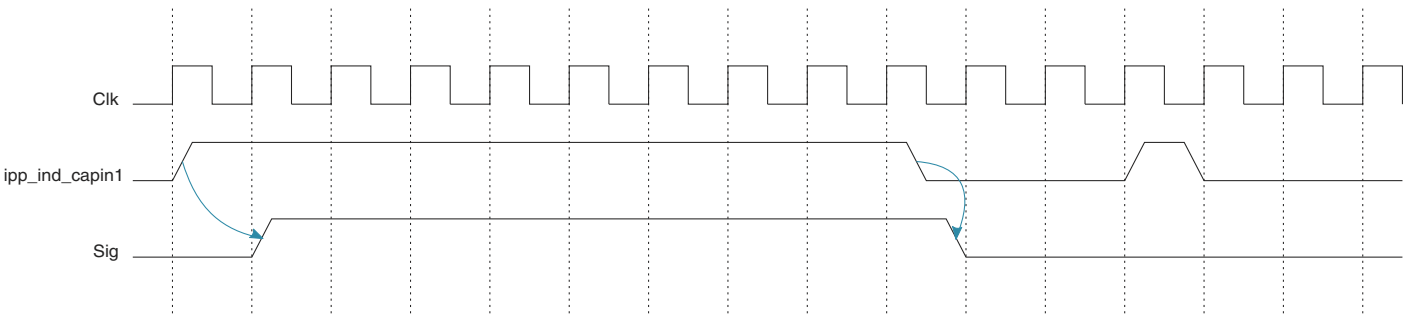
- If the GPT timer is disabled (EN=0), then the Main Counter *and* Prescaler Counter freeze their current count values. The ENMOD bit determines the value of the GPT counter when the EN bit is set and the Counter is enabled again.
  - If the ENMOD bit is set (=1), then the Main Counter and Prescaler Counter values are reset to 0, when GPT is enabled (EN=1).
  - If ENMOD bit is programmed to 0, then the Main Counter and Prescaler Counter restart counting from their frozen values, when GPT is enabled again (EN=1).
- If GPT is programmed to be disabled in a low power mode (STOP/WAIT), then the Main Counter and Prescaler Counter freeze at their current count values *when* GPT enters low power mode. When GPT exits a low power mode, the Main Counter and Prescaler Counter start counting from their frozen values *regardless* of the ENMOD bit value. Note that the GPT_CNT can be read *at any time* by the processor, and that *both* Input Capture Channels use the *same* counter (GPT_CNT).
- A hardware reset resets all the GPT registers to their respective reset values. All registers except the Output Compare Registers (OCR1, OCR2, OCR3) obtain a value of 0x0. The Compare registers are reset to 0xFFFF_FFFF.
- The software reset (SWR bit in the GPT_CR control register) resets *all* of the register bits *except* the EN, ENMOD, STOPEN, WAITEN, and DBGEN bits. The state of these bits is not affected by a software reset. Note that a software reset can be given *while the GPT is disabled.*

52.5.2.1 Input Capture

There are two Input Capture Channels, and each Input Capture Channel has a dedicated capture pin, capture register and input edge detection/selection logic. Each input capture function has an associated status flag, and can cause the processor to make an interrupt service request.

When a selected edge transition occurs on an Input Capture pin, the contents of the GPT_CNT is captured on the corresponding capture register and the appropriate interrupt status flag is set. An interrupt request can be generated when the transition is detected *if* its corresponding enable bit is set (in the Interrupt Register). The capture can be programmed to occur on the input pin's rising edge, falling edge, on both rising and

falling edges, or the capture can be disabled. The events are synchronized with the clock that was selected to run the counter. Only those transitions that occur at least one clock cycle (clock selected to run the counter) *after* the previous recorded transition will be guaranteed to trigger a capture event. There can be up to one clock cycle of uncertainty in the latching of the input transition. The Input Capture registers can be read *at any time* without affecting their values.



Clk = The clock selected from CLKSRC bit field setting

ipp_ind_capin1 = Pad signal for capture channel 1

Sig = Capture signal as sensed by the module

Figure 52-4. Input Capture Event Timing

52.5.2.2 Output Compare

The three Output Compare Channels *use the same counter* (GPT_CNT) as the Input Capture Channels. When the programmed content of an Output Compare register matches the value in GPT_CNT, an output compare status flag is set and an interrupt is generated (if the corresponding bit is set in the interrupt register). Consequently, the Output Compare timer pin is set, cleared, toggled, not affected at all or provide an active-low pulse for one input clock period according to the Output Compare Operating Mode bits in the Control Register (CR[OM*n*]).

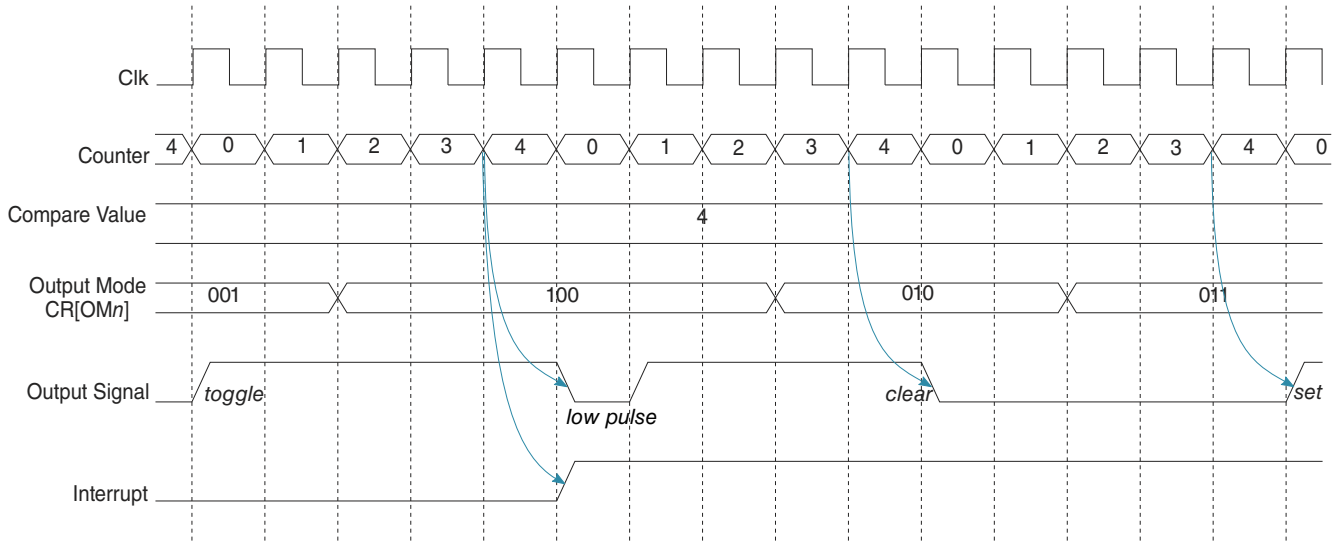


Figure 52-5. Output Compare and Interrupt Timing

There is also a "forced-compare" feature that allows the software to generate a compare event when required, *without the condition of the counter value that is equal to the compare value*. The action taken as a result of a forced compare is the same as when an output compare match occurs, *except that the status flags are not set and no interrupt can be generated*. Forced channels take programmed action immediately after the write to the force-compare bits. These bits are self-negating and always read as zeros.

52.5.2.3 Interrupts

There are 6 different interrupts that are generated by the GPT. If the selected clock for running the counter is available, then *all interrupts can be generated in Low Power and Debug modes*.

- Rollover Interrupt

  The Rollover Interrupt is generated when the GPT counter reaches 0xffffffff, then resets to 0x00000000 and continues counting. The Rollover Interrupt is enabled by the ROVIE bit in the GPT_IR register; the associated status bit is the ROV bit in the GPT_SR register.

- Input Capture Interrupt 1, 2

  After a capture event occurs, the associated Input Capture Channel generates an interrupt. The "capture event" interrupts are enabled by the IF2IE and IF1IE bits (in the GPT_IR register); the associated status bits are IF2 and IF1 (in the GPT_SR register). The capture of the counter value because of a capture event is *not affected*

by a pending capture interrupt. The Capture register is updated with a new counter value when a capture event occurs, regardless of whether that Capture Channels' interrupt has been serviced or not.

- Output Compare Interrupt 1, 2, 3

  After a compare event occurs, the associated Output Compare Channel generates an interrupt. The "compare event" interrupts are enabled by the OF3IE, OF2IE, and OF1IE bits (in the GPT_IR register); the associated status bits are OF3, OF2, and OF1 (in the GPT_SR register). A "forced compare" does not generate an interrupt.

A *cumulative* interrupt line is also present, which is asserted whenever any of the above interrupts are posted. The cumulative interrupt line has *no* associated enables or status bits.

52.5.2.4 Low Power Mode Behavior

In Low Power modes, if the clock from the selected clock source is available (except for the External Clock (GPT_CLK), which can be used *only if* the Peripheral Clock (ipg_clk) is available), the counter will continue to run depending on whether the control bit for that mode is set. If the clock is not present or if the corresponding low power bit in the GPT_CR control register is 0, the Main Counter and the Prescaler Counter freeze at their current values and resume counting (from their frozen values) when the Low Power mode is exited.

52.5.2.5 Debug Mode Behavior

In Debug mode, the modules in the device have the option of continuing to run or be halted.

- If the DBGEN bit is set, then the GPT timer will continue to run in Debug mode.
- If the DBGEN bit is not set (in the GPT_CR control register), then the GPT timer is halted.

52.6 Initialization/ Application Information

52.6.1 Selecting the Clock Source

The Clock Source field in the Control Register (CR[CLKSRC]) selects the clock source. CLKSRC should be changed only after disabling the GPT (CR[EN]=0). To change the clock source, follow this sequence:

1. Disable the GPT: CR[EN]=0
2. Disable GPT interrupts: Clear the Interrupt Register (IR)
3. Configure the Output Mode to unconnected/ disconnected for all channels: CR[OM*n*]=0
4. Disable the Input Capture Modes: CR[IM*n*]=0
5. Change the clock source CR[CLKSRC] to the desired value.
6. Assert a GPT software reset: CR[SWR]=1
7. Clear the flags in the GPT Status Register (SR) by writing 1s to them.
8. Reset the GPT counter to zero: CR[ENMOD]=1
9. Enable the GPT: CR[EN]=1
10. Enable GPT interrupts: For the desired interrupts, write 1s to their enable bits in IR.

52.7 Memory map and register definitions

52.7.1 GPT register descriptions

The GPT has 10 user-accessible 32-bit registers used to configure, operate and monitor the GPT.

A write access to the GPT Status Registers (Read-only registers ICR1, ICR2, CNT) generates a bus exception.

52.7.1.1 GPT memory map

GPT1 base address: 401E_C000h

GPT2 base address: 401F_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	GPT Control Register (CR)	32	RW	0000_0000h
4h	GPT Prescaler Register (PR)	32	RW	0000_0000h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
8h	GPT Status Register (SR)	32	W1C	0000_0000h
Ch	GPT Interrupt Register (IR)	32	RW	0000_0000h
10h - 18h	GPT Output Compare Register (OCR1 - OCR3)	32	RW	FFFF_FFFFh
1Ch - 20h	GPT Input Capture Register (ICR1 - ICR2)	32	RO	0000_0000h
24h	GPT Counter Register (CNT)	32	RO	0000_0000h

52.7.1.2 GPT Control Register (CR)

52.7.1.2.1 Offset

Register	Offset
CR	0h

52.7.1.2.2 Function

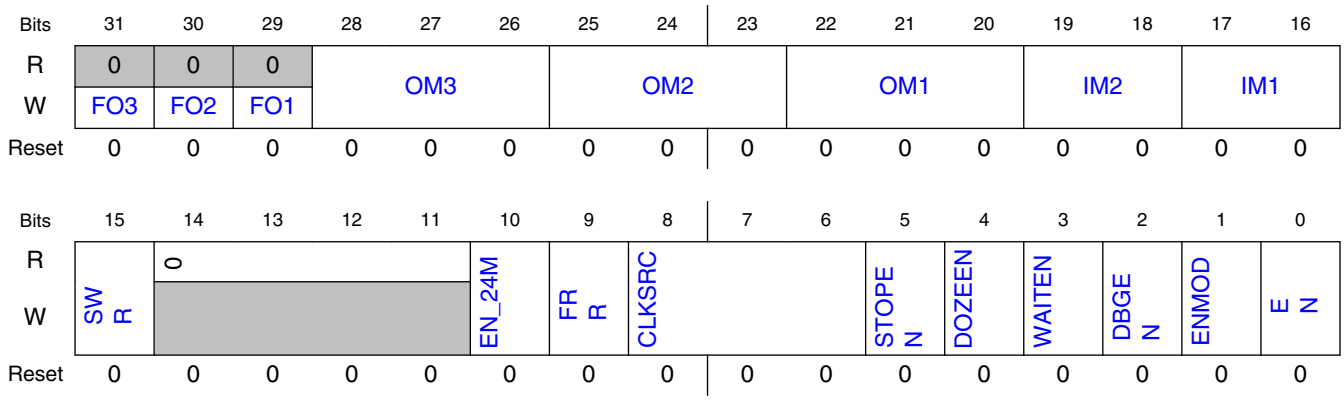
Contains configuration options for GPT operations, such as selecting the clock source for the counter and selecting a channel's output pin response when a compare event occurs. The CR register also includes the control bit (SWR) to trigger a software reset.

NOTE

Setting the SWR bit resets **all of the GPT registers** to their default reset values except for the EN, ENMOD, STOPEN, DOZEEN, WAITEN, and DBGEN bits in this register.

For applications where precise timing is critical, be aware that writes to the CR register take effect after one cycle of the module's register bus clock (1 wait state of latency). Reads to the register occur immediately (0 wait states).

52.7.1.2.3 Diagram



52.7.1.2.4 Fields

Field	Function
31-29 FOn	Force Output Compare for Channel n By setting the FO*n* bit, software can force the timer Output Compare *n* pin to respond according to the pin action programmed in the OM*n* bits (in this register). This bit is self-clearing and always reads as zero. **NOTE:** The OF*n* flag in the Status register (SR) is not set. 0b - No effect 1b - Trigger the programmed response on the pin
28-26: OM3 25-23: OM2 22-20: OM1	Output Compare Operating Mode for Channel n Selects the response on the output pin when a compare event occurs. 000b - Output disabled. No response on pin. 001b - Toggle output pin 010b - Clear output pin 011b - Set output pin 1xxb - Generate a low pulse that is one input clock cycle wide on the output pin. When OMn is first programmed as 1xx, the output pin is set to one immediately on the next input clock (if it was not one already). "Input clock" here refers to the clock selected by the CLKSRC field of this register.
19-18: IM2 17-16: IM1	Input Capture Operating Mode for Channel n Selects what transition on the input pin triggers a capture event. 00b - Capture disabled 01b - Capture on rising edge only 10b - Capture on falling edge only 11b - Capture on both edges
15 SWR	Software Reset Setting SWR resets **all of the registers** to their default reset values except for the DOZEEN, EN, ENMOD, STOPEN, WAITEN, and DBGEN bits in this register. The SWR bit remains set while the module is resetting. The SWR bit automatically clears when the reset procedure finishes. 0b - GPT is not in software reset state 1b - GPT is in software reset state
14-11 —	Reserved Software should write 0s to this reserved field. This field always reads as zero.

Table continues on the next page...

Field	Function
10 EN_24M	Enable Oscillator Clock Input Enables the 24-MHz clock input from an oscillator. 0b - Disable 1b - Enable
9 FRR	Free-Run or Restart Mode Selects the behavior of the GPT when a compare event in channel 1 occurs. 0b - Restart mode. After a compare event, the counter resets to 0x0000_0000 and resumes counting. 1b - Free-Run mode. After a compare event, the counter continues counting until 0xFFFF_FFFF and then rolls over to 0.
8-6 CLKSRC	Clock Source Select Selects which clock goes to the prescaler and subsequently runs the GPT counter. **NOTE:** The CLKSRC field value should be changed only after disabling the GPT (EN=0). For other programming requirements when changing the clock source, see section Selecting the Clock Source. 000b - No clock 001b - Peripheral Clock (ipg_clk) 010b - High Frequency Reference Clock (ipg_clk_highfreq) 011b - External Clock 100b - Low Frequency Reference Clock (ipg_clk_32k) 101b - Oscillator as Reference Clock (ipg_clk_24M)
5 STOPEN	GPT Stop Mode Enable Enables GPT operation *during Stop mode*. 0b - Disable in Stop mode 1b - Enable in Stop mode
4 DOZEEN	GPT Doze Mode Enable Enables GPT operation *during Doze mode*. 0b - Disable in Doze mode 1b - Enable in Doze mode
3 WAITEN	GPT Wait Mode Enable Enables GPT operation *during Wait mode*. 0b - Disable in Wait mode 1b - Enable in Wait mode
2 DBGEN	GPT Debug Mode Enable Enables GPT operation *during Debug mode*. 0b - Disable in Debug mode 1b - Enable in Debug mode
1 ENMOD	GPT Enable Mode When the GPT is disabled (EN=0), both the Main Counter and Prescaler Counter *freeze their current count values*. The ENMOD bit selects the value of the Main Counter and Prescaler Counter when GPT is enabled again (EN=1). **NOTE:** If GPT is programmed to be disabled in a low power mode (STOP/WAIT), the Main Counter and Prescaler Counter *freeze at their current count values* when the GPT enters low power mode. When GPT exits low power mode, the Main Counter and Prescaler Counter start counting from their frozen values, regardless of the ENMOD bit value.

Table continues on the next page...

Field	Function
	NOTE: A software reset (SWR=1) clears the Main Counter and Prescaler Counter values, regardless of the value of EN or ENMOD bits. 0b - Restart counting from their frozen values after GPT is enabled (EN=1). 1b - Reset counting from 0 after GPT is enabled (EN=1).
0 EN	GPT Enable Enables the GPT module. **NOTE:** Before setting the EN bit, *all registers should be properly programmed*. 0b - Disable 1b - Enable

52.7.1.3 GPT Prescaler Register (PR)

52.7.1.3.1 Offset

Register	Offset
PR	4h

52.7.1.3.2 Function

Contains the *divide value* of the clock that runs the counter.

52.7.1.3.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PRESCALER24M				PRESCALER											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

52.7.1.3.4 Fields

Field	Function
31-16	Reserved

Table continues on the next page...

Field	Function
—	Software should write 0s to these reserved bits. These bits always read as zero.
15-12 PRESCALER24M	Prescaler divide value for the oscillator clock The 24-MHz oscillator clock is divided by [PRESCALER24M + 1] before selected by the CLKSRC field. If the 24-MHz oscillator clock is not selected, this field takes no effect. 0000b - Divide by 1 0001b - Divide by 2 1111b - Divide by 16
11-0 PRESCALER	Prescaler divide value The clock selected by the CLKSRC field is divided by [PRESCALER + 1], and then used to run the counter. **NOTE:** A change in the value of PRESCALER causes the Prescaler counter to reset and a new count period to start immediately. See Clocks for an example timing diagram. 000000000000b - Divide by 1 000000000001b - Divide by 2 111111111111b - Divide by 4096

52.7.1.4 GPT Status Register (SR)

52.7.1.4.1 Offset

Register	Offset
SR	8h

52.7.1.4.2 Function

Contains flags to indicate that a counter has rolled over, and if any event has occurred on the Input Capture and Output Compare channels. The flags are cleared by writing a 1 to them.

52.7.1.4.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0										ROV	IF2	IF1	OF3	OF2	OF1
W											W1C	W1C	W1C	W1C	W1C	W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

52.7.1.4.4 Fields

Field	Function
31-6 —	Reserved Software should write 0s to these reserved bits. These bits always read as zero.
5 ROV	Rollover Flag Indicates that the counter has reached its *maximum possible value* and *rolled over* to 0 (from which the counter continues counting). ROV is set only when the counter has reached 0xFFFFFFFF in both Restart and Free-Run modes. 0b - Rollover has not occurred. 1b - Rollover has occurred.
4-3 IFn	Input Capture Flag for Channel n Indicates that a capture event has occurred on Input Capture channel *n*. 0b - Capture event has not occurred. 1b - Capture event has occurred.
2-0 OFn	Output Compare Flag for Channel n Indicates that a compare event has occurred on Output Compare channel *n*. 0b - Compare event has not occurred. 1b - Compare event has occurred.

52.7.1.5 GPT Interrupt Register (IR)

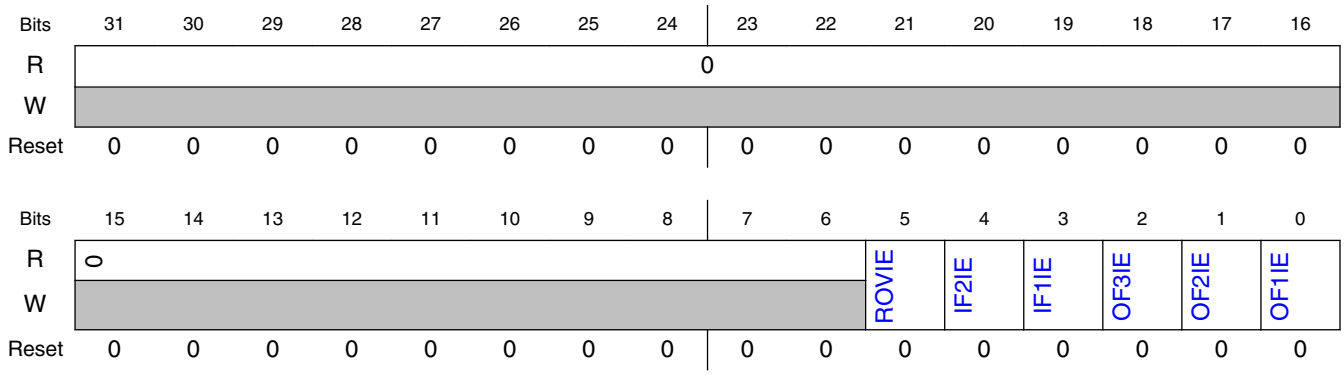
52.7.1.5.1 Offset

Register	Offset
IR	Ch

52.7.1.5.2 Function

Contains enable bits that control whether interrupts are generated after rollover, input capture and output compare events.

52.7.1.5.3 Diagram



52.7.1.5.4 Fields

Field	Function
31-6 —	Reserved • Writing a value to these reserved bits does not affect GPT operations. • These reserved bits are always read as zero.
5 ROVIE	Rollover Interrupt Enable Enables the Rollover interrupt. 0b - Disable 1b - Enable
4-3 IFnIE	Input Capture Flag for Channel n Interrupt Enable Enables the Input Capture Flag for Channel *n* interrupt. 0b - Disable 1b - Enable
2-0 OFnIE	Output Compare Flag for Channel n Interrupt Enable Enables the Output Compare Flag for Channel *n* interrupt. 0b - Disable 1b - Enable

52.7.1.6 GPT Output Compare Register (OCR1 - OCR3)

52.7.1.6.1 Offset

Register	Offset
OCR1	10h
OCR2	14h
OCR3	18h

52.7.1.6.2 Function

Holds the value that determines when a compare event is generated on the corresponding Output Compare Channel *n*.

NOTE

The behavior of OCR1 is different compared with OCR2 and OCR3:

- A write access to OCR1 while in Restart mode (CR[FRR]=0) resets the GPT counter.
- Writes to OCR1 take effect after one cycle of the module's register bus clock (1 wait state of latency). Reads to the register occur immediately (0 wait states).

52.7.1.6.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	COMP															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	COMP															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

52.7.1.6.4 Fields

Field	Function
31-0 COMP	Compare Value When the counter value equals COMP, a compare event is generated.

52.7.1.7 GPT Input Capture Register (ICR1 - ICR2)

52.7.1.7.1 Offset

Register	Offset
ICR1	1Ch
ICR2	20h

52.7.1.7.2 Function

Holds the value that was in the counter during the last capture event on the corresponding Input Capture Channel *n*.

52.7.1.7.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	CAPT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CAPT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

52.7.1.7.4 Fields

Field	Function
31-0 CAPT	Capture Value After a capture event on Input Capture Channel *n* occurs, the current value of the counter is loaded into the corresponding ICR*n*[CAPT].

52.7.1.8 GPT Counter Register (CNT)

52.7.1.8.1 Offset

Register	Offset
CNT	24h

52.7.1.8.2 Function

Contains the current value of the main counter. CNT can be read at any time *without affecting the counting process* of the GPT.

52.7.1.8.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	COUNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	COUNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

52.7.1.8.4 Fields

Field	Function
31-0 COUNT	Counter Value Provides the current value of the GPT counter.

Chapter 53
Periodic Interrupt Timer (PIT)

53.1 Chip-specific PIT information

Table 53-1. Reference links to related information

Topic	Related module(s)	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
Interrupts, DMA Events and XBAR Assignments	-	Interrupts, DMA Events and XBAR Assignments
PIT channel to DMA channel correspondence	-	PIT Channel Assignments For Periodic DMA Triggering

There are 4 PIT channels in this device.

53.2 Introduction

PIT includes an array of timers that can be used to raise interrupts and trigger DMA channels. Some of the registers listed in PIT register descriptions are used to configure these timers, and after the expiry of a timer, the respective registers indicate a timeout status, which can be used to raise interrupts and triggers.

53.2.1 Block diagram

The following figure shows the block diagram of PIT module.

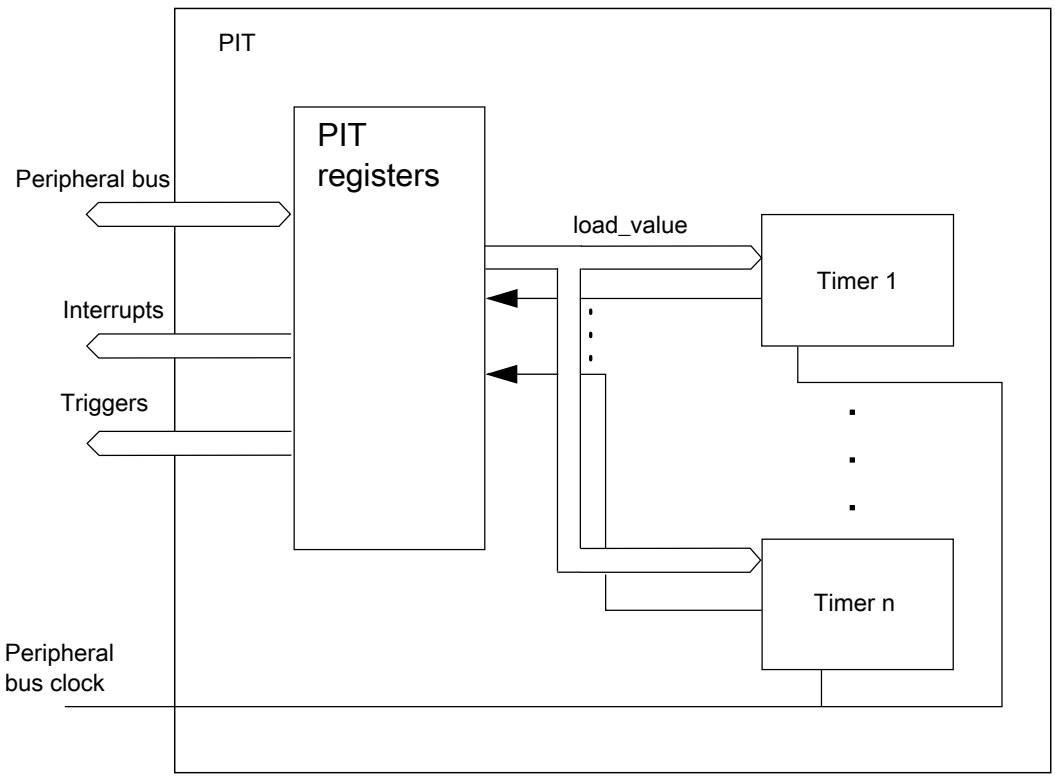


Figure 53-1. Block diagram of PIT

53.2.2 Features

The key features of the module are:

- Ability of timers to generate DMA trigger pulses
- Ability of timers to generate interrupts
- Maskable interrupts
- Independent timeout periods for each timer
- All the timers use a downcounter

53.3 Modes of operation

This subsection briefly describes all operating modes supported by PIT.

- Run mode

  All functional parts of the PIT are running during normal Run mode.

53.4 Functional description

This section provides the functional description of the module.

53.4.1 General operation

This section provides detailed information on the internal operation of the module. Each timer can be used to generate trigger pulses and interrupts, and each interrupt is available on a separate interrupt line.

53.4.1.1 Timers

The timers generate triggers at periodic intervals, when enabled. The timers load the start values as specified in their LDVAL registers, count down to 0 and then load the respective start values again. Each time a timer reaches 0, it generates a trigger pulse and sets the interrupt flag.

All interrupts can be enabled or masked by setting TCTRLn[TIE]. A new interrupt can be generated only after the previous one is cleared.

If desired, the current counter value of the timer can be read via the CVAL registers.

The counter period can be restarted by first disabling and then enabling the timer with TCTRLn[TEN]. See the following figure.

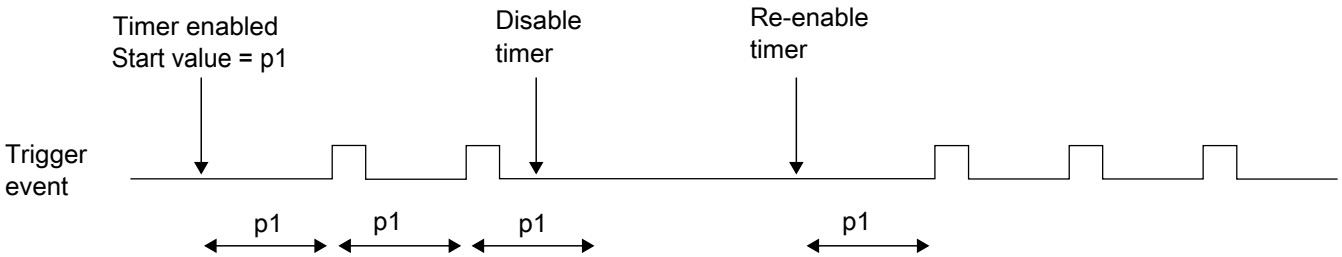


Figure 53-2. Stopping and starting a timer

The counter period of a running timer can be modified by first disabling the timer, setting a new load value, and then enabling the timer again. See the following figure.

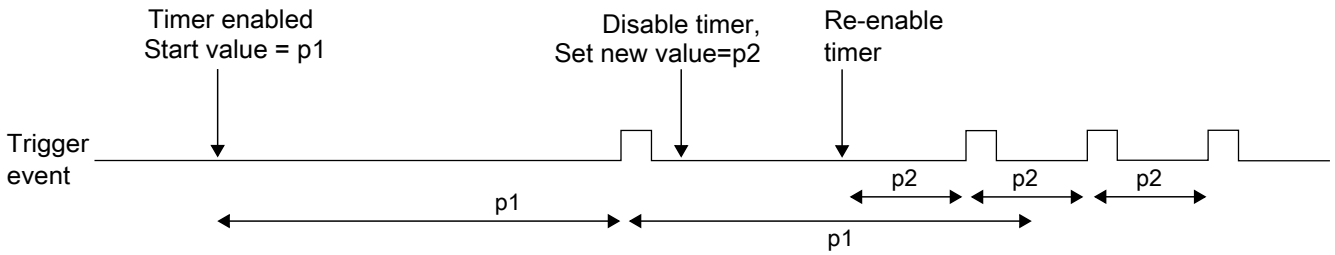


Figure 53-3. Modifying running timer period

It is also possible to change the counter period without restarting the timer, by writing LDVAL with the new load value. This value then loads after the next trigger event. See the following figure.

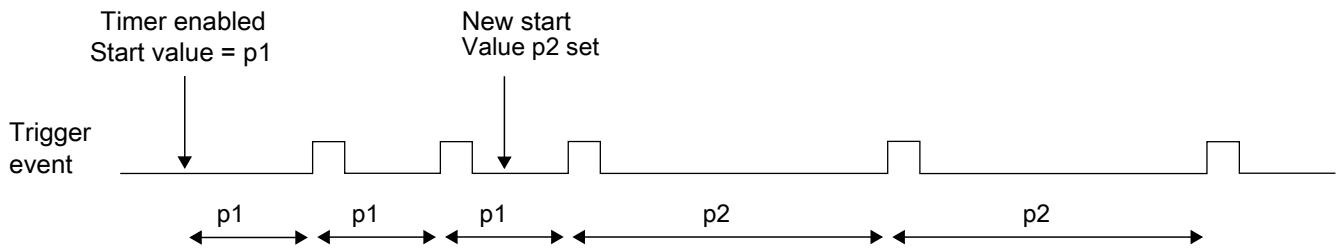


Figure 53-4. Dynamically setting a new load value

NOTE

- If MCR[FRZ] is written as 1 and when the timer is nearing 0 (CVALn =0x0), the debug mode command may not make it to the IP before the timer expires and generates a trigger.

- Debug Mode will be ignored if CVALn =0x0, but the trigger will remain asserted until the mode is removed. The user is recommended to exit Debug mode and then clear the interrupt TFLGn[TIF].
- If the timers are to be freezed for debug, sufficient time must be ensured for the IP to react.

53.4.1.2 Debug mode

In the Debug mode, the timers are frozen based on MCR[FRZ]. This is intended to aid software development, allowing the developer to halt the processor, investigate the current state of the system, for example, the timer values, and then continue the operation.

53.4.2 Interrupts

All the timers support interrupt generation.

Timer interrupts can be enabled by setting TCTRLn[TIE].

TFLGn[TIF] are set to 1 when a timeout occurs on the associated timer, and are cleared to 0 by writing a 1 to the corresponding TFLGn[TIF].

53.4.3 Chained timers

When a timer has chain mode enabled, it counts after the previous timer has expired. So if timer n-1 counts down to 0, counter n decrements the value by one. This allows to chain some of the timers together to form a longer timer. The first timer (timer 0) cannot be chained to any other timer. See Example configuration for chained timers for details.

53.4.4 Example configuration for chained timers

In the example configuration:

- The PIT clock has a frequency of 100 MHz.

- Timers 1 and 2 are available.
- An interrupt is raised every 1 minute.

The PIT module needs to be activated by writing a 0 to MCR[MDIS].

The 100 MHz clock frequency equates to a clock period of 10 ns, so the PIT needs to count for 6000 million cycles, which is more than a single timer can do. So, Timer 1 is set up to trigger every 6 s (600 million cycles). Timer 2 is chained to Timer 1 and programmed to trigger 10 times.

The value for the LDVAL register trigger is calculated as number of cycles-1, so LDVAL1 receives the value 0x23C345FF and LDVAL2 receives the value 0x00000009.

The interrupt for Timer 2 is enabled by setting TCTRL2[TIE], the Chain mode is activated by setting TCTRL2[CHN], and the timer is started by writing a 1 to TCTRL2[TEN].

TCTRL1[TEN] needs to be set, and TCTRL1[CHN] and TCTRL1[TIE] are cleared.

The following example code matches the described setup:

```
// turn on PIT
PIT_MCR = 0x00;


// Timer 2
PIT_LDVAL2 = 0x00000009; // setup Timer 2 for 10 counts
PIT_TCTRL2 = TIE; // enable Timer 2 interrupt
PIT_TCTRL2 |= CHN; // chain Timer 2 to Timer 1
PIT_TCTRL2 |= TEN; // start Timer 2


// Timer 1
PIT_LDVAL1 = 0x23C345FF; // setup Timer 1 for 600 000 000 cycles
PIT_TCTRL1 = TEN; // start Timer 1
```

53.4.5 Example configuration for the Lifetime Timer

To configure the Lifetime Timer, channels 0 and 1 need to be chained together.

First, the PIT module needs to be activated by writing a 0 to the MDIS bit in the CTRL register, and then the LDVAL registers need to be set to the maximum value.

The following example code matches the described setup:

```
// turn on PIT
PIT_MCR = 0x00;


// Timer 1
```

```
PIT_LDVAL1 = 0xFFFFFFFF; // setup timer 1 for maximum counting period
PIT_TCTRL1 = 0x0; // disable timer 1 interrupts
PIT_TCTRL1 |= CHN; // chain timer 1 to timer 0
PIT_TCTRL1 |= TEN; // start timer 1


// Timer 0
PIT_LDVAL0 = 0xFFFFFFFF; // setup timer 0 for maximum counting period
PIT_TCTRL0 = TEN; // start timer 0
```

To access the Lifetime Timer, read first LTMR64H and then LTMR64L.

```
current_uptime = PIT_LTMR64H<<32;
current_uptime = current_uptime + PIT_LTMR64L;
```

53.5 Initialization and application information

In the example configuration:

- The PIT clock has a frequency of 50 MHz.
- Timer 1 creates an interrupt every 5.12 ms.
- Timer 3 creates a trigger event every 30 ms.

The PIT module must be activated by writing a 0 to MCR[MDIS].

The 50 MHz clock frequency equates to a clock period of 20 ns. Timer 1 needs to trigger every 5.12 ms/20 ns = 256,000 cycles and Timer 3 every 30 ms/20 ns = 1,500,000 cycles. The value for the LDVAL register trigger is calculated as:

LDVAL trigger = (period / clock period) -1

This means LDVAL1 and LDVAL3 must be written with 0x0003E7FF and 0x0016E35F, respectively.

The interrupt for Timer 1 is enabled by setting TCTRL1[TIE]. The timer is started by writing 1 to TCTRL1[TEN].

Timer 3 shall be used only for triggering. Therefore, Timer 3 is started by writing a 1 to TCTRL3[TEN]. Also, TCTRL3[TIE] stays at 0.

The following example code matches the described setup:

```
// turn on PIT
PIT_MCR = 0x00;


// Timer 1
PIT_LDVAL1 = 0x0003E7FF; // setup timer 1 for 256000 cycles
PIT_TCTRL1 = TIE; // enable Timer 1 interrupts
PIT_TCTRL1 |= TEN; // start Timer 1
```

```
// Timer 3
PIT_LDVAL3 = 0x0016E35F; // setup timer 3 for 1500000 cycles
PIT_TCTRL3 |= TEN; // start Timer 3
```

53.6 PIT register descriptions

This section provides a detailed description of all registers accessible in the PIT module.

- See the chip-specific PIT information for the number of PIT channels used in this MCU.

53.6.1 PIT memory map

PIT base address: 4008_4000h

Offset	Register	Width (In bits)	Access	Reset value
0h	PIT Module Control Register (MCR)	32	RW	0000_0002h
E0h	PIT Upper Lifetime Timer Register (LTMR64H)	32	RO	0000_0000h
E4h	PIT Lower Lifetime Timer Register (LTMR64L)	32	RO	0000_0000h
100h	Timer Load Value Register (LDVAL0)	32	RW	0000_0000h
104h	Current Timer Value Register (CVAL0)	32	RO	0000_0000h
108h	Timer Control Register (TCTRL0)	32	RW	0000_0000h
10Ch	Timer Flag Register (TFLG0)	32	W1C	0000_0000h
110h	Timer Load Value Register (LDVAL1)	32	RW	0000_0000h
114h	Current Timer Value Register (CVAL1)	32	RO	0000_0000h
118h	Timer Control Register (TCTRL1)	32	RW	0000_0000h
11Ch	Timer Flag Register (TFLG1)	32	W1C	0000_0000h
120h	Timer Load Value Register (LDVAL2)	32	RW	0000_0000h
124h	Current Timer Value Register (CVAL2)	32	RO	0000_0000h
128h	Timer Control Register (TCTRL2)	32	RW	0000_0000h
12Ch	Timer Flag Register (TFLG2)	32	W1C	0000_0000h
130h	Timer Load Value Register (LDVAL3)	32	RW	0000_0000h
134h	Current Timer Value Register (CVAL3)	32	RO	0000_0000h
138h	Timer Control Register (TCTRL3)	32	RW	0000_0000h
13Ch	Timer Flag Register (TFLG3)	32	W1C	0000_0000h

53.6.2 PIT Module Control Register (MCR)

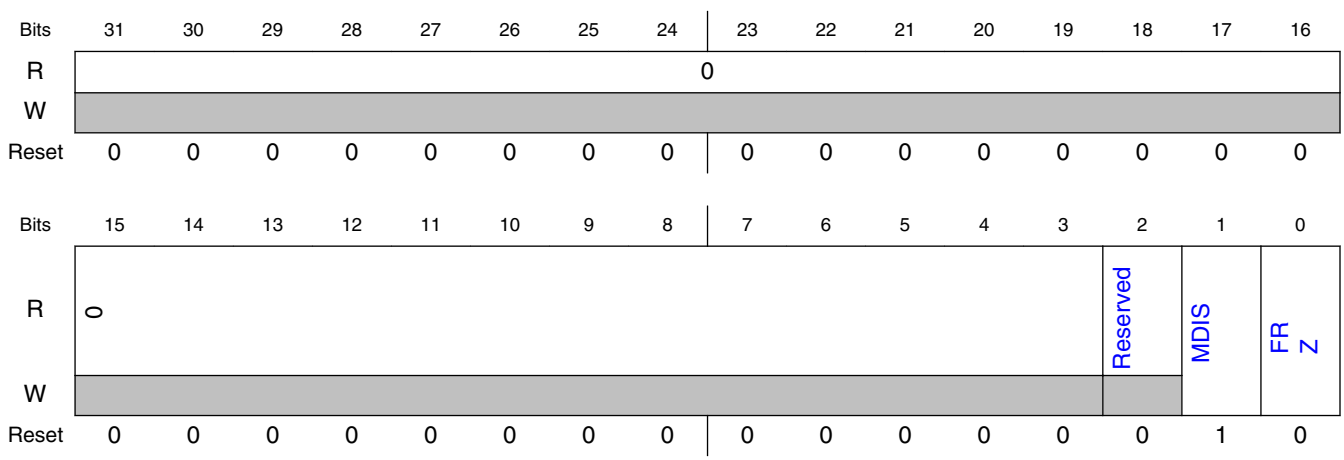
53.6.2.1 Offset

Register	Offset
MCR	0h

53.6.2.2 Function

This register enables or disables the PIT timer clocks and controls the timers when the PIT enters the Debug mode.

53.6.2.3 Diagram



53.6.2.4 Fields

Field	Function
31-3 —	Reserved
2 —	Reserved
1	Module Disable for PIT

Table continues on the next page...

Field	Function
MDIS	Disables the standard timers. You must write 1 to this field before setting up anything. 0b - Clock for standard PIT timers is enabled. 1b - Clock for standard PIT timers is disabled.
0 FRZ	Freeze Allows the timers to be stopped when the device enters the Debug mode. 0b - Timers continue to run in Debug mode. 1b - Timers are stopped in Debug mode.

53.6.3 PIT Upper Lifetime Timer Register (LTMR64H)

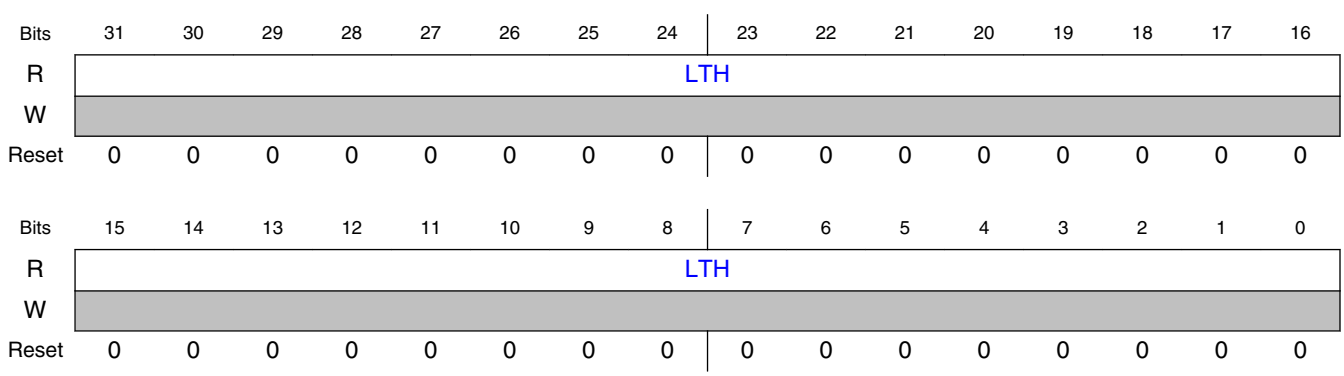
53.6.3.1 Offset

Register	Offset
LTMR64H	E0h

53.6.3.2 Function

This register is intended for applications that chain timer 0 and timer 1 to build a 64-bit Lifetime Timer.

53.6.3.3 Diagram



53.6.3.4 Fields

Field	Function
31-0 LTH	Life Timer value Shows the timer value of timer 1. If this register is read at a time t1, LTMR64L shows the value of timer 0 at time t1.

53.6.4 PIT Lower Lifetime Timer Register (LTMR64L)

53.6.4.1 Offset

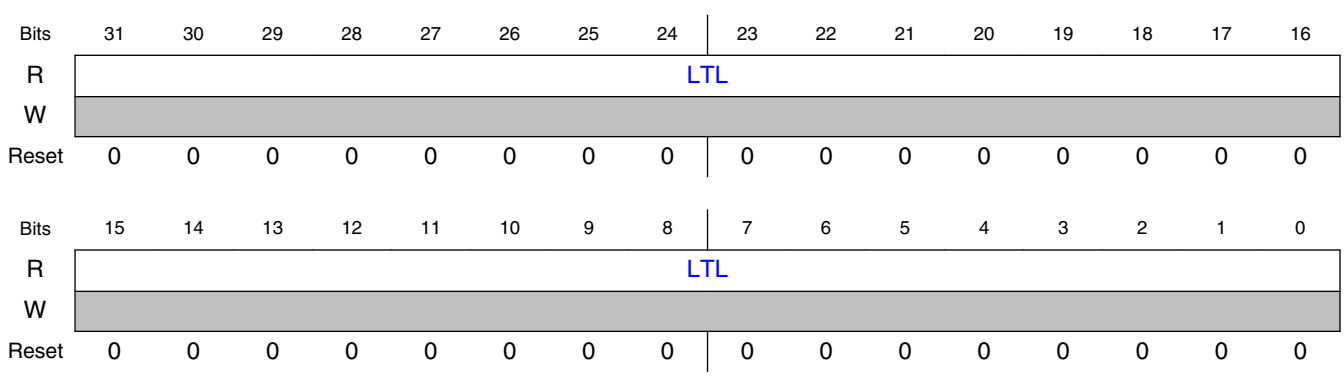
Register	Offset
LTMR64L	E4h

53.6.4.2 Function

This register is intended for applications that chain timer 0 and timer 1 to build a 64-bit Lifetime Timer.

To use LTMR64H and LTMR64L, timer 0 and timer 1 need to be chained. To obtain the correct value, first read LTMR64H and then LTMR64L. The value for the LTMR64H register is set to CVAL1 register at the time of the first access and the value of the LTMR64L register is set to CVAL0 register at first access. Therefore, the application is not affected by the carry-over effects of the running counter.

53.6.4.3 Diagram



53.6.4.4 Fields

Field	Function
31-0 LTL	Life Timer value Shows the value of timer 0 at the time LTMR64H was last read. It will only update if LTMR64H is read.

53.6.5 Timer Load Value Register (LDVAL0 - LDVAL3)

53.6.5.1 Offset

Register	Offset
LDVAL0	100h
LDVAL1	110h
LDVAL2	120h
LDVAL3	130h

53.6.5.2 Function

These registers select the timeout period for the timer interrupts.

53.6.5.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	TSV															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TSV															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

53.6.5.4 Fields

Field	Function
31-0 TSV	Timer Start Value Sets the timer start value. The timer counts down until it reaches 0, then generates an interrupt and loads this register value again. Writing a new value to this register does not restart the timer; instead the value is loaded after the timer expires. To abort the current cycle and start a timer period with the new value, the timer must be disabled and enabled again.

53.6.6 Current Timer Value Register (CVAL0 - CVAL3)

53.6.6.1 Offset

Register	Offset
CVAL0	104h
CVAL1	114h
CVAL2	124h
CVAL3	134h

53.6.6.2 Function

These registers indicate the current timer position.

53.6.6.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TVL															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	TVL															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

53.6.6.4 Fields

Field	Function
31-0 TVL	Current Timer Value Represents the current timer value, if the timer is enabled. **NOTE:** • If the timer is disabled, do not use this field because its value is unreliable. • The timer uses a downcounter. The timer values are frozen in the Debug mode if MCR[FRZ] is set.

53.6.7 Timer Control Register (TCTRL0 - TCTRL3)

53.6.7.1 Offset

Register	Offset
TCTRL0	108h
TCTRL1	118h
TCTRL2	128h
TCTRL3	138h

53.6.7.2 Function

These registers contain the control bits for each timer.

53.6.7.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0													CHN	TIE	TEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

53.6.7.4 Fields

Field	Function
31-3 —	Reserved
2 CHN	Chain Mode When activated, timer n-1 needs to expire before timer n (n is > 0) can decrement by 1. Timer 0 cannot be chained. 0b - Timer is not chained. 1b - Timer is chained to a previous timer. For example, for channel 2, if this field is set, Timer 2 is chained to Timer 1.
1 TIE	Timer Interrupt Enable When an interrupt is pending, or if TFLGn[TIF] is set, enabling the interrupt causes an interrupt event. To avoid this, the associated TFLGn[TIF] must be cleared first. 0b - Interrupt requests from Timer n are disabled. 1b - Interrupt is requested whenever TIF is set.
0 TEN	Timer Enable Enables or disables the timer. 0b - Timer n is disabled. 1b - Timer n is enabled.

53.6.8 Timer Flag Register (TFLG0 - TFLG3)

53.6.8.1 Offset

Register	Offset
TFLG0	10Ch
TFLG1	11Ch
TFLG2	12Ch
TFLG3	13Ch

53.6.8.2 Function

These registers hold the PIT interrupt flags.

53.6.8.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0															TIF
W																W1C
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

53.6.8.4 Fields

Field	Function
31-1 —	Reserved
0 TIF	Timer Interrupt Flag Sets to 1 at the end of the timer period. Writing 1 to this flag clears it and writing 0 has no effect. If enabled, or, when TCTRLn[TIE] = 1, TIF causes an interrupt request. 0b - Timeout has not yet occurred. 1b - Timeout has occurred.

Chapter 54
Quad Timer (TMR)

54.1 Chip-specific TMR information

Table 54-1. Reference links to related information

Topic	Related module(s)	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
Interrupts, DMA Events and XBAR Assignments	-	Interrupts, DMA Events and XBAR Assignments

54.2 Overview

Each timer module (TMR) contains four identical counter/timer groups. Each 16-bit counter/timer group contains a prescaler, a counter, a load register, a hold register, a capture register, two compare registers, two status and control registers, and one control register. All of the registers except the prescaler are read/writable.

NOTE

This document uses the terms "Timer" and "Counter" interchangeably because the counter/timers may perform either or both tasks.

The load register provides the initialization value to the counter when the counter's terminal value has been reached.

The hold register captures the counter's value when other counters are being read. This feature supports the reading of cascaded counters.

The capture register enables an external signal to take a "snap shot" of the counter's current value.

The COMP1 and COMP2 registers provide the values to which the counter is compared. If a match occurs, the OFLAG (TMR Output signal) can be set, cleared, or toggled. At match time, an interrupt is generated if enabled, and the new compare value is loaded into the COMP1 or COMP2 registers from CMPLD1 and CMPLD2 if enabled.

The prescaler provides different time bases useful for clocking the counter/timer.

The counter provides the ability to count internal or external events.

Within a timer module (set of four timer/counters), the four inputs are shared by the four counters, but the four outputs are dedicated to each counter.

54.2.1 Block Diagram

Each of the timer/counter groups within the quad-timer are shown in this figure.

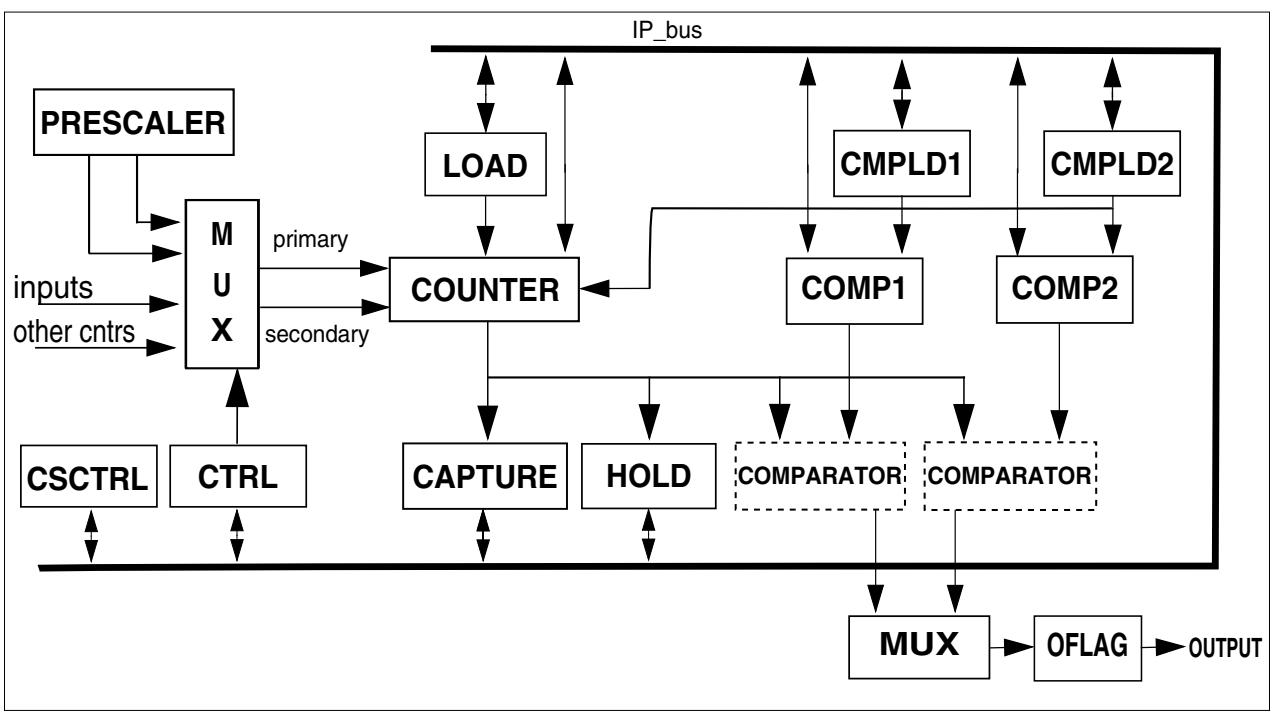


Figure 54-1. Quad Timer Block Diagram

54.2.2 Features

The TMR module design includes these distinctive features:

- Four 16-bit counters/timers
- Count up/down
- Counters are cascadable
- Programmable count modulo
- Max count rate equals BUS_CLOCK_ROOT/2 for external clocks
- Max count rate equals BUS_CLOCK_ROOT for internal clocks
- Count once or repeatedly
- Counters are preloadable
- Compare registers are preloadable (available with compare load feature)
- Counters can share available input pins
- Separate prescaler for each counter
- Each counter has capture and compare capability
- Programmable operation during debug mode
- Inputs may act as fault inputs
- Programmable input filter
- Counting start can be synchronized across counters

54.3 Modes of Operation

The TMR module design operates in only a single mode of operation: Functional Mode. The various counting modes are detailed in the Functional Description.

54.4 Functional Description

54.4.1 General

The counter/timer has two basic modes of operation: it can count internal or external events, or it can count an internal clock source while an external input signal is asserted, thus timing the width of the external input signal.

- The counter can count the rising, falling, or both edges of the selected input pin.
- The counter can decode and count quadrature encoded input signals.
- The counter can count up and down using dual inputs in a "count with direction" format.
- The counter's terminal count value (modulo) is programmable.
    - The value that is loaded into the counter after reaching its terminal count is programmable.
- The counter can count repeatedly, or it can stop after completing one count cycle.
- The counter can be programmed to count to a programmed value and then immediately reinitialize, or it can count through the compare value until the count "rolls over" to zero.

The external inputs to each counter/timer are shareable among each of the four counter/timers within the module. The external inputs can be used as:

- Count commands
- Timer commands
- They can trigger the current counter value to be "captured"
- They can be used to generate interrupt requests

The polarity of the external inputs are selectable.

The primary output of each timer/counter is the output signal OFLAG. The OFLAG output signal can be:

- Set, cleared, or toggled when the counter reaches the programmed value.
- The OFLAG output signal may be output to an external pin instead of having that pin serve as a timer input.
- The OFLAG output signal enables each counter to generate square waves, PWM, or pulse stream outputs.
- The polarity of the OFLAG output signal is selectable.

Any counter/timer can be assigned as a master. A master's compare signal can be broadcast to the other counter/timers within the module. The other counters can be configured to reinitialize their counters and/or force their OFLAG output signals to predetermined values when a master's counter/timer compare event occurs.

54.4.2 Usage of Compare Registers

The dual compare registers (COMP1 and COMP2) provide a bidirectional modulo count capability. The COMP1 register is used when the counter is *counting up*, and the COMP2 register is used when the counter is *counting down*. Alternating compare mode is the only exception.

The COMP1 register should be set to the desired maximum count value or FFFFh to indicate the maximum unsigned value prior to roll-over, and the COMP2 register should be set to the minimum count value or 0000h to indicate the minimum unsigned value prior to roll-under.

If CTRL[OUTMODE] is set to 100, the OFLAG will toggle while using alternating compare registers. In this variable frequency PWM mode, the COMP2 value defines the desired pulse width of the on time, and the COMP1 register defines the off time.

Use caution when changing COMP1 and COMP2 while the counter is active. If the counter has already passed the new value, it will count to FFFFh or 0000h, roll over, then begin counting toward the new value. The check is: Count=CMPx, *not* Count > COMP1 or Count < COMP2.

The use of the CMPLD1 and CMPLD2 registers to compare values will help to minimize this problem.

54.4.3 Usage of Compare Load Registers

The CMPLD1, CMPLD2, and CSCTRL registers offer a high degree of flexibility for loading compare registers with user-defined values on different compare events. To ensure correct functionality while using these registers we strongly suggest using the following method described in this section.

The purpose of the compare load feature is to allow quicker updating of the compare registers. In the past, a compare register could be updated using interrupts. However, because of the latency between an interrupt event occurring and the service of that interrupt, there was the possibility that the counter may have already counted past the

new compare value by the time the compare register was updated by the interrupt service routine. The counter would then continue counting until it rolled over and reached the new compare value.

To address this, the compare registers are now updated in hardware in the same way the counter register is re-initialized to the value stored in the load register. The compare load feature allows the user to calculate new compare values and store them in to the comparator load registers. When a compare event occurs, the new compare values in the comparator load registers are written to the compare registers eliminating the use of software to do this.

The compare load feature is intended to be used in variable frequency PWM mode. The COMP1 register determines the pulse width for the logic low part of OFLAG and COMP2 determines the pulse width for the logic high part of OFLAG. The period of the waveform is determined by the COMP1 and COMP2 values and the frequency of the primary clock source. See the following figure.

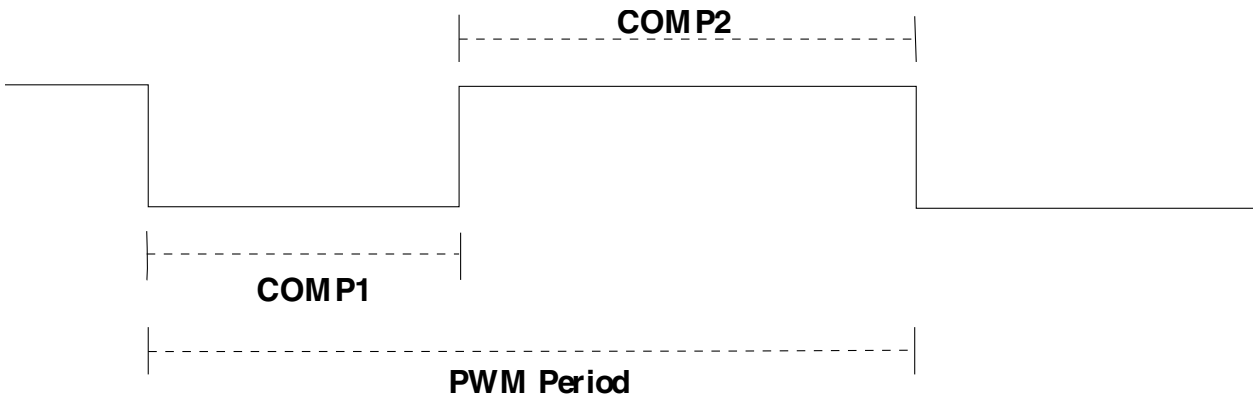


Figure 54-2. Variable PWM Waveform

Should we desire to update the duty cycle or period of the above waveform, we would need to update the COMP1 and COMP2 values using the compare load feature.

54.4.4 Usage of the Capture Register

The capture register stores a copy of the counter's value when an input edge (positive, negative, or both) is detected. After a capture event occurs, no further updating of the capture register will occur until the SCTRL[IEF] (input edge flag) is cleared by writing a zero to the SCTRL[IEF].

54.4.5 Functional Modes

The selected external count signals are sampled at the TMR's base clock rate and then run through a transition detector. The maximum count rate is one-half of the TMR's base clock rate. Internal clock sources can be used to clock the counters at the TMR's base clock rate.

If a counter is programmed to count to a specific value and then stop, the CTRL[CM] field is cleared when the count terminates.

54.4.5.1 Stop Mode

If CTRL[CM] is set to '000', the counter is inert. No counting will occur. Stop mode will also disable the interrupts caused by input transitions on a selected input pin.

54.4.5.2 Count Mode

If CTRL[CM] is set to '001', the counter will count the rising edges of the selected clock source. This mode is useful for generating periodic interrupts for timing purposes, or counting external events such as "widgets" on a conveyor belt passing a sensor. If the selected input is inverted by setting SCTRL[IPS] (input polarity select), then the negative edge of the selected external input signal is counted.

Example: 54.4.5.2.1 Count Pulses from External Source

```
//    This example uses TMR1 to count pulse (actually counts rising edges of the pulse)
//      from an external source (QT3).
//
void Pulse_Init(void)
{
  /* TMR1_CTRL: CM=0,PCS=3,SCS=0,ONCE=0,LENGTH=0,DIR=0,COINIT=0,OUTMODE=0 */
  setReg(TMR1_CTRL,0x0600);           /* Set up mode */
  /* TMR1_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                 Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
  setReg(TMR1_SCTRL,0x00);
  setReg(TMR1_CNTR,0x00);             /* Reset counter register */
  setReg(TMR1_LOAD,0x00);             /* Reset load register */
  setRegBitGroup(TMR1_CTRL,CM,0x01);  /* Run counter */
}
```

Example: 54.4.5.2.2 Generate Periodic Interrupt By Counting Internal Clocks

```
//   This example generates an interrupt every 100ms,
//     assuming the chip is operating at 60 MHz.
//
```

```
// It does this by using the IP_bus_clk divided by 128 as the counter clock source.
// The counter then counts to 46874 where it matches the COMP1 value.
// At that time an interrupt is generated, the counter is reloaded and
// the next COMP1 value is loaded from CMPLD1.
//
void TimerInt_Init(void)
{
  /* TMR0_CTRL: CM=0,PCS=0,SCS=0,ONCE=0,LENGTH=1,DIR=0,COINIT=0,OUTMODE=0 */
  setReg(TMR0_CTRL,0x20);              /* Stop all functions of the timer */
  /* TMR0_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                 Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
  setReg(TMR0_SCTRL,0x00);
  setReg(TMR0_LOAD,0x00);             /* Reset load register */
  setReg(TMR0_COMP1,46874);           /* Set up compare 1 register */
  setReg(TMR0_CMPLD1,46874);          /* Also set the compare preload register */
  /* TMR0_CSCTRL: DBG_EN=0,FAULT=0,ALT_LOAD=0,ROC=0,TCI=0,UP=0,OFLAG=0,TCF2EN=0,TCF1EN=1,
                  TCF2=0,TCF1=0,CL2=0,CL1=1 */
  setReg(TMR0_CSCTRL,0x41);           /* Enable compare 1 interrupt and */
                                       /*   compare 1 preload */
  setRegBitGroup(TMR0_CTRL,PCS,0xF); /* Primary Count Source to IP_bus_clk / 128 */
  setReg(TMR0_CNTR,0x00);             /* Reset counter register */
  setRegBitGroup(TMR0_CTRL,CM,0x01); /* Run counter */
}
```

54.4.5.3 Edge-Count Mode

If CTRL[CM] is set to '010', the counter will count both edges of the selected external clock source. This mode is useful for counting the changes in the external environment, such as a simple encoder wheel.

Example: 54.4.5.3.1 Count Both Edges of External Source Signal

```
//  This example uses TMR1 to count pulse (actually counts both edges of the pulse)
//    from an external source (QT3).
//
void Pulse_Init(void)
{
  /* TMR1_CTRL: CM=0,PCS=3,SCS=0,ONCE=0,LENGTH=0,DIR=0,COINIT=0,OUTMODE=0 */
  setReg(TMR1_CTRL,0x0600);           /* Set up mode */
  /* TMR1_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                 Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
  setReg(TMR1_SCTRL,0x00);
  setReg(TMR1_CNTR,0x00);             /* Reset counter register */
  setReg(TMR1_LOAD,0x00);             /* Reset load register */
  setRegBitGroup(TMR1_CTRL,CM,0x02);  /* Run counter */
}
```

54.4.5.4 Gated-Count Mode

If CTRL[CM] is set to '011', the counter will count while the selected secondary input signal is high. This mode is used to time the duration of external events. If the selected input is inverted by setting SCTRL[IPS] (input polarity select), then the counter will count while the selected secondary input is low.

Example: 54.4.5.4.1 Capture Duration of External Pulse

```
//   This example uses TMR1 to determine the duration of an external pulse.
//
//   The IP_bus clock is used as the primary counter. If the duration of the
//     external pulse is longer than 0.001 seconds one of the other IP_bus clock
//     dividers can be used. If the pulse duration is longer than 0.128 seconds
//     an external clock source will have to be used as the primary clock source.
//
void Pulse1_Init(void)
{
  /* TMR1_CTRL: CM=0,PCS=8,SCS=1,ONCE=0,LENGTH=0,DIR=0,COINIT=0,OUTMODE=0 */
  setReg(TMR1_CTRL,0x1080);             /* Set up mode */
  /* TMR1_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                 Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
  setReg(TMR1_SCTRL,0x00);
  setReg(TMR1_CNTR,0x00);              /* Reset counter register */
  setReg(TMR1_LOAD,0x00);              /* Reset load register */
  setRegBitGroup(TMR1_CTRL,CM,0x03);  /* Run counter */
}
```

54.4.5.5 Quadrature-Count Mode

If CTRL[CM] is set to '100', the counter will decode the primary and secondary external inputs as quadrature encoded signals. Quadrature signals are usually generated by rotary or linear sensors used to monitor movement of motor shafts or mechanical equipment. The quadrature signals are square waves that are 90 degrees out of phase. The decoding of quadrature signal provides both count and direction information.

This figure shows a timing diagram illustrating the basic operation of a quadrature incremental position encoder.

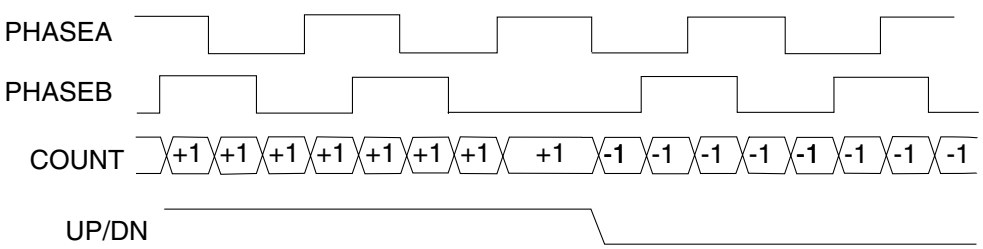


Figure 54-3. Quadrature Incremental Position Encoder

Example: 54.4.5.5.1 Quadrature Count Mode Example

```
//  This example uses TMR0 for counting states of a quadrature position encoder.
//
//  Timer input 0 is used as the primary count source (PHASEA).
//  Timer input 1 is used as the secondary count source (PHASEB).
//
void Pulse_Init(void)
{
  /* TMR0_CTRL: CM=0,PCS=0,SCS=1,ONCE=0,LENGTH=0,DIR=0,COINIT=0,OUTMODE=0 */
```

```
  setReg(TMRC0_CTRL,0x80);              /* Set up mode */
  /* TMR0_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                 Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=1 */
  setReg(TMR0_SCTRL,0x00);
  setReg(TMR0_CNTR,0x00);             /* Reset counter register */
  setReg(TMR0_LOAD,0x00);             /* Reset load register */
  setReg(TMR0_COMP1,0xFFFF);           /* Set up compare 1 register */
  setReg(TMR0_COMP2,0x00);             /* Set up compare 2 register */
  /* TMR0_CSCTRL: DBG_EN=0,FAULT=0,ALT_LOAD=0,ROC=0,TCI=0,UP=0,OFLAG=0,
                  TCF2EN=0,TCF1EN=0,TCF2=0,TCF1=0,CL2=0,CL1=0 */
  setReg(TMR0_CSCTRL,0x00);
  setRegBitGroup(TMR0_CTRL,CM,0x04);  /* Run counter */
}
```

54.4.5.6 Quadrature-Count Mode with Index Input

As an extension to the quadrature count mode discussed in the previous paragraph, some rotary shafts have a HOME or INDEX indicator. This would be a third input to the timer that is used to reset the timer's counter.

In this example, channel 0 is used to decode the quadrature inputs, but it doesn't actually count. Because its upper and lower limits are both set to 0, its output is cascaded count up and count down signals each time the quadrature inputs indicate a change in count. Channel 1 works in cascaded count mode receiving its counting instructions from channel 0. When an input capture event occurs, channel 1 is programmed to reset its counter value. The channel 1 counter contains the position value for the shaft.

Example: 54.4.5.6.1 Quadrature Count Mode with Index Input Example

```
//  This example uses TMR0and TMR1 for counting states of a quadrature position encoder.
//
// Timer input 0 is used as the primary count source (PHASEA).
// Timer input 1 is used as the secondary count source (PHASEB).
// Timer input 2 is used as the index input source (INDEX).
//
void Pulse_Init(void)
{
  /* TMR0_CTRL: CM=0,PCS=0,SCS=1,ONCE=0,LENGTH=1,DIR=0,COINIT=0,OUTMODE=0 */
  setReg(TMR0_CTRL,0xA0);              /* Set up mode */
  /* TMR0_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                 Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
  setReg(TMR0_SCTRL,0x00);
  setReg(TMR0_CNTR,0x00);             /* Reset counter register */
  setReg(TMR0_LOAD,0x00);             /* Reset load register */
  setReg(TMR0_COMP1,0x00);           /* Set up compare 1 register */
  setReg(TMR0_COMP2,0x00);             /* Set up compare 2 register */
  /* TMR0_CSCTRL: DBG_EN=0,FAULT=0,ALT_LOAD=0,ROC=0,TCI=0,UP=0,OFLAG=0,
                  TCF2EN=0,TCF1EN=0,TCF2=0,TCF1=0,CL2=0,CL1=0 */
  setReg(TMR0_CSCTRL,0x00);
/* TMR1_CTRL: CM=7,PCS=100,SCS=2,ONCE=0,LENGTH=0,DIR=0,COINIT=0,OUTMODE=0 */
  setReg(TMR1_CTRL,0xEA00);            /* Set up capture edge */
/* TMR1_SCTRL: Capture_Mode=10 */
  setReg(TMR1_SCTRL,0x0080);
/* TMR1_CSTRL: ROC=1 */
  setReg(TMR1_CTRL,0x0800);            /* Set up reload on capture */
setRegBitGroup(TMR0_CTRL,CM,0x04);  /* Run counter */
}
```

54.4.5.7 Signed-Count Mode

If CTRL[CM] is set to '101', the counter counts the primary clock source while the selected secondary source provides the selected count direction (up/down).

Example: 54.4.5.7.1 Signed Count Mode Example

```
//  This example uses TMR0 for signed mode counting.
//
//  Timer input 2 is used as the primary count source.
//  Timer input 1 is used to determine the count direction.
//
void Pulse_Init(void)
{
  /* TMR0_CTRL: CM=0,PCS=2,SCS=1,ONCE=0,LENGTH=0,DIR=0,COINIT=0,OUTMODE=0 */
  setReg(TMR0_CTRL,0x0480);              /* Set up mode */
  /* TMR0_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=1,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                 Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
  setReg(TMR0_SCTRL,0x1000);
  setReg(TMR0_CNTR,0x00);              /* Reset counter register */
  setReg(TMR0_LOAD,0x00);              /* Reset load register */
  setRegBitGroup(TMR0_CTRL,CM,0x05);  /* Run counter */
}
```

54.4.5.8 Triggered-Count Mode 1

If CSCTRL[TCI] is clear and CTRL[CM] is set to '110', the counter will begin counting the primary clock source after a positive transition (negative edge if SCTRL[IPS]=1) of the secondary input occurs. The counting will continue until a compare event occurs or another positive input transition is detected. If a second input transition occurs before a terminal count is reached, counting will stop and SCTRL[TCF] (timer compare flag) will be set. Subsequent secondary input transitions will continue to restart and stop the counting until a compare event occurs.

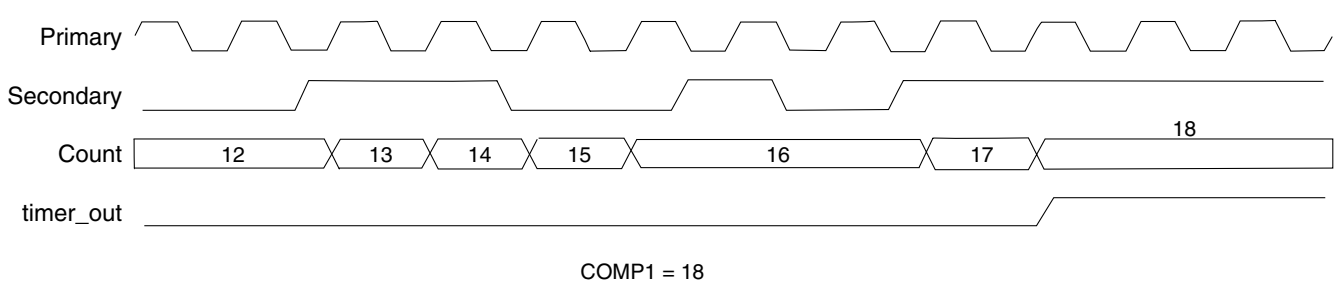


Figure 54-4. Triggered Count Mode 1 (CTRL[LENGTH]=0)

Example: 54.4.5.8.1 Triggered Count Mode 1 Example

```
//   This example uses TMR1 for triggered mode counting.
//
//   Timer input 3 is used as the primary count source.
//   Timer input 2 is used for the trigger input.
//
void Pulse_Init(void)
{
  /* TMR1_CTRL: CM=0,PCS=3,SCS=2,ONCE=0,LENGTH=0,DIR=0,COINIT=0,OUTMODE=0 */
  setReg(TMR1_CTRL,0x0700);             /* Set up mode */
  /* TMR1_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=1,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                 Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
  setReg(TMR1_SCTRL,0x1000);
  setReg(TMR1_CNTR,0x00);               /* Reset counter register */
  setReg(TMR1_LOAD,0x00);               /* Reset load register */
  setReg(TMR1_COMP1,0x0012);             /* Set up compare 1 register */
  /* TMR1_CSCTRL: DBG_EN=0,FAULT=0,ALT_LOAD=0,ROC=0,TCI=0,UP=0,OFLAG=0,
                  TCF2EN=0,TCF1EN=0,TCF2=0,TCF1=0,CL2=0,CL1=0 */
  setReg(TMR1_CSCTRL,0x00);
  setRegBitGroup(TMR1_CTRL,CM,0x06);    /* Run counter */
}
```

54.4.5.9 Triggered-Count Mode 2

If CSCTRL[TCI] is set and CTRL[CM] is set to '110', the counter will begin counting the primary clock source after a positive transition (negative edge if SCTRL[IPS]=1) of the secondary input occurs. The counting will continue until a compare event occurs or another positive input transition is detected. If a second input transition occurs before a terminal count was reached, the counter will reload and continue counting. When CSCTRL[TCI] is set, the OFLAG output mode, CTRL[OUTMODE], should probably be set to '101' (cleared on init, set on compare) to ensure the output will be in a known state after the second input transition and subsequent reload takes place.

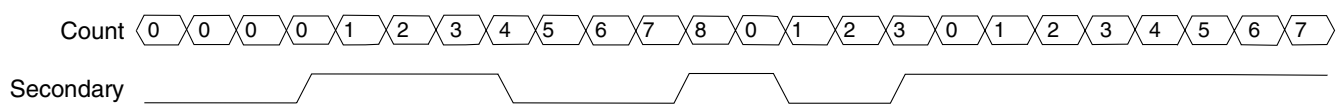


Figure 54-5. Triggered Count Mode 2 (CTRL[LENGTH]=0)

Example: 54.4.5.9.1 Triggered Count Mode 2Example

```
//   This example uses TMR1 for triggered mode counting.
//
//   Timer input 3 is used as the primary count source.
//   Timer input 2 is used for the trigger input.
//
void Pulse_Init(void)
{
  /* TMR1_CTRL: CM=0,PCS=3,SCS=2,ONCE=0,LENGTH=0,DIR=0,COINIT=0,OUTMODE=0 */
  setReg(TMR1_CTRL,0x0700);             /* Set up mode */
  /* TMR1_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=1,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                 Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
  setReg(TMR1_SCTRL,0x1000);

  setReg(TMR1_CNTR,0x00);               /* Reset counter register */
```

```
  setReg(TMR1_LOAD,0x00);             /* Reset load register */
  setReg(TMR1_COMP1,0x0012);           /* Set up compare 1 register */
  /* TMR1_CSCTRL: DBG_EN=0,FAULT=0,ALT_LOAD=0,ROC=0,TCI=0,UP=0,OFLAG=0,
                  TCF2EN=0,TCF1EN=0,TCF2=0,TCF1=0,CL2=0,CL1=0 */
  setReg(TMR1_CSCTRL,0x00);
  setRegBitGroup(TMR1_CTRL,CM,0x06); /* Run counter */
}
```

54.4.5.10 One-Shot Mode

If CTRL[CM] is set to '110', and the counter is set to reinitialize at a compare event (CTRL[LENGTH]=1),and CTRL[OUTMODE] is set to '101' (cleared on init, set on compare), the counter works in a one-shot mode. An external event causes the counter to count, and when the terminal count is reached, the output is asserted. This delayed output can be used to provide timing delays.

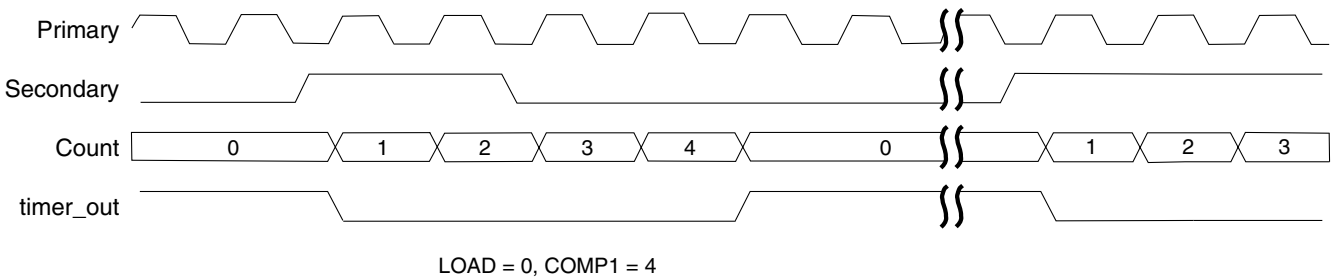


Figure 54-6. One-Shot Mode (CTRL[LENGTH]=1)

Example: 54.4.5.10.1 One-Shot Mode Example

```
//  This example uses TMR1 for one-shot mode counting.
//
//  Timer input 3 is used as the primary count source.
//  Timer input 2 is used for the trigger input.
//
void Pulse_Init(void)
{
  /* TMR1_CTRL: CM=0,PCS=3,SCS=2,ONCE=0,LENGTH=0,DIR=0,COINIT=0,OUTMODE=5 */
  setReg(TMR1_CTRL,0x0725);           /* Set up mode */
  /* TMR1_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=1,IEF=0,IEFIE=0,IPS=0,INPUT=0,
              Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
  setReg(TMR1_SCTRL,0x1000);
  setReg(TMR1_CNTR,0x00);             /* Reset counter register */
  setReg(TMR1_LOAD,0x00);             /* Reset load register */
  setReg(TMR1_COMP1,0x0004);           /* Set up compare 1 register */
  /* TMR1_CSCTRL: DBG_EN=0,FAULT=0,ALT_LOAD=0,ROC=0,TCI=0,UP=0,OFLAG=0,
                  TCF2EN=0,TCF1EN=0,TCF2=0,TCF1=0,CL2=0,CL1=0 */
  setReg(TMR1_CSCTRL,0x00);
  setRegBitGroup(TMR1_CTRL,CM,0x06);  /* Run counter */
}
```

54.4.5.11 Cascade-Count Mode

If CTRL[CM] is set to '111', the counter's input is connected to the output of another selected counter. The counter will count up and down as compare events occur in the selected source counter. This cascade or daisy-chained mode enables multiple counters to be cascaded to yield longer counter lengths. When operating in cascade mode, a special high-speed signal path is used between modules rather than the OFLAG output signal. If the selected source counter is counting up and it experiences a compare event, the counter will be incremented. If the selected source counter is counting down and it experiences a compare event, the counter will be decremented.

Up to four counters may be cascaded to create a 64-bit wide synchronous counter. Check the data sheet to see if there are any frequency limits for cascaded counting mode.

Whenever any counter is read within a counter module, all of the counters' values within the module are captured in their respective hold registers. This action supports the reading of a cascaded counter chain. First read any counter of a cascaded counter chain, then read the hold registers of the other counters in the chain. The cascaded counter mode is synchronous.

Note

It is possible to connect counters together by using the other (non-cascade) counter modes and selecting the outputs of other counters as a clock source. In this case, the counters are operating in a ripple mode, where higher order counters will transition a clock later than a purely synchronous design.

Example: 54.4.5.11.1 Generate Periodic Interrupt Cascading Two Counters

```
// This example generates an interrupt every 30 seconds,
//    assuming the chip is operating at 60 MHz.
//
// To do this, counter 2 is used to count 60,000 IP_bus clocks, which means it
//    will compare and reload every 0.001 seconds.
// Counter 3 is cascaded and used to count the 0.001 second ticks and
//    generate the desired interrupt interval.
//
void TimerInt_Init(void)
{
  // Set counter 2 to count IP_bus clocks
  /* TMR2_CTRL: CM=0,PCS=8,SCS=0,ONCE=0,LENGTH=1,DIR=0,COINIT=0,OUTMODE=0 */
  setReg(TMR2_CTRL,0x1020);           /* Stop all functions of the timer */
  // Set counter 3 as cascaded and to count counter 2 outputs
  /* TMR3_CTRL: CM=7,PCS=6,SCS=0,ONCE=0,LENGTH=1,DIR=0,COINIT=0,OUTMODE=0 */
  setReg(TMR3_CTRL,0xEC20);            /* Set up cascade counter mode */
  /* TMR3_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                 Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
  setReg(TMR3_SCTRL,0x00);
  /* TMR2_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                 Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
  setReg(TMR2_SCTRL,0x00);
```

```
  setReg(TMR3_CNTR,0x00);             /* Reset counter register */
  setReg(TMR2_CNTR,0x00);
  setReg(TMR3_LOAD,0x00);             /* Reset load register */
  setReg(TMR2_LOAD,0x00);
  setReg(TMR3_COMP1, 30000-1);        /* milliseconds in 30 seconds */
  setReg(TMR3_CMPLD1,30000-1);
  setReg(TMR2_COMP1, 60000-1);        /* Set to cycle every milisecond */
  setReg(TMR2_CMPLD1,60000-1);
  /* TMR3_CSCTRL: DBG_EN=0,FAULT=0,ALT_LOAD=0,ROC=0,TCI=0,UP=0,OFLAG=0,
                  TCF2EN=0,TCF1EN=1,TCF2=0,TCF1=0,CL2=0,CL1=1 */
  setReg(TMR3_CSCTRL,0x41);           /* Enable compare 1 interrupt and */
                                      /* compare 1 preload              */
  /* TMR2_CSCTRL: DBG_EN=0,FAULT=0,ALT_LOAD=0,ROC=0,TCI=0,UP=0,OFLAG=0,
                  TCF2EN=0,TCF1EN=0,TCF2=0,TCF1=0,CL2=0,CL1=1 */
  setReg(TMR2_CSCTRL,0x01);           /* Enable Compare 1 preload */
  setRegBitGroup(TMR2_CTRL,CM,0x01);  /* Run counter */
}
```

54.4.5.12 Pulse-Output Mode

If CTRL[CM]=001, and CTRL[OUTMODE] is set to 111' (gated clock output), and CTRL[ONCE] is set, then the counter will output a pulse stream of pulses that has the same frequency of the selected clock source, and the number of output pulses is equal to the compare value minus the init value. This mode is useful for driving step motor systems.

Note

This does not work if CTRL[PCS] is set to 1000 (IP_bus/1).

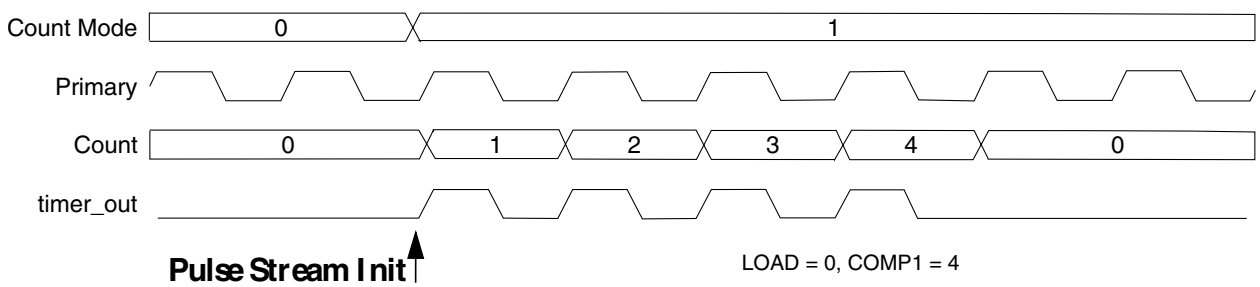


Figure 54-7. Pulse Output Mode

Example: 54.4.5.12.1 Pulse Outputs Using Two Counters

```
// This example generates six 10ms pulses, from QT1 output.
// Assuming the chip is operating at 60 MHz.
//
// To do this, timer 3 is used to generate a clock with a period of 10ms.
//
// Timer 1 is used to gate these clocks and count the number of pulses that have
//  been generated.
//
void PulseStream_Init(void)
{
// Select IP_bus_clk/16 as the clock source for Timer 3
```

```
  /* TMR3_CTRL: CM=0,PCS=0x0C,SCS=0,ONCE=0,LENGTH=1,DIR=0,COINIT=0,OUTMODE=3 */
   setReg(TMR3_CTRL,0x1823);             /* Set up mode */
  /* TMR3_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                 Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=0 */
   setReg(TMR3_SCTRL,0x00);
   setReg(TMR3_LOAD,0x00);              /* Reset load register */
   setReg(TMR3_COMP1,37500-1);          /* (16 * 37500 ) / 60e6 = 0.01 sec */
 /* TMR3_CSCTRL: DBG_EN=0,FAULT=0,ALT_LOAD=0,ROC=0,TCI=0,UP=0,OFLAG=0,
                   TCF2EN=0,TCF1EN=0,TCF2=0,TCF1=0,CL2=0,CL1=0 */
  setReg(TMR3_CSCTRL,0x00);             /* Set up comparator control register */



// Timer 3 output is the clock source for this timer.
  /* TMR1_CTRL: CM=0,PCS=7,SCS=0,ONCE=1,LENGTH=1,DIR=0,COINIT=0,OUTMODE=7 */
  setReg(TMR1_CTRL,0x0E67);            /* Set up mode */
  /* TMR1_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                 Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=0,OPS=0,OEN=1 */
   setReg(TMR1_SCTRL,0x01);
   setReg(TMR1_CNTR,0x00);             /* Reset counter register */
   setReg(TMR1_LOAD,0x00);             /* Reset load register */
   setReg(TMR1_COMP1,0x04);             /* Set up compare 1 register */

// set to interrupt after the last pulse
  /* TMR1_CSCTRL: DBG_EN=0,FAULT=0,ALT_LOAD=0,ROC=0,TCI=0,UP=0,OFLAG=0,
                  TCF2EN=0,TCF1EN=1,TCF2=0,TCF1=0,CL2=0,CL1=0 */
  setReg(TMR1_CSCTRL,0x40);             /* Set up comparator control register */
// Finally, start the counters running
  setReg(TMR3_CNTR,0);                  /* Reset counter */
  setRegBitGroup(TMR3_CTRL,CM,0x01);    /* Run source clock counter */
  setRegBitGroup(TMR1_CTRL,CM,0x01);    /* Run counter */
}
```

54.4.5.13 Fixed-Frequency PWM Mode

If CTRL[CM]=001, count through roll-over (CTRL[LENGTH]=0), continuous count (CTRL[ONCE]=0) and CTRL[OUTMODE] is '110' (set on compare, cleared on counter roll-over), then the counter output yields a pulse-width modulated (PWM) signal with a frequency equal to the count clock frequency divided by 65,536 and a pulse-width duty cycle equal to the compare value divided by 65,536. This mode of operation is often used to drive PWM amplifiers used to power motors and inverters.

Example: 54.4.5.13.1 Fixed-Frequency PWM Mode Example

```
// This example uses TMR0 for Fixed-Frequency PWM mode timing.
//
// The timer will count IP_bus clocks continuously until it rolls over.
// This results in a PWM period of 65536 / 60e6 = 1092.267 usec
//
// Initially, an output pulse width of 25 usec is generated ( 1500 / 60e6 )
//   giving a PWM ratio of 1500 / 65536 = 2.289%
// This pulse width can be changed by changing the COMP1 register value (using CMPLD1).
//
void PWM1_Init(void)
{
  setReg(TMR0_CNTR,0);           /* Reset counter */
  /* TMR0_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                 Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=1,OPS=0,OEN=1 */
```

```
  setReg(TMR0_SCTRL,0x05);       /* Enable output */
  setReg(TMR0_COMP1,1500-1);     /* Store initial value to the duty-compare register */
  /* TMR0_CTRL: CM=1,PCS=8,SCS=0,ONCE=0,LENGTH=0,DIR=0,COINIT=0,OUTMODE=6 */
  setReg(TMR0_CTRL,0x3006);      /* Run counter */
}
```

54.4.5.14 Variable-Frequency PWM Mode

If CTRL[CM]=001, count until compare (CTRL[LENGTH]=1), continuous count (CTRL[ONCE] = 0) and CTRL[OUTMODE] is '100' (toggle OFLAG and alternate compare registers), then the counter output yields a pulse-width modulated (PWM) signal whose frequency and pulse width is determined by the values programmed into the COMP1 and COMP2 registers, and the input clock frequency. This method of PWM generation has the advantage of allowing almost any desired PWM frequency and/or constant on or off periods. This mode of operation is often used to drive PWM amplifiers used to power motors and inverters. The CMPLD1 and CMPLD2 registers are especially useful for this mode, as they allow the programmer time to calculate values for the next PWM cycle while the PWM current cycle is underway.

To set up the TMR to run in variable frequency PWM mode with compare preload please use the following setup for the specific counter you would like to use. When performing the setup, update the TMR_CTRL register last because the counter will start counting if the count mode is changed to any value other than 000 (assuming the primary count source is already active).

Timer Control Register (CTRL)

- CM=001 (count rising edges of primary source)
- PCS=1000 (IP bus clock for best granularity for waveform timing)
- SCS=Any (ignored in this mode)
- ONCE=0 (want to count repeatedly)
- LENGTH=1 (want to count until compare value is reached and re-initialize counter register)
- DIR=Any (user's choice. The compare register values must be chosen carefully to account for things like roll-under, etc.)
- COINIT=0 (user can set this if they need this function)
- OUTMODE=100 (toggle OFLAG output using alternating compare registers)

Timer Status and Control Register (SCTRL)

- OEN = 1 (output enable to allow OFLAG output to be put on an external pin. Set this bit as needed.)
- OPS = Any (user's choice)
- Make sure the rest of the bits are cleared for this register. We will enable interrupts in the comparator status and control register instead of in this register.

Comparator Status and Control Register (CSCTRL)

- TCF2EN=1 (allow interrupt to be issued when CSCTRL[TCF2] is set)
- TCF1EN=0 (do not allow interrupt to be issued when CSCTRL[TCF1] is set)
- TCF1=0 (clear timer compare 1 interrupt source flag. This is set when counter register equals compare register 1 value and OFLAG is low)
- TCF2=0 (clear timer compare 2 interrupt source flag. This is set when counter register equals compare register 2 value and OFLAG is high)
- CL1=10 (load compare register when CSCTRL[TCF2] is asserted)
- CL2=01 (load compare register when CSCTRL[TCF1] is asserted)

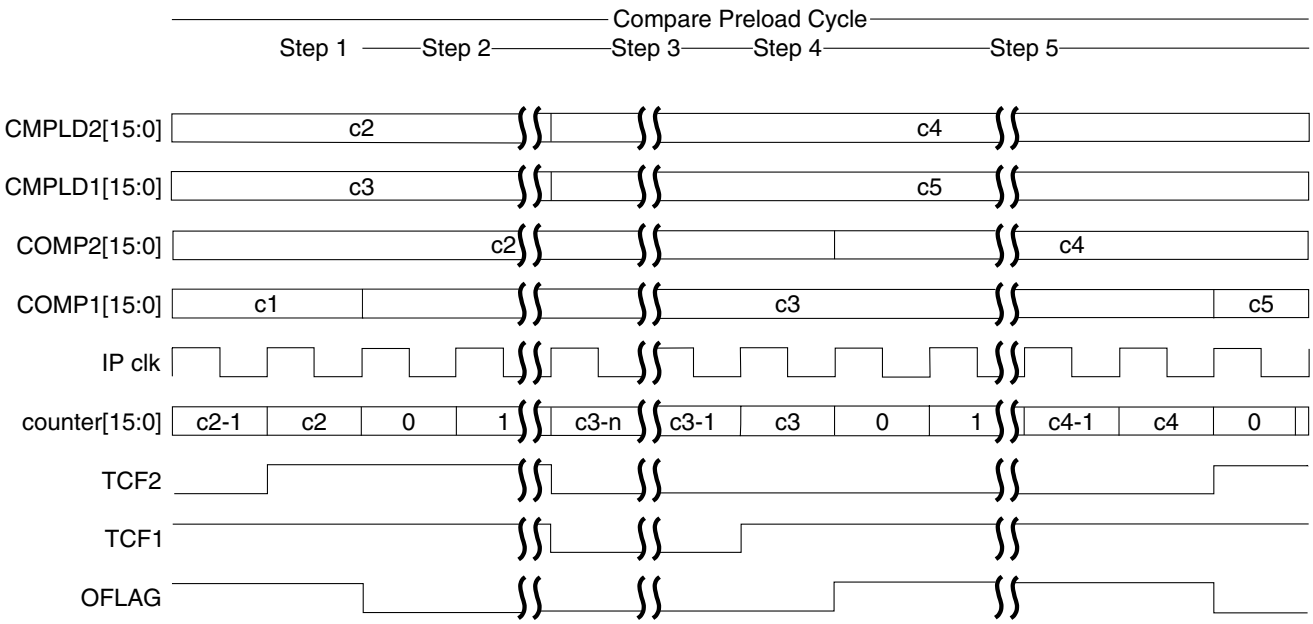
Interrupt Service Routines

To service the CSCTRL[TCF2] interrupts generated by the timer, the interrupt controller must be configured to enable the interrupts for the particular timer being used. Additionally the user will need to write an interrupt service routine to do at a minimum the following:

- Clear CSCTRL[TCF2] and CSCTRL[TCF1] flags.
- Calculate and write new values for both CMPLD1 and CMPLD2.

Timing

This figure contains the timing for using the compare preload feature. The compare preload cycle begins with a compare event on COMP2 causing CSCTRL[TCF2] to be asserted. COMP1 is loaded with the value in the CMPLD1 (c3) one IP bus clock later. In addition an interrupt is asserted by the timer and the interrupt service routine is executed during which both comparator load registers are updated with new values (c4 and c5). When CSCTRL[TCF1] is asserted, COMP2 is loaded with the value in CMPLD2 (c4). And on the subsequent CSCTRL[TCF2] event, COMP1 is loaded with the value in CMPLD1 (c5). The cycle starts over again as an interrupt is asserted and the interrupt service routine clears CSCTRL[TCF1] and CSCTRL[TCF2] and calculates new values for CMPLD1 and CMPLD2.



Step 1-- CNTR matches COMP2 value. CSCTRL[TCF2] is asserted and an interrupt request is generated.

Step 2-- One clock later, OFLAG toggles, CMPLD1 is copied to COMP1, LOAD is copied to CNTR, the counter starts counting.

Step 3-- The interrupt service routine clears CSCTRL[TCF1] and CSCTRL[TCF2] and the ISR loads CMPLD1 and CMPLD2 with the values for the next cycle. The counter continues counting until CNTR matches COMP1.

Step 4-- CSCTRL[TCF1] is asserted. One clock later, OFLAG toggles, CMPLD2 is copied to COMP2, LOAD is copied to CNTR and the counter starts counting.

Step 5--The counter continues counting until CNTR matches COMP2.

Figure 54-8. Compare Load Timing

Example: 54.4.5.14.1 Variable Frequency PWM Mode

```
// This example starts with an 11 msec with a 31 msec cycle.
// Assuming the chip is operating at 60 MHz, the timer use IP_bus_clk/32 as its
//   clock source.
//
// Initial pulse period: 60e6/32 clocks/sec * 31 ms = 58125 total clocks in period
// Initial pulse width: 60e6/32 clocks/sec * 11 ms = 20625 clocks in pulse
//
//
// Once the initial values of COMP1/CMPLD1 and COMP2/CMPLD2 are set the pulse width
//   can be varied by load new values of CMPLD1 and CMPLD2 on each compare interrupt.
//    (See Usage of Compare Load Registers.)
//
void PPG1_Init(void)
{
  setReg(TMR0_LOAD,0);                /* Clear load register */
  setReg(TMR0_CNTR,0);                /* Clear counter */
```

```
  /* TMR0_SCTRL: TCF=0,TCFIE=0,TOF=0,TOFIE=0,IEF=0,IEFIE=0,IPS=0,INPUT=0,
                 Capture_Mode=0,MSTR=0,EEOF=0,VAL=0,FORCE=1,OPS=0,OEN=1 */
  setReg(TMR0_SCTRL,5);                /* Set Status and Control Register */
  // Set compare preload operation and enable an interrupt on compare2 events.
  /* TMR0_CSCTRL: TCF2EN=1,TCF1EN=0,TCF2=0,TCF1=0,CL21=0,CL20=1,CL11=1,CL10=0 */
  setReg(TMR0_CSCTRL,0x86);            /* Set Comparator Status and Control Register */

  setReg(TMR0_COMP1, 20625-1);         /* Set the pulse width of the off time */
  setReg(TMR0_CMPLD1,20625-1);         /* Set the pulse width of the off time */
  setReg(TMR0_COMP2, 58125-20625-1);   /* Set the pulse width of the on time */
  setReg(TMR0_CMPLD2,58125-20625-1);   /* Set the pulse width of the on time */
  /* TMR0_CTRL: CM=1,PCS=0xD,SCS=0,ONCE=0,LENGTH=1,DIR=0,COINIT=0,OUTMODE=4 */
  setRegBits(TMR0_CTRL,0x3A24);        /* Set variable PWM mode and run counter */
}
```

54.5 Resets

54.5.1 General

The TMR module can be reset only by the RST_B signal. This forces all registers to their reset state and clears the OFLAG signal if it is asserted. The counter will be turned off until the settings in the control register are changed.

Table 54-2. Reset Summary

Reset	Priority	Source	Characteristics
RST_B	n/a	Hardware Reset	Full System Reset

54.6 Clocks

54.6.1 General

The timer only receives BUS_CLK_ROOT .

54.7 Interrupts

54.7.1 General

The TMR module can generate 20 interrupts, Five for each of the four counters/channels.

Table 54-3. Interrupt Summary

Core Interrupt	Interrupt	Description
TMR Channel 0	TMR0_COMP_IRQ_B	Compare Interrupt Request for Timer Channel 0
	TMR0_COMP1_IRQ_B	Compare 1 Interrupt Request for Timer Channel 0
	TMR0_COMP2_IRQ_B	Compare 2 Interrupt Request for Timer Channel 0
	TMR0_OVF_IRQ_B	Overflow Interrupt Request for Timer Channel 0
	TMR0_EDGE_IRQ_B	Input Edge Interrupt Request for Timer Channel 0
TMR Channel 1	TMR1_COMP_IRQ_B	Compare Interrupt Request for Timer Channel 1
	TMR1_COMP1_IRQ_B	Compare 1 Interrupt Request for Timer Channel 1
	TMR1_COMP2_IRQ_B	Compare 2 Interrupt Request for Timer Channel 1
	TMR1_OVF_IRQ_B	Overflow Interrupt Request for Timer Channel 1
	TMR1_EDGE_IRQ_B	Input Edge Interrupt Request for Timer Channel 1
TMR Channel 2	TMR2_COMP_IRQ_B	Compare Interrupt Request for Timer Channel 2
	TMR2_COMP1_IRQ_B	Compare 1 Interrupt Request for Timer Channel 2
	TMR2_COMP2_IRQ_B	Compare 2 Interrupt Request for Timer Channel 2
	TMR2_OVF_IRQ_B	Overflow Interrupt Request for Timer Channel 2
	TMR2_EDGE_IRQ_B	Input Edge Interrupt Request for Timer Channel 2
TMR Channel 3	TMR3_COMP_IRQ_B	Compare Interrupt Request for Timer Channel 3
	TMR3_COMP1_IRQ_B	Compare 1 Interrupt Request for Timer Channel 3
	TMR3_COMP2_IRQ_B	Compare 2 Interrupt Request for Timer Channel 3
	TMR3_OVF_IRQ_B	Overflow Interrupt Request for Timer Channel 3
	TMR3_EDGE_IRQ_B	Input Edge Interrupt Request for Timer Channel 3

Overflow interrupts do not work if counting upward - Errata ERR050194

54.7.2 Description of Interrupt Operation

54.7.2.1 Timer Compare Interrupts

These interrupts are generated when a successful compare occurs between a counter and it's compare registers while SCTRL[TCFIE] is set. These interrupts are cleared by writing a zero to the appropriate SCTRL[TCF].

When a timer compare interrupt is set in TMR_SCTRL and the Compare Load registers are available, one of the following two interrupts will also be asserted.

54.7.2.1.1 Timer Compare 1 Interrupts (Available with Compare Load Feature)

These interrupts are generated when a successful compare occurs between a counter and its COMP1 register while CSCTRL[TCF1EN] is set. These interrupts are cleared by writing a zero to the appropriate CSCTRL[TCF1].

54.7.2.1.2 Timer Compare 2 Interrupts (Available with Compare Load Feature)

These interrupts are generated when a successful compare occurs between a counter and its COMP2 register while CSCTRL[TCF2EN] is set. These interrupts are cleared by writing a zero to the appropriate CSCTRL[TCF2].

54.7.2.2 Timer Overflow Interrupts

These interrupts are generated when a counter rolls over its maximum value while SCTRL[TOFIE] is set. These interrupts are cleared by writing zero to the appropriate SCTRL[TOF].

54.7.2.3 Timer Input Edge Interrupts

These interrupts are generated by a transition of the input signal (either positive or negative depending on IPS setting) while SCTRL[IEFIE] is set. These interrupts are cleared by writing a zero to the appropriate SCTRL[IEF].

54.8 DMA

The TMR module can generate twelve DMA requests: three for each of the four counters/channels. Refer to the following table.

Table 54-4. DMA Summary

DMA Request	DMA Enable	Description
Channels 0-3	DMA[IEFDE]	CAPT contains a value
	DMA[CMPLD1DE]	CMPLD1 needs an update
	DMA[CMPLD2DE]	CMPLD2 needs an update

54.9 Memory map/register definition

54.9.1 TMR register descriptions

The address of a register is the sum of a base address and an address offset. The base address is defined at the chip level and the address offset is defined at the module level. Make certain to check which quad timer is available on the chip being used, and which timer channels have external I/O.

54.9.1.1 TMR memory map

TMR1 base address: 401D_C000h

TMR2 base address: 401E_0000h

TMR3 base address: 401E_4000h

TMR4 base address: 401E_8000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Timer Channel Compare Register 1 (COMP10)	16	RW	0000h
2h	Timer Channel Compare Register 2 (COMP20)	16	RW	0000h
4h	Timer Channel Capture Register (CAPT0)	16	RW	0000h
6h	Timer Channel Load Register (LOAD0)	16	RW	0000h
8h	Timer Channel Hold Register (HOLD0)	16	RW	0000h
Ah	Timer Channel Counter Register (CNTR0)	16	RW	0000h
Ch	Timer Channel Control Register (CTRL0)	16	RW	0000h
Eh	Timer Channel Status and Control Register (SCTRL0)	16	RW	0000h
10h	Timer Channel Comparator Load Register 1 (CMPLD10)	16	RW	0000h
12h	Timer Channel Comparator Load Register 2 (CMPLD20)	16	RW	0000h
14h	Timer Channel Comparator Status and Control Register (CSCTRL0)	16	RW	0000h
16h	Timer Channel Input Filter Register (FILT0)	16	RW	0000h
18h	Timer Channel DMA Enable Register (DMA0)	16	RW	0000h
1Eh	Timer Channel Enable Register (ENBL)	16	RW	000Fh
20h	Timer Channel Compare Register 1 (COMP11)	16	RW	0000h
22h	Timer Channel Compare Register 2 (COMP21)	16	RW	0000h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
24h	Timer Channel Capture Register (CAPT1)	16	RW	0000h
26h	Timer Channel Load Register (LOAD1)	16	RW	0000h
28h	Timer Channel Hold Register (HOLD1)	16	RW	0000h
2Ah	Timer Channel Counter Register (CNTR1)	16	RW	0000h
2Ch	Timer Channel Control Register (CTRL1)	16	RW	0000h
2Eh	Timer Channel Status and Control Register (SCTRL1)	16	RW	0000h
30h	Timer Channel Comparator Load Register 1 (CMPLD11)	16	RW	0000h
32h	Timer Channel Comparator Load Register 2 (CMPLD21)	16	RW	0000h
34h	Timer Channel Comparator Status and Control Register (CSCTRL1)	16	RW	0000h
36h	Timer Channel Input Filter Register (FILT1)	16	RW	0000h
38h	Timer Channel DMA Enable Register (DMA1)	16	RW	0000h
40h	Timer Channel Compare Register 1 (COMP12)	16	RW	0000h
42h	Timer Channel Compare Register 2 (COMP22)	16	RW	0000h
44h	Timer Channel Capture Register (CAPT2)	16	RW	0000h
46h	Timer Channel Load Register (LOAD2)	16	RW	0000h
48h	Timer Channel Hold Register (HOLD2)	16	RW	0000h
4Ah	Timer Channel Counter Register (CNTR2)	16	RW	0000h
4Ch	Timer Channel Control Register (CTRL2)	16	RW	0000h
4Eh	Timer Channel Status and Control Register (SCTRL2)	16	RW	0000h
50h	Timer Channel Comparator Load Register 1 (CMPLD12)	16	RW	0000h
52h	Timer Channel Comparator Load Register 2 (CMPLD22)	16	RW	0000h
54h	Timer Channel Comparator Status and Control Register (CSCTRL2)	16	RW	0000h
56h	Timer Channel Input Filter Register (FILT2)	16	RW	0000h
58h	Timer Channel DMA Enable Register (DMA2)	16	RW	0000h
60h	Timer Channel Compare Register 1 (COMP13)	16	RW	0000h
62h	Timer Channel Compare Register 2 (COMP23)	16	RW	0000h
64h	Timer Channel Capture Register (CAPT3)	16	RW	0000h
66h	Timer Channel Load Register (LOAD3)	16	RW	0000h
68h	Timer Channel Hold Register (HOLD3)	16	RW	0000h
6Ah	Timer Channel Counter Register (CNTR3)	16	RW	0000h
6Ch	Timer Channel Control Register (CTRL3)	16	RW	0000h
6Eh	Timer Channel Status and Control Register (SCTRL3)	16	RW	0000h
70h	Timer Channel Comparator Load Register 1 (CMPLD13)	16	RW	0000h
72h	Timer Channel Comparator Load Register 2 (CMPLD23)	16	RW	0000h
74h	Timer Channel Comparator Status and Control Register (CSCTRL3)	16	RW	0000h
76h	Timer Channel Input Filter Register (FILT3)	16	RW	0000h
78h	Timer Channel DMA Enable Register (DMA3)	16	RW	0000h

54.9.1.2 Timer Channel Compare Register 1 (COMP10 - COMP13)

54.9.1.2.1 Offset

Register	Offset
COMP10	0h
COMP11	20h
COMP12	40h
COMP13	60h

54.9.1.2.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	COMPARISON_1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

54.9.1.2.3 Fields

Field	Function
15-0 COMPARISON_1	Comparison Value 1 This read/write register stores the value used for comparison with the counter value in count up mode.

54.9.1.3 Timer Channel Compare Register 2 (COMP20 - COMP23)

54.9.1.3.1 Offset

Register	Offset
COMP20	2h
COMP21	22h
COMP22	42h
COMP23	62h

54.9.1.3.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	COMPARISON_2															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

54.9.1.3.3 Fields

Field	Function
15-0 COMPARISON_2	Comparison Value 2 This read/write register stores the value used for comparison with the counter value in count down mode or alternating compare mode.

54.9.1.4 Timer Channel Capture Register (CAPT0 - CAPT3)

54.9.1.4.1 Offset

Register	Offset
CAPT0	4h
CAPT1	24h
CAPT2	44h
CAPT3	64h

54.9.1.4.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	CAPTURE															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

54.9.1.4.3 Fields

Field	Function
15-0 CAPTURE	Capture Value This read/write register stores the value captured from the counter.

54.9.1.5 Timer Channel Load Register (LOAD0 - LOAD3)

54.9.1.5.1 Offset

Register	Offset
LOAD0	6h
LOAD1	26h
LOAD2	46h
LOAD3	66h

54.9.1.5.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	LOAD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

54.9.1.5.3 Fields

Field	Function
15-0 LOAD	Timer Load Register This read/write register stores the value used to initialize the counter after counter compare.

54.9.1.6 Timer Channel Hold Register (HOLD0 - HOLD3)

54.9.1.6.1 Offset

Register	Offset
HOLD0	8h
HOLD1	28h
HOLD2	48h
HOLD3	68h

54.9.1.6.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	HOLD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

54.9.1.6.3 Fields

Field	Function
15-0 HOLD	HOLD This read/write register stores the counter's values of specific channels whenever any of the four counters within a module is read.

54.9.1.7 Timer Channel Counter Register (CNTR0 - CNTR3)

54.9.1.7.1 Offset

Register	Offset
CNTR0	Ah
CNTR1	2Ah
CNTR2	4Ah
CNTR3	6Ah

54.9.1.7.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	COUNTER															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

54.9.1.7.3 Fields

Field	Function
15-0	COUNTER

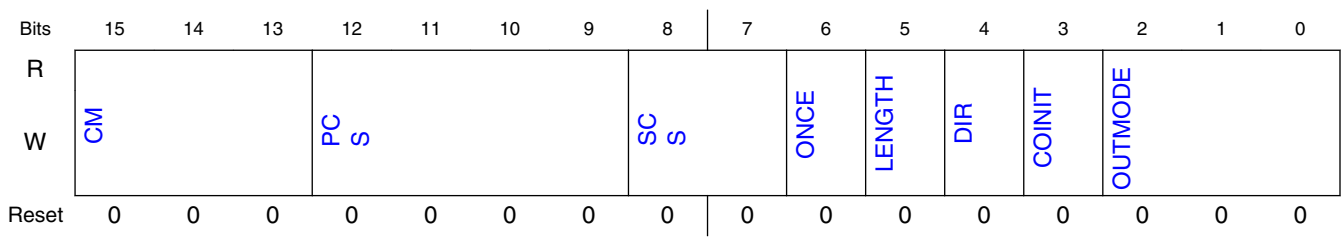
Field	Function
COUNTER	This read/write register is the counter for the corresponding channel in a timer module.

54.9.1.8 Timer Channel Control Register (CTRL0 - CTRL3)

54.9.1.8.1 Offset

Register	Offset
CTRL0	Ch
CTRL1	2Ch
CTRL2	4Ch
CTRL3	6Ch

54.9.1.8.2 Diagram



54.9.1.8.3 Fields

Field	Function
15-13 CM	Count Mode These bits control the basic counting and behavior of the counter. 000b - No operation 001b - Count rising edges of primary source[1] 010b - Count rising and falling edges of primary source[2] 011b - Count rising edges of primary source while secondary input high active 100b - Quadrature count mode, uses primary and secondary sources 101b - Count rising edges of primary source; secondary source specifies direction[3] 110b - Edge of secondary source triggers primary count until compare 111b - Cascaded counter mode (up/down)[4]
12-9 PCS	Primary Count Source These bits select the primary count source. **NOTE:** A timer selecting its own output for input is not a legal choice. The result is no counting. 0000b - Counter 0 input pin

Table continues on the next page...

Field	Function
	0001b - Counter 1 input pin 0010b - Counter 2 input pin 0011b - Counter 3 input pin 0100b - Counter 0 output 0101b - Counter 1 output 0110b - Counter 2 output 0111b - Counter 3 output 1000b - IP bus clock divide by 1 prescaler 1001b - IP bus clock divide by 2 prescaler 1010b - IP bus clock divide by 4 prescaler 1011b - IP bus clock divide by 8 prescaler 1100b - IP bus clock divide by 16 prescaler 1101b - IP bus clock divide by 32 prescaler 1110b - IP bus clock divide by 64 prescaler 1111b - IP bus clock divide by 128 prescaler
8-7 SCS	Secondary Count Source These bits identify the external input pin to be used as a count command or timer command. The selected input can trigger the timer to capture the current value of CNTR . The selected input can also be used to specify the count direction. The selected signal can also be used as a fault input when CSCTRL[FAULT] is set. The polarity of the signal can be inverted by SCTRL[IPS]. 00b - Counter 0 input pin 01b - Counter 1 input pin 10b - Counter 2 input pin 11b - Counter 3 input pin
6 ONCE	Count Once This bit selects continuous or one shot counting mode. 0b - Count repeatedly. 1b - Count until compare and then stop. If counting up, a successful compare occurs when the counter reaches a COMP1 value. If counting down, a successful compare occurs when the counter reaches a COMP2 value. When output mode $4 is used, the counter re-initializes after reaching the COMP1 value, continues to count to the COMP2 value, and then stops.
5 LENGTH	Count Length This bit determines whether the counter: • counts to the compare value and then re-initializes itself to the value specified in the LOAD (or CMPLD2) register, or • continues counting past the compare value to the binary roll over. 0b - Count until roll over at $FFFF and continue from $0000. 1b - Count until compare, then re-initialize. If counting up, a successful compare occurs when the counter reaches a COMP1 value. If counting down, a successful compare occurs when the counter reaches a COMP2 value. When output mode $4 is used, alternating values of COMP1 and COMP2 are used to generate successful comparisons. For example, the counter counts until a COMP1 value is reached, re-initializes, counts until COMP2 value is reached, re-initializes, counts until COMP1 value is reached, and so on.
4 DIR	Count Direction This bit selects either the normal count direction up, or the reverse direction, down. 0b - Count up. 1b - Count down.
3 COINIT	Co-Channel Initialization This bit enables another counter/timer within the module to force the re-initialization of this counter/timer when it has an active compare event.

Table continues on the next page...

Field	Function
	0b - Co-channel counter/timers cannot force a re-initialization of this counter/timer 1b - Co-channel counter/timers may force a re-initialization of this counter/timer
2-0 OUTMODE	Output Mode These bits determine the mode of operation for the OFLAG output signal. 000b - Asserted while counter is active 001b - Clear OFLAG output on successful compare 010b - Set OFLAG output on successful compare 011b - Toggle OFLAG output on successful compare 100b - Toggle OFLAG output using alternating compare registers 101b - Set on compare, cleared on secondary source input edge 110b - Set on compare, cleared on counter rollover 111b - Enable gated clock output while counter is active

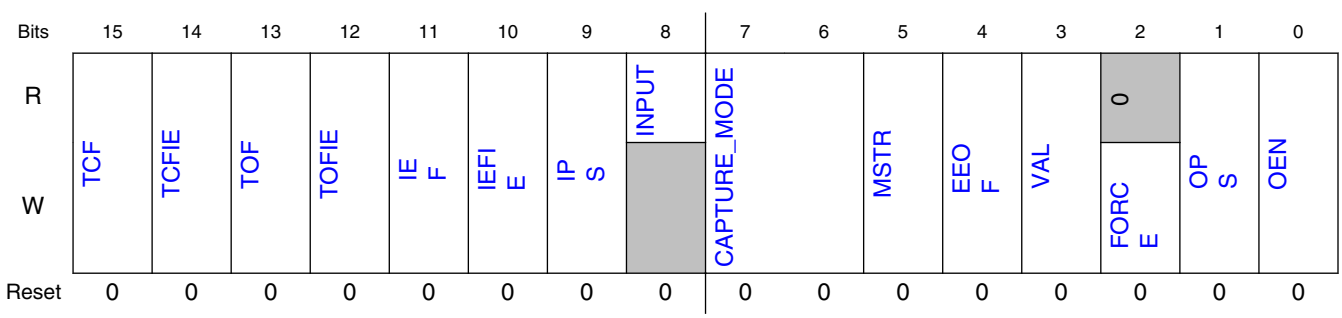
1. Rising edges are counted only when SCTRL[IPS] = 0. Falling edges are counted when SCTRL[IPS] = 1. If the primary count source is IP bus clock divide by 1, only rising edges are counted regardless of the value of SCTRL[IPS].
2. IP bus clock divide by 1 cannot be used as a primary count source in edge count mode.
3. Rising edges are counted only when SCTRL[IPS] = 0. Falling edges are counted when SCTRL[IPS] = 1.
4. The primary count source must be set to one of the counter outputs.

54.9.1.9 Timer Channel Status and Control Register (SCTRL0 - SCTRL3)

54.9.1.9.1 Offset

Register	Offset
SCTRL0	Eh
SCTRL1	2Eh
SCTRL2	4Eh
SCTRL3	6Eh

54.9.1.9.2 Diagram



54.9.1.9.3 Fields

Field	Function
15 TCF	Timer Compare Flag This bit is set when a successful compare occurs. This bit is cleared by writing a zero to this bit location.
14 TCFIE	Timer Compare Flag Interrupt Enable Can use compare with 0xFFFF as workaround for TOF This bit (when set) enables interrupts when TCF is set.
13 TOF	Timer Overflow Flag This bit is never set if counting upward - Errata ERR050144 This bit is set when the counter rolls over its maximum value $FFFF or $0000 (depending on count direction). This bit is cleared by writing a zero to this bit location.
12 TOFIE	Timer Overflow Flag Interrupt Enable This bit (when set) enables interrupts when TOF is set.
11 IEF	Input Edge Flag This bit is set when CAPTMODE is enabled and a proper input transition occurs (on an input selected as a secondary count source) while the count mode does not equal 000. This bit is cleared by writing a zero to this bit position. This bit can also be cleared automatically by a read of CAPT when DMA[IEFDE] is set. **NOTE:** Setting the input polarity select bit (IPS) changes the edge to be detected. Also, the control register's secondary count source determines which external input pin is monitored by the detection circuitry.
10 IEFIE	Input Edge Flag Interrupt Enable This bit (when set) enables interrupts when IEF is set. **Restriction:** Do not set both this bit and DMA[IEFDE].
9 IPS	Input Polarity Select This bit (when set) inverts the input signal polarity.
8 INPUT	External Input Signal This read-only bit reflects the current state of the external input pin selected via the secondary count source after application of IPS and filtering.
7-6 CAPTURE_MODE	Input Capture Mode These bits specify the operation of the capture register as well as the operation of the input edge flag. The input source is the secondary count source. 00b - Capture function is disabled 01b - Load capture register on rising edge (when IPS=0) or falling edge (when IPS=1) of input 10b - Load capture register on falling edge (when IPS=0) or rising edge (when IPS=1) of input 11b - Load capture register on both edges of input
5 MSTR	Master Mode This bit (when set) enables the compare function's output to be broadcasted to the other counters/timers in the module. This signal then can be used to re-initialize the other counters and/or force their OFLAG signal outputs.
4 EEOF	Enable External OFLAG Force This bit (when set) enables the compare from another counter/timer within the same module to force the state of this counter's OFLAG output signal.
3 VAL	Forced OFLAG Value This bit determines the value of the OFLAG output signal when software triggers a FORCE command.
2 FORCE	Force OFLAG Output

Table continues on the next page...

Field	Function
	This write only bit forces the current value of VAL to be written to the OFLAG output. This bit always reads as a zero. VAL and FORCE can be written simultaneously in a single write operation. Write to FORCE only if the counter is disabled. Setting this bit while the counter is enabled may yield unpredictable results.
1 OPS	Output Polarity Select This bit determines the polarity of the OFLAG output signal. 0b - True polarity. 1b - Inverted polarity.
0 OEN	Output Enable This bit determines the direction of the external pin. 0b - The external pin is configured as an input. 1b - The OFLAG output signal is driven on the external pin. Other timer groups using this external pin as their input see the driven value. The polarity of the signal is determined by OPS.

54.9.1.10 Timer Channel Comparator Load Register 1 (CMPLD10 - CMPLD13)

54.9.1.10.1 Offset

Register	Offset
CMPLD10	10h
CMPLD11	30h
CMPLD12	50h
CMPLD13	70h

54.9.1.10.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	COMPARATOR_LOAD_1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

54.9.1.10.3 Fields

Field	Function
15-0 COMPARATOR_LOAD_1	COMPARATOR_LOAD_1 This read/write register is the comparator 1 preload value for the COMP1 register for the corresponding channel in a timer module.

54.9.1.11 Timer Channel Comparator Load Register 2 (CMPLD20 - CMPLD23)

54.9.1.11.1 Offset

Register	Offset
CMPLD20	12h
CMPLD21	32h
CMPLD22	52h
CMPLD23	72h

54.9.1.11.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	COMPARATOR_LOAD_2															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

54.9.1.11.3 Fields

Field	Function
15-0 COMPARATOR_LOAD_2	COMPARATOR_LOAD_2 This read/write register is the comparator 2 preload value for the COMP2 register for the corresponding channel in a timer module.

54.9.1.12 Timer Channel Comparator Status and Control Register (CSCTRL0 - CSCTRL3)

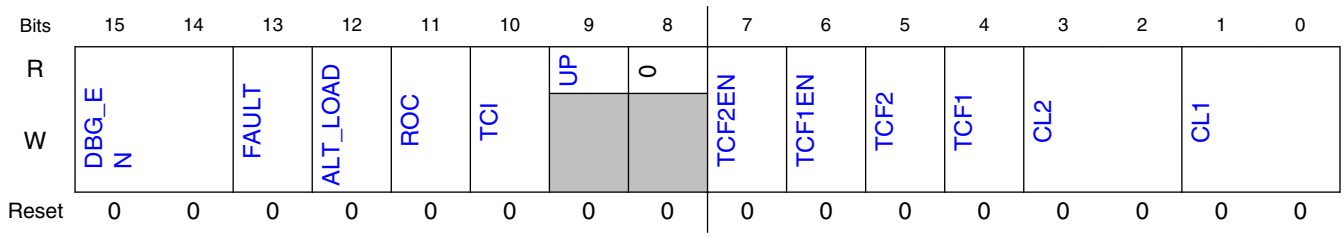
54.9.1.12.1 Offset

Register	Offset
CSCTRL0	14h
CSCTRL1	34h

Table continues on the next page...

Register	Offset
CSCTRL2	54h
CSCTRL3	74h

54.9.1.12.2 Diagram



54.9.1.12.3 Fields

Field	Function
15-14 DBG_EN	Debug Actions Enable These bits allow the TMR module to perform certain actions in response to the chip entering debug mode. 00b - Continue with normal operation during debug mode. (default) 01b - Halt TMR counter during debug mode. 10b - Force TMR output to logic 0 (prior to consideration of SCTRL[OPS]). 11b - Both halt counter and force output to 0 during debug mode.
13 FAULT	Fault Enable The selected secondary input acts as a fault signal so that the timer OFLAG is cleared when the secondary input is set. When the secondary input is used in this mode, there is no resynchronization of the input so that there is a combinational path to clear the OFLAG. Fault inputs less than two clock periods wide will not be latched. Latched faults will be cleared the next time that the counter logic sets the OFLAG. 0b - Fault function disabled. 1b - Fault function enabled.
12 ALT_LOAD	Alternative Load Enable This bit allows for an alternative method for loading the counter during modulo counting. Normally, the counter can be loaded only with the value from the LOAD register. When this bit is set, the counter is loaded from the LOAD register when counting up and a match with COMP1 occurs, or the counter is loaded from the CMPLD2 register when counting down and a match with COMP2 occurs. 0b - Counter can be re-initialized only with the LOAD register. 1b - Counter can be re-initialized with the LOAD or CMPLD2 registers depending on count direction.
11 ROC	Reload on Capture This bit enables the capture function to cause the counter to be reloaded from the LOAD register. 0b - Do not reload the counter on a capture event. 1b - Reload the counter on a capture event.
10	Triggered Count Initialization Control

Table continues on the next page...

Field	Function
TCI	This bit is used during triggered count mode, CTRL[CM] = 110, to enable the counter to be re-initialized when a second trigger occurs while the counter is still counting. Normally, the second trigger causes the counting to stop/pause until a third trigger occurs. With this bit set, a second trigger event causes the counter to re-initialize and continue counting. 0b - Stop counter upon receiving a second trigger event while still counting from the first trigger event. 1b - Reload the counter upon receiving a second trigger event while still counting from the first trigger event.
9 UP	Counting Direction Indicator This read-only bit is used during quadrature count mode, CTRL[CM] = 100, to read the direction of the last count. CTRL[DIR] reverses the sense of this bit. 0b - The last count was in the DOWN direction. 1b - The last count was in the UP direction.
8 —	RESERVED
7 TCF2EN	Timer Compare 2 Interrupt Enable An interrupt is issued when both this bit and TCF2 are set.
6 TCF1EN	Timer Compare 1 Interrupt Enable An interrupt is issued when both this bit and TCF1 are set.
5 TCF2	Timer Compare 2 Interrupt Flag When set, this bit indicates a successful comparison of the timer and the the COMP2 register has occurred. This bit is sticky, and will remain set until explicitly cleared by writing a zero to this bit location.
4 TCF1	Timer Compare 1 Interrupt Flag When set, this bit indicates a successful comparison of the timer and the the COMP1 register has occurred. This bit is sticky, and will remain set until explicitly cleared by writing a zero to this bit location.
3-2 CL2	Compare Load Control 2 These bits control when COMP2 is preloaded with the value from CMPLD2. 00b - Never preload 01b - Load upon successful compare with the value in COMP1 10b - Load upon successful compare with the value in COMP2 11b - Reserved
1-0 CL1	Compare Load Control 1 These bits control when COMP1 is preloaded with the value from CMPLD1. 00b - Never preload 01b - Load upon successful compare with the value in COMP1 10b - Load upon successful compare with the value in COMP2 11b - Reserved

54.9.1.13 Timer Channel Input Filter Register (FILT0 - FILT3)

54.9.1.13.1 Offset

Register	Offset
FILT0	16h
FILT1	36h
FILT2	56h
FILT3	76h

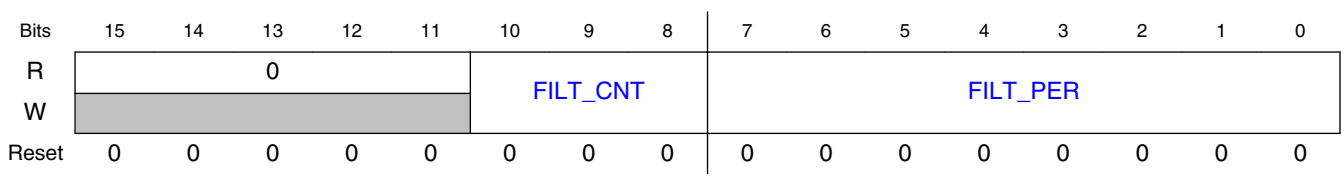
54.9.1.13.2 Function

The FILT register programs the values for the filtering of the corresponding input without regard for the fact that any timer channel can use the input as a count source.

Input filter considerations:

- Set the FILT_PER value such that the sampling period is larger than the period of the expected noise. In this way, a noise spike will corrupt only one sample. Choose the FILT_CNT value to reduce the probability that noisy samples cause an incorrect transition to be recognized. The probability of an incorrect transistion is defined as the probability of an incorrect sample raised to the power of (FILT_CNT + 3).
- The values of FILT_PER and FILT_CNT must also be balanced against the desire for minimal latency in recognizing input transitions. Turning on the input filter (setting FILT_PER to a non-zero value) introduces a latency of (((FILT_CNT + 3) x FILT_PER) + 2) IP bus clock periods.

54.9.1.13.3 Diagram



54.9.1.13.4 Fields

Field	Function
15-11 —	RESERVED
10-8 FILT_CNT	Input Filter Sample Count These bits represent the number of consecutive samples that must agree prior to the input filter accepting an input transition. A value of 0x0 represents 3 samples. A value of 0x7 represents 10 samples. The value of FILT_CNT affects the input latency.

Table continues on the next page...

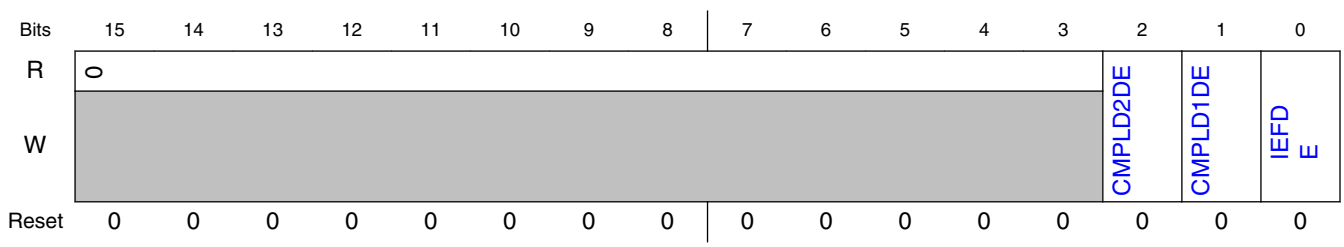
Field	Function
7-0 FILT_PER	Input Filter Sample Period These bits represent the sampling period (in IP bus clock cycles) of the TMR input signals. Each input is sampled multiple times at the rate specified by this field. If FILT_PER is 0x00 (default), then the input filter is bypassed. The value of FILT_PER affects the input latency. When changing values for FILT_PER from one non-zero value to another non-zero value, write a value of zero first to clear the filter.

54.9.1.14 Timer Channel DMA Enable Register (DMA0 - DMA3)

54.9.1.14.1 Offset

Register	Offset
DMA0	18h
DMA1	38h
DMA2	58h
DMA3	78h

54.9.1.14.2 Diagram



54.9.1.14.3 Fields

Field	Function
15-3 —	RESERVED
2 CMPLD2DE	Comparator Preload Register 2 DMA Enable Setting this bit enables DMA write requests for CMPLD2 whenever data is transferred out of the CMPLD2 register into the CNTR or COMP2 registers.
1 CMPLD1DE	Comparator Preload Register 1 DMA Enable

Table continues on the next page...

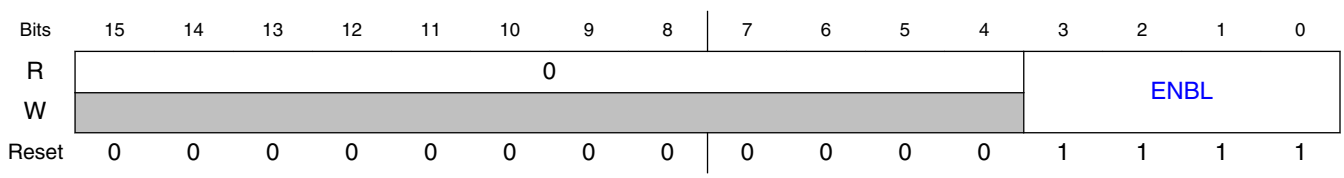
Field	Function
	Setting this bit enables DMA write requests for CMPLD1 whenever data is transferred out of the CMPLD1 register into the COMP1 register.
0 IEFDE	Input Edge Flag DMA Enable Setting this bit enables DMA read requests for CAPT when SCTRL[IEF] is set. **Restriction:** Do not set both this bit and SCTRL[IEFIE].

54.9.1.15 Timer Channel Enable Register (ENBL)

54.9.1.15.1 Offset

Register	Offset
ENBL	1Eh

54.9.1.15.2 Diagram



54.9.1.15.3 Fields

Field	Function
15-4 —	RESERVED
3-0 ENBL	Timer Channel Enable These bits enable the prescaler (if it is being used) and counter in each channel. Multiple ENBL bits can be set at the same time to synchronize the start of separate counters. If an ENBL bit is set, then the corresponding channel starts its counter as soon as the CTRL[CM] field has a value other than 0. When an ENBL bit is clear, the corresponding counter maintains its current value. 0000b - Timer channel is disabled. 0001b - Timer channel is enabled. (default)

Chapter 55
Enhanced Flex Pulse Width Modulator (eFlexPWM)

55.1 Chip-specific FlexPWM information

Table 55-1. Reference links to related information

Topic	Related module(s)	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
Interrupts, DMA Events and XBAR Assignments	-	Interrupts, DMA Events and XBAR Assignments

NOTE

In this device, PWM*x*_EXTB is not applicable. It also does not support external ADC input. Also there is **no** NanoEdge placement block in this device.

NOTE

PWM_X only applicable to FlexPWM1, while other FlexPWM instances do not have the PWMX outputs or inputs.

55.2 Overview

The pulse width modulator (PWM) module contains PWM submodules, each of which is set up to control a single half-bridge power stage. Fault channel support is provided.

eFlexPwm Library:
https://epsilonrt.github.io/eFlexPwm/
https://forum.pjrc.com/index.php?threads/72458

This PWM module can generate various switching patterns, including highly sophisticated waveforms. It is ideal for controlling different Switched Mode Power Supplies (SMPS) topologies.

55.2.1 Block Diagram

The following figure is a block diagram of the PWM.

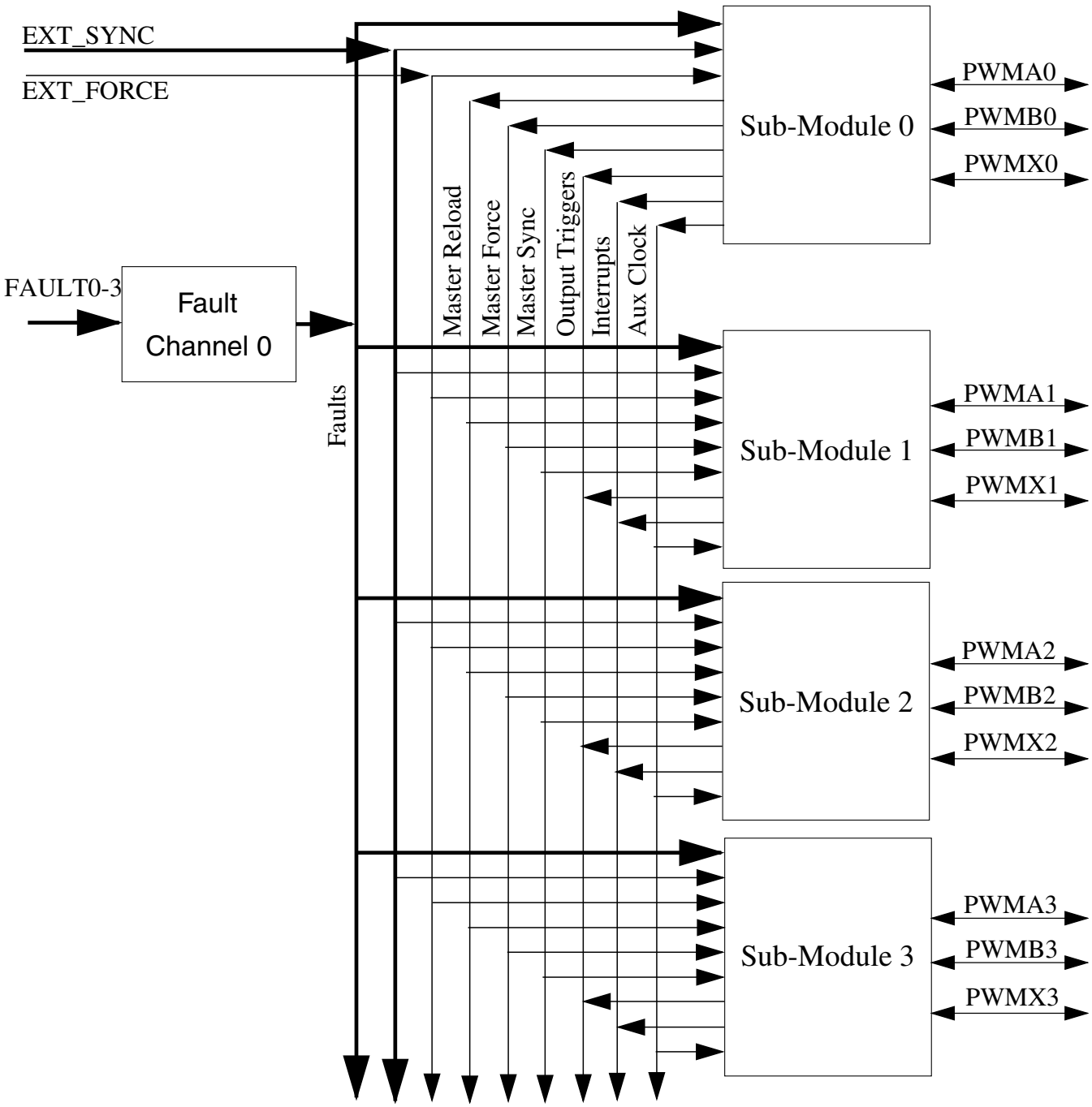


Figure 55-1. PWM Block Diagram

55.2.1.1 PWM Submodule

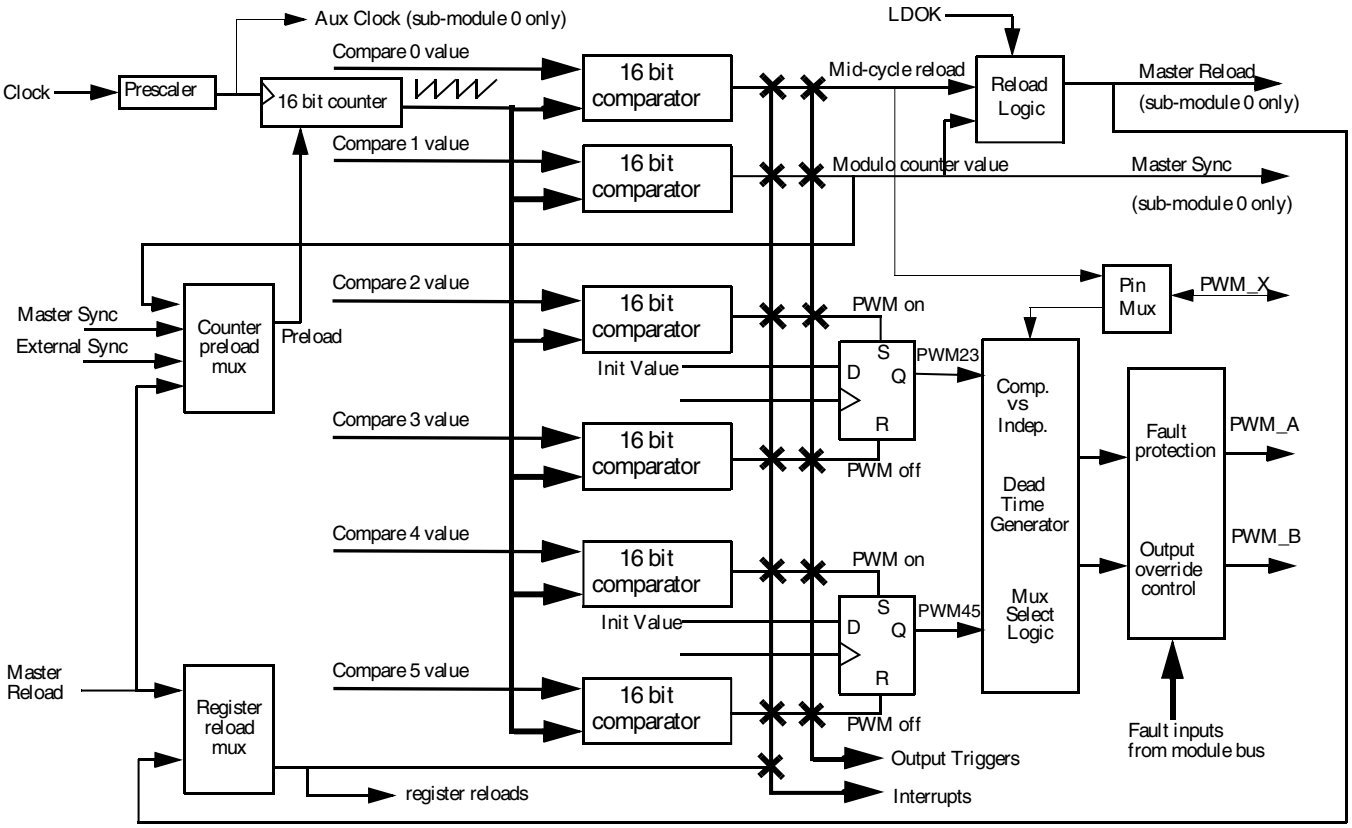


Figure 55-2. PWM Submodule Block Diagram

55.2.2 Features

- 16 bits of resolution for center, edge-aligned, and asymmetrical PWMs
- Fractional PWM clock generation for enhanced resolution of the PWM period and duty cycle
- PWM outputs that can operate as complementary pairs or independent channels
- Ability to accept signed numbers for PWM generation
- Independent control of both edges of each PWM output
- Support for synchronization to external hardware or other PWM
- Double buffered PWM registers
  - Integral reload rates from 1 to 16
  - Half cycle reload capability
- Multiple output trigger events can be generated per PWM cycle via hardware
- Support for double switching PWM outputs
- Fault inputs can be assigned to control multiple PWM outputs

- Programmable filters for fault inputs
- Independently programmable PWM output polarity
- Independent top and bottom deadtime insertion
- Each complementary pair can operate with its own PWM frequency and deadtime values
- Individual software control for each PWM output
- All outputs can be programmed to change simultaneously via a FORCE_OUT event
- PWM_X pin can optionally output a third PWM signal from each submodule
- Channels not used for PWM generation can be used for buffered output compare functions
- Channels not used for PWM generation can be used for input capture functions
- Enhanced dual edge capture functionality

55.2.3 Modes of operation

Be careful when using this module in Stop, Wait and Debug operating modes.

CAUTION

Some applications require regular software updates for proper operation. Failure to provide regular software updates could result in destroying the hardware setup.

To accommodate this situation, PWM outputs are placed in their inactive states in Stop mode, and they can optionally be placed in inactive states in Wait and Debug modes. PWM outputs are reactivated (assuming they were active beforehand) when these modes are exited.

Table 55-2. Modes when PWM Operation is Restricted

Mode	Description
Stop	PWM outputs are inactive.
Wait	PWM outputs are driven or inactive as a function of CTRL2[WAITEN].
Debug	CPU and peripheral clocks continue to run, but the CPU may stall for periods of time. PWM outputs are driven or inactive as a function of CTRL2[DBGEN].

55.3 Functional Description

55.3.1 PWM Capabilities

This section describes some capabilities of the PWM module.

55.3.1.1 Center Aligned PWMs

Each submodule has its own timer that is capable of generating PWM signals on two output pins. The edges of each of these signals are controlled independently as shown in Figure 55-3.

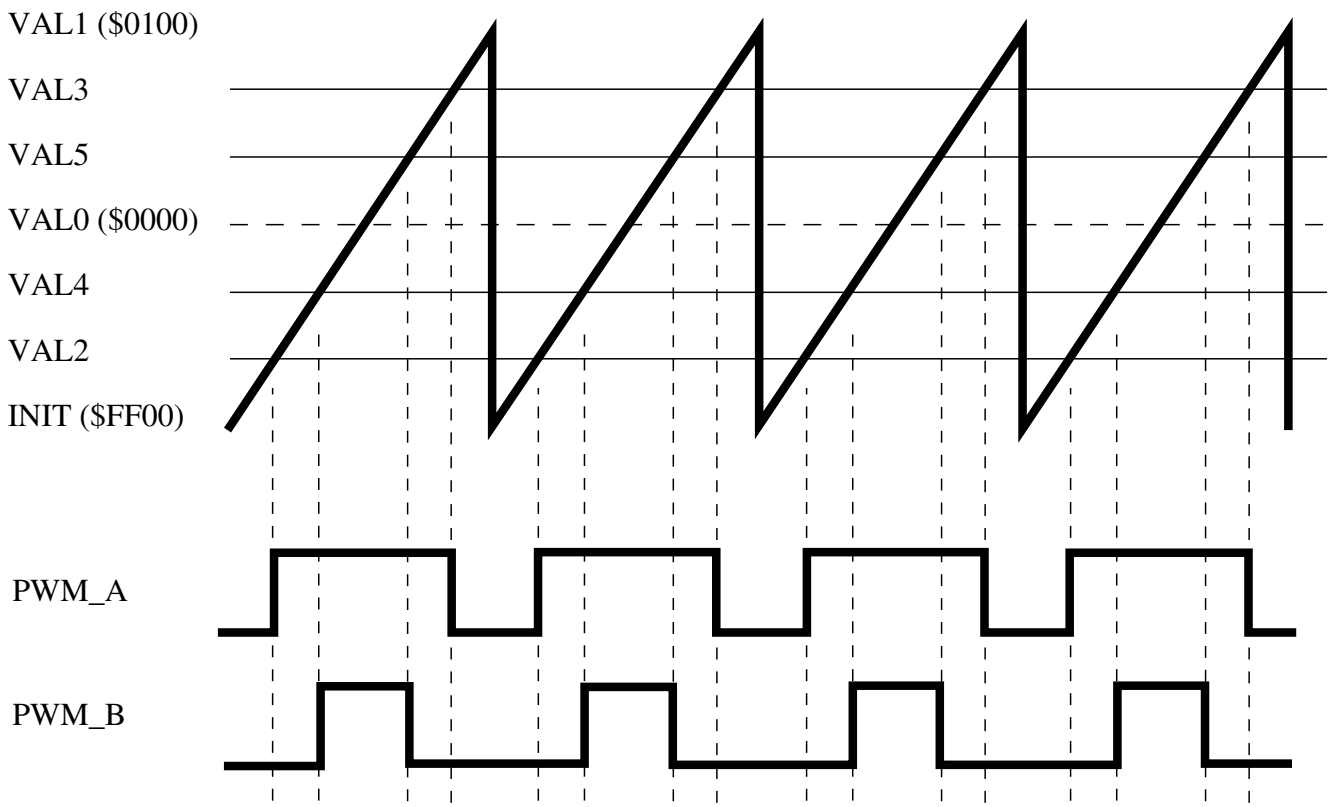


Figure 55-3. Center Aligned Example

The submodule timers only count in the up direction and then reset to the INIT value. Instead of having a single value that determines pulse width, there are two values that must be specified: the turn on edge and the turn off edge. This double action edge generation not only gives the user control over the pulse width, but over the relative alignment of the signal as well. As a result, there is no need to support separate PWM alignment modes since the PWM alignment mode is inherently a function of the turn on and turn off edge values.

Figure 55-3 also illustrates an additional enhancement to the PWM generation process. When the counter resets, it is reloaded with a user specified value, which may or may not be zero. If the value chosen happens to be the 2's complement of the modulus value, then

the PWM generator operates in "signed" mode. This means that if each PWM's turn on and turn off edge values are also the same number but only different in their sign, the "on" portion of the output signal will be centered around a count value of zero. Therefore, only one PWM value needs to be calculated in software and then this value and its negative are provided to the submodule as the turn off and turn on edges respectively. This technique will result in a pulse width that always consists of an odd number of timer counts. If all PWM signal edge calculations follow this same convention, then the signals will be center aligned with respect to each other, which is the goal. Of course, center alignment between the signals is not restricted to symmetry around the zero count value, as any other number would also work. However, centering on zero provides the greatest range in signed mode and also simplifies the calculations.

55.3.1.2 Edge Aligned PWMs

When the turn on edge for each pulse is specified to be the INIT value, then edge aligned operation results, as the following figure shows. Therefore, only the turn off edge value needs to be periodically updated to change the pulse width.

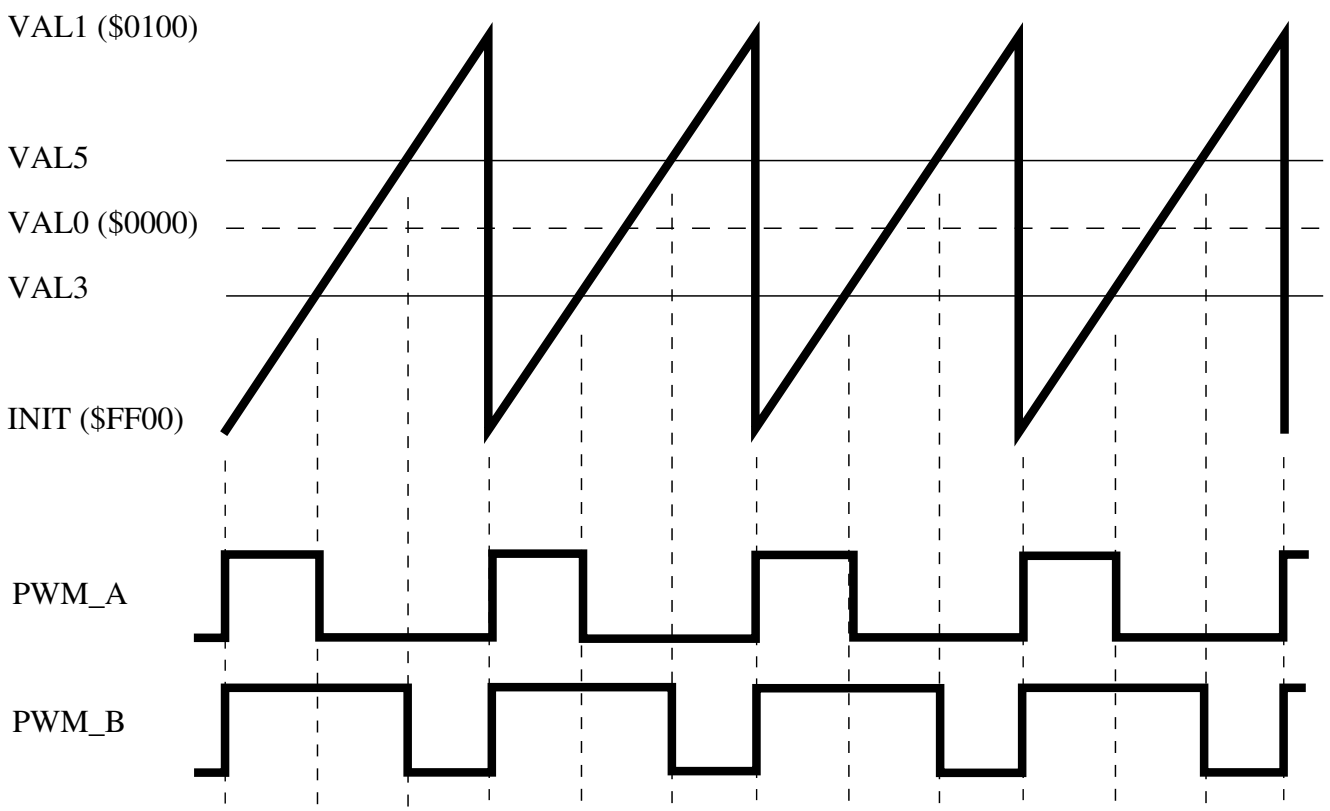


Figure 55-4. Edge Aligned Example (INIT=VAL2=VAL4)

With edge aligned PWMs, another example of the benefits of signed mode can be seen. A common way to drive an H-bridge is to use a technique called "bipolar" PWMs where a 50% duty cycle results in zero volts on the load. Duty cycles less than 50% result in negative load voltages and duty cycles greater than 50% generate positive load voltages. If the module is set to signed mode operation (the INIT and VAL1 values are the same number with opposite signs), then there is a direct proportionality between the PWM turn off edge value and the motor inverter voltage, INCLUDING the sign. So once again, signed mode of operation simplifies the software interface to the PWM module since no offset calculations are required to translate the output variable control algorithm to the voltage on an H-Bridge load.

55.3.1.3 Phase Shifted PWMs

In the previous sections, the benefits of signed mode of operation were discussed in the context of simplifying the required software calculations by eliminating the requirement to bias up signed variables before applying them to the module. However, if numerical biases are applied to the turn on and turn off edges of different PWM signal, the signals will be phase shifted with respect to each other, as the following figure shows. This results in certain advantages when applied to a power stage. For example, when operating a multi-phase inverter at a low modulation index, all of the PWM switching edges from the different phases occur at nearly the same time. This can be troublesome from a noise standpoint, especially if ADC readings of the inverter must be scheduled near those times. Phase shifting the PWM signals can open up timing windows between the switching edges to allow a signal to be sampled by the ADC. However, phase shifting does NOT affect the duty cycle so average load voltage is not affected.

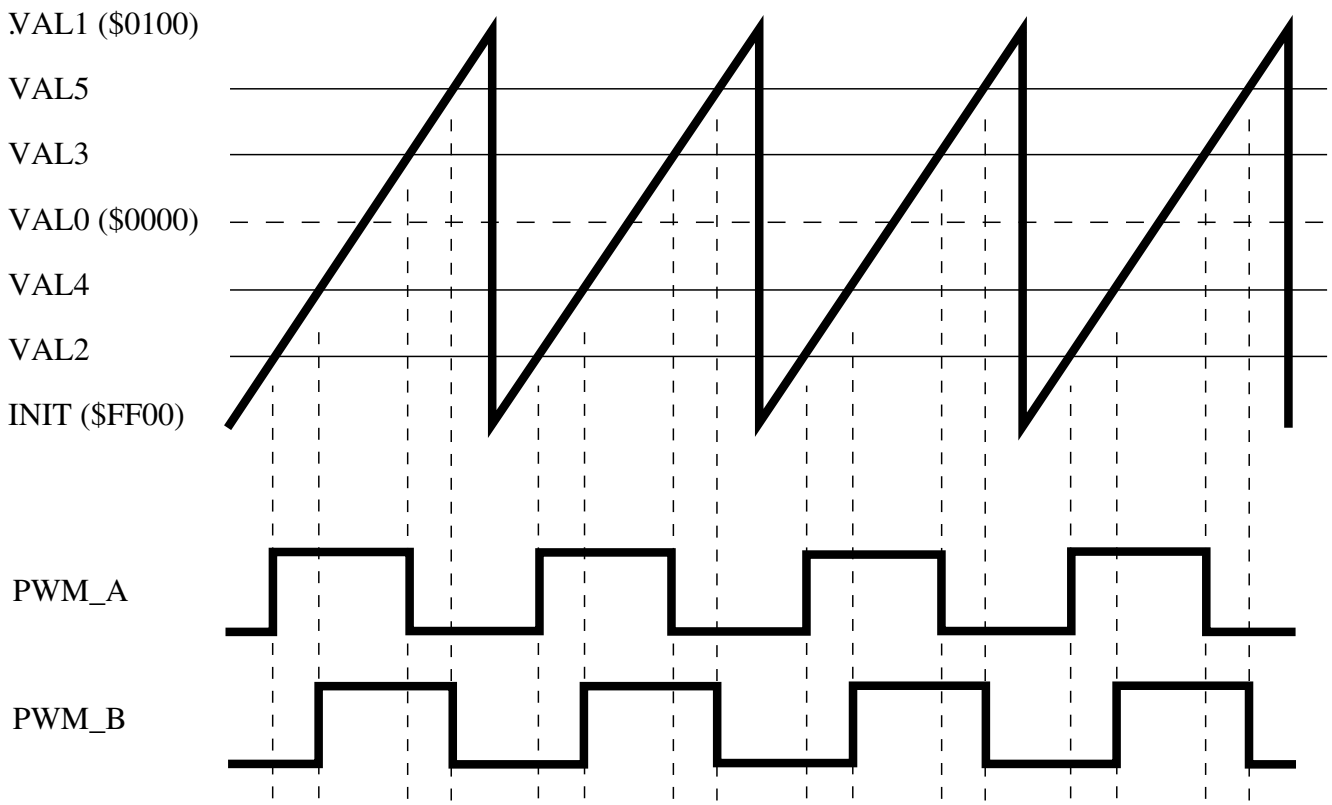


Figure 55-5. Phase Shifted Outputs Example

An additional benefit of phase shifted PWMs appears in Figure 55-6. In this case, an H-Bridge circuit is driven by 4 PWM signals to control the voltage waveform on the primary of a transformer. Both left and right side PWMs are configured to always generate a square wave with 50% duty cycle. This works out nicely for the H-Bridge since no narrow pulse widths are generated reducing the high frequency switching requirements of the transistors. Notice that the square wave on the right side of the H-Bridge is phase shifted compared to the left side of the H-Bridge. As a result, the transformer primary sees the bottom waveform across its terminals. The RMS value of this waveform is directly controlled by the amount of phase shift of the square waves. Regardless of the phase shift, no DC component appears in the load voltage as long as the duty cycle of each square wave remains at 50% making this technique ideally suited for transformer loads. As a result, this topology is frequently used in industrial welders to adjust the amount of energy delivered to the weld arc.

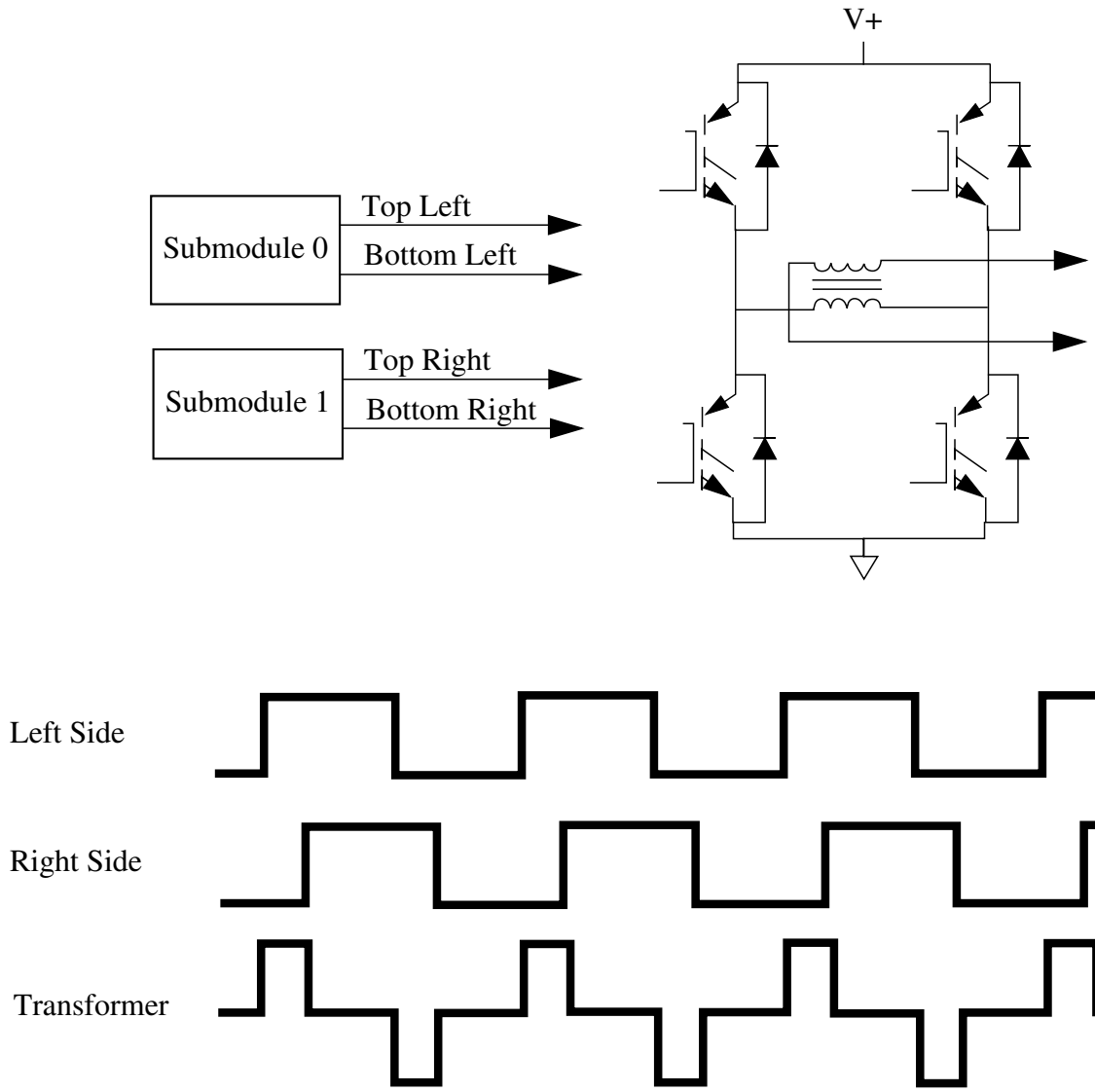


Figure 55-6. Phase Shifted PWMs Applied to a Transformer Primary

55.3.1.4 Double Switching PWMs

Double switching PWM output is supported to aid in single shunt current measurement and three phase reconstruction. This method support two independent rising edges and two independent falling edges per PWM cycle. The VAL2 and VAL3 registers are used to generate the even channel (labelled as PWM_A in the figure) while VAL4 and VAL5 are used to generate the odd channel. The two channels are combined using XOR logic (force out logic) as the following figure shows. The DBLPWM signal can be run through the deadtime insertion logic.

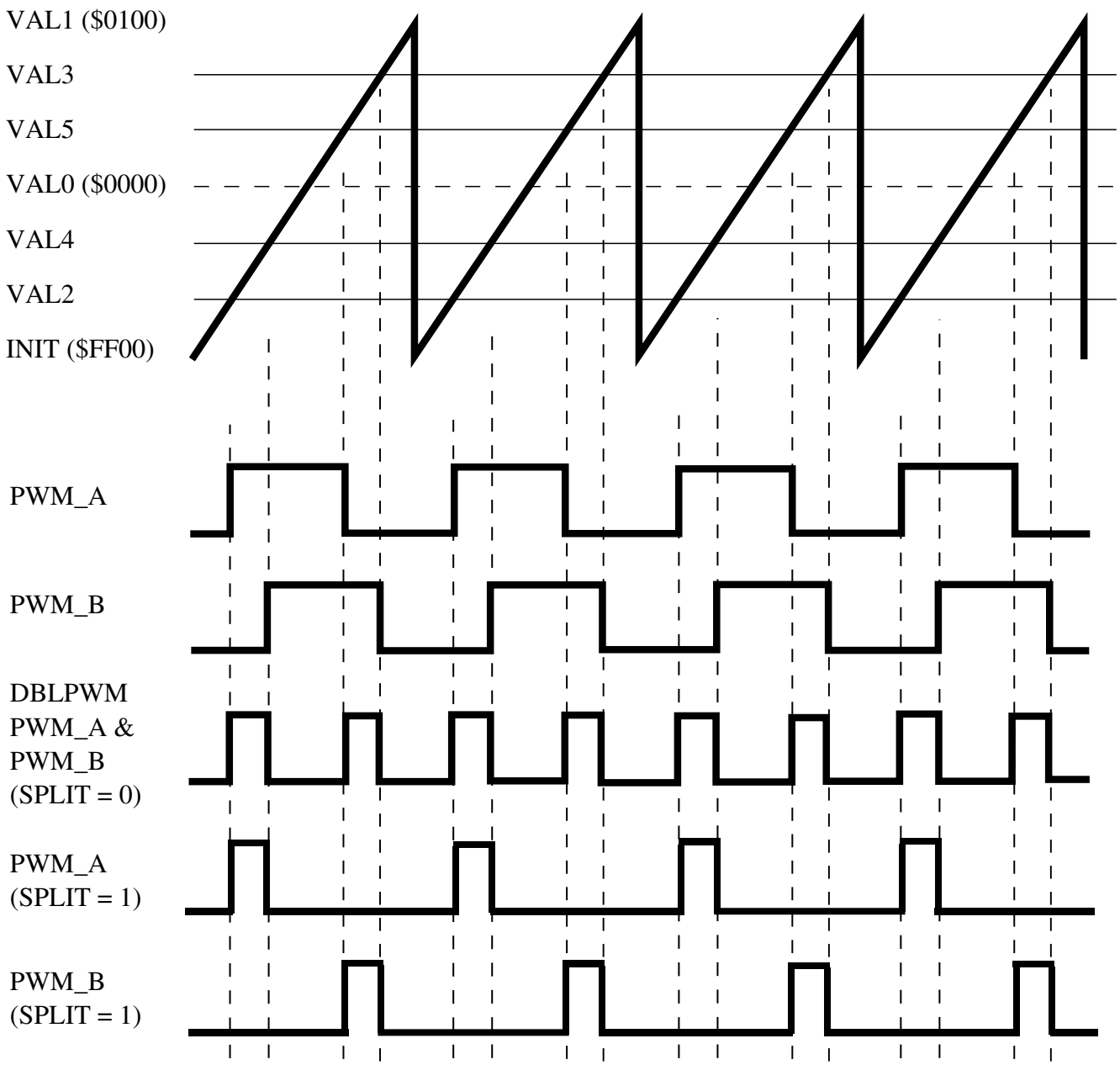


Figure 55-7. Double Switching Output Example

55.3.1.5 ADC Triggering

In cases where the timing of the ADC triggering is critical, it must be scheduled as a hardware event instead of software activated. With this PWM module, multiple ADC triggers can be generated in hardware per PWM cycle without the requirement of another timer module. Figure 55-8 shows how this is accomplished. When specifying complementary mode of operation, only two edge comparators are required to generate the output PWM signals for a given submodule. This means that the other comparators

are free to perform other functions. In this example, the software does not need to quickly respond after the first conversion to set up other conversions that must occur in the same PWM cycle.

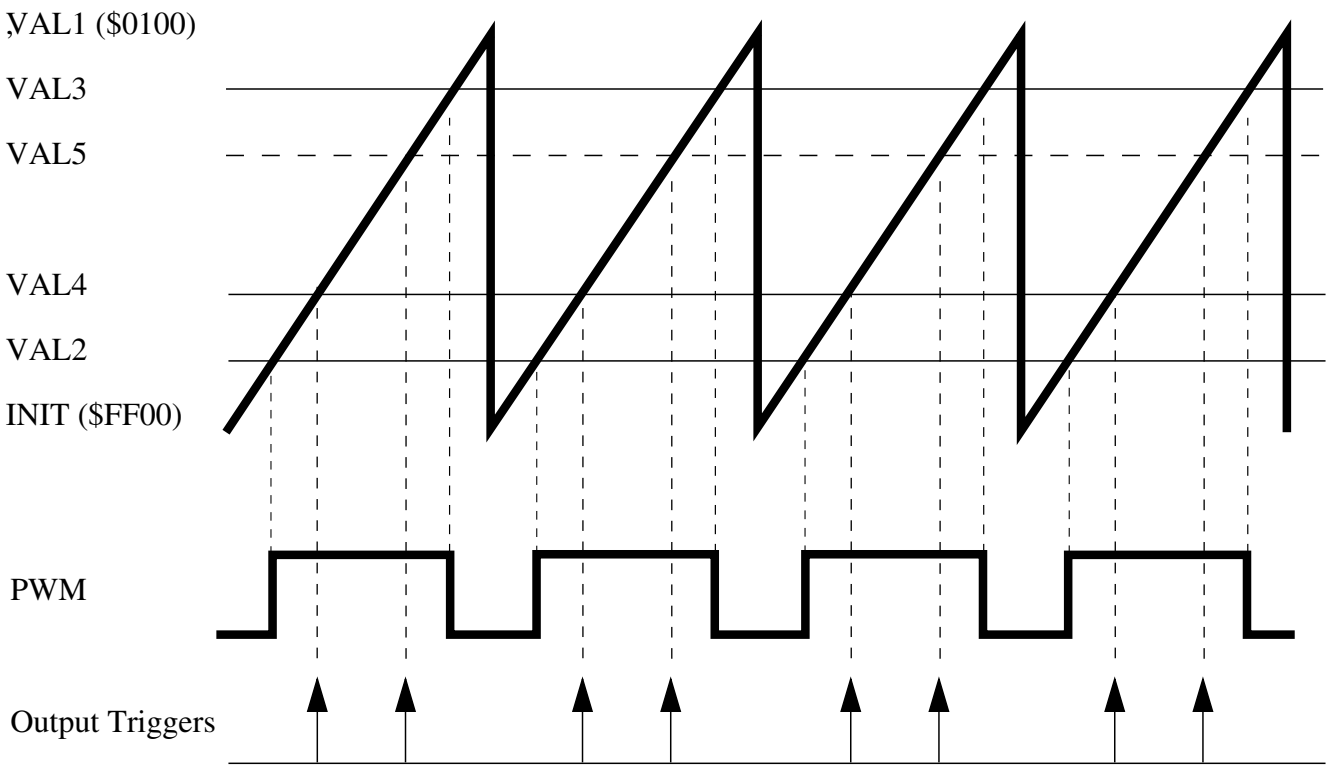


Figure 55-8. Multiple Output Trigger Generation in Hardware

Because each submodule has its own timer, it is possible for each submodule to run at a different frequency. One of the options possible with this PWM module is to have one or more submodules running at a lower frequency, but still synchronized to the timer in submodule0. Figure 55-9 shows how this feature can be used to schedule ADC triggers over multiple PWM cycles. A suggested use for this configuration would be to use the lower-frequency submodule to control the sampling frequency of the software control algorithm where multiple ADC triggers can now be scheduled over the entire sampling period. In Figure 55-9, *all* submodule comparators are shown being used for ADC trigger generation.

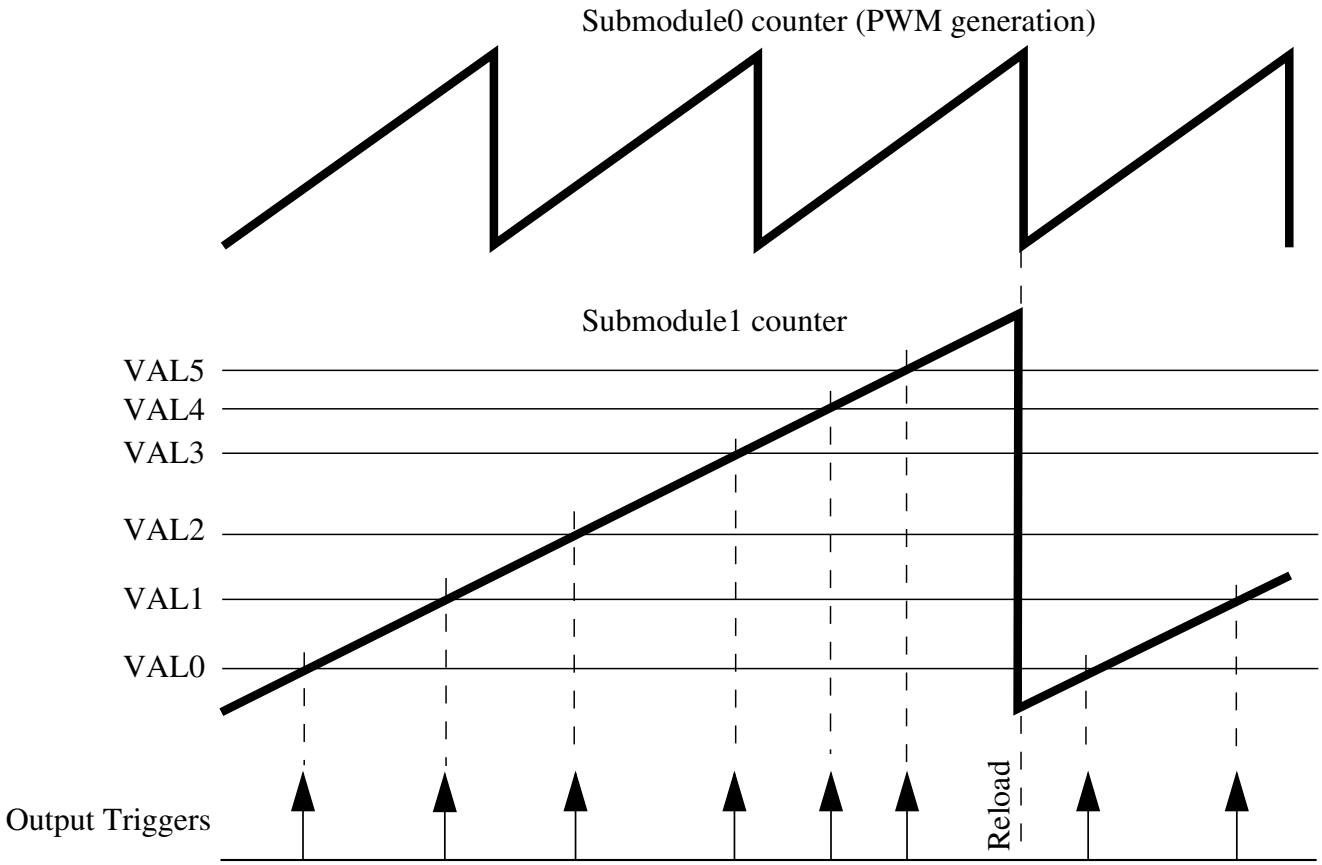


Figure 55-9. Multiple Output Triggers Over Several PWM Cycles

55.3.1.6 Enhanced Capture Capabilities (E-Capture)

When a PWM pin is not being used for PWM generation, it can be used to perform input captures. Recall that for PWM generation BOTH edges of the PWM signal are specified via separate compare register values. When programmed for input capture, both of these registers work on the same pin to capture multiple edges, toggling from one to the other in either a free running or one-shot fashion. By simply programming the desired edge of each capture circuit, period and pulse width of an input signal can easily be measured without the requirement to re-arm the circuit. In addition, each edge of the input signal can clock an 8 bit counter where the counter output is compared to a user specified value (EDGCMP). When the counter output equals EDGCMP, the value of the submodule timer is captured and the counter is automatically reset. This feature allows the module to count a specified number of edge events and then perform a capture and interrupt. The following figure illustrates some of the functionality of the E-Capture circuit.

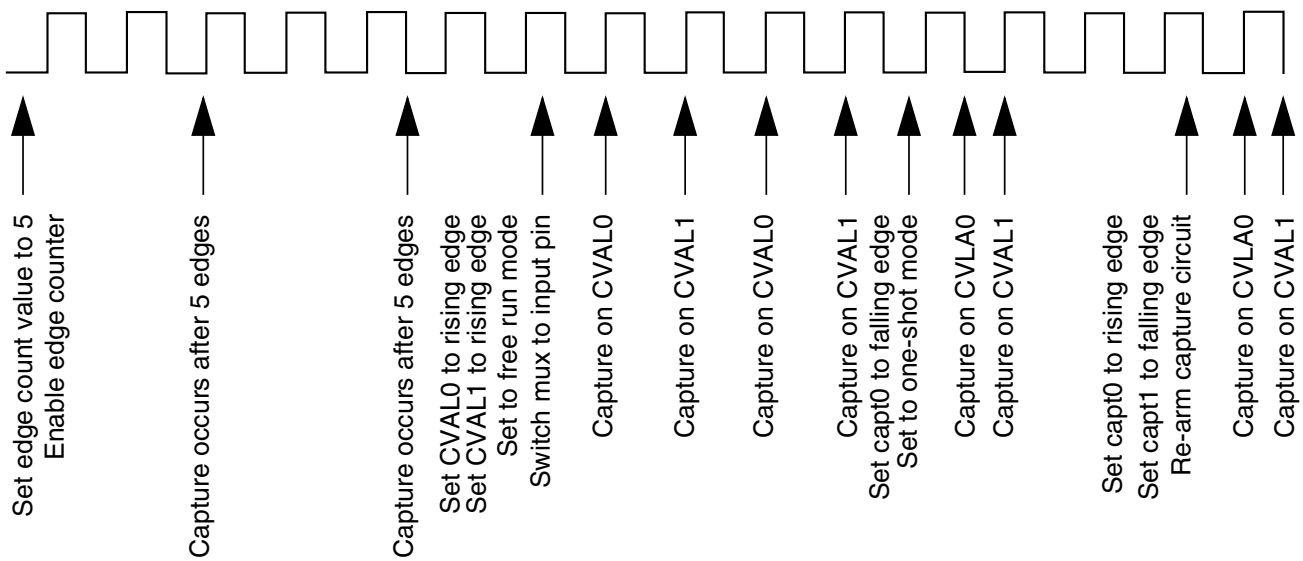


Figure 55-10. Capture Capabilities of the E-Capture Circuit

When a submodule is being used for PWM generation, its timer counts up to the modulus value used to specify the PWM frequency and then is re-initialized. Therefore, using this timer for input captures on one of the other pins (for example, PWM_X) has limited utility since it does not count through all of the numbers and the timer reset represents a discontinuity in the 16 bit number range. However, when measuring a signal that is synchronous to the PWM frequency, the timer modulus range is perfectly suited for the application. Consider the following figure as an example. In this application the output of a PWM power stage is connected to the PWM_X pin that is configured for free running input captures. Specifically, the CVAL0 capture circuitry is programmed for rising edges and the CVAL1 capture circuitry is set for falling edges. This will result in new load pulse width data being acquired every PWM cycle. To calculate the pulse width, simply subtract the CVAL0 register value from the CVAL1 register value. This measurement is extremely beneficial when performing dead-time distortion correction on a half bridge circuit driving an inductive load. Also, these values can be directly compared to the VALx registers responsible for generating the PWM outputs to obtain a measurement of system propagation delays. For details, refer to the separate discussion of deadtime distortion correction.

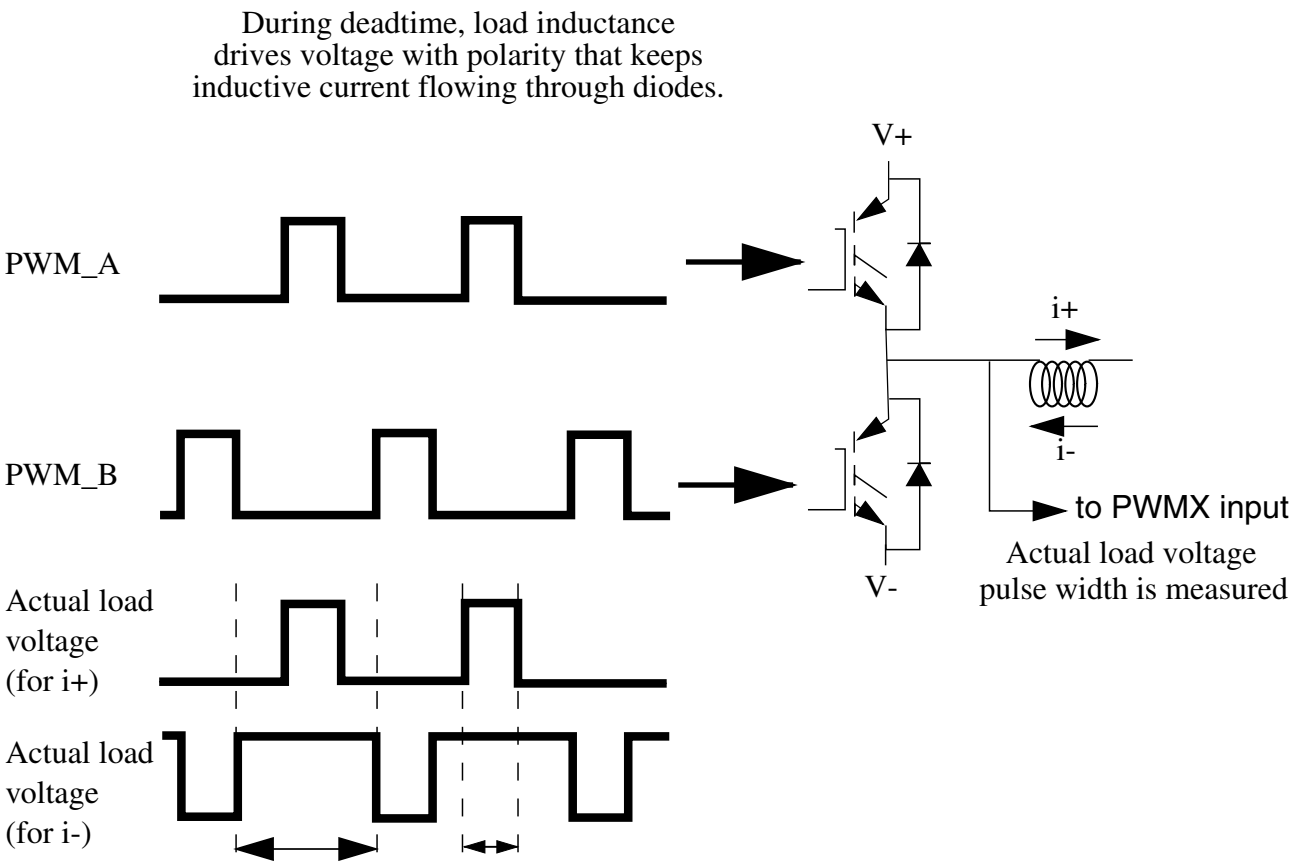


Figure 55-11. Output Pulse Width Measurement Possible with the E-Capture Circuit

55.3.1.7 Synchronous Switching of Multiple Outputs

Before the PWM signals are routed to the output pins, they are processed by a hardware block that permits all submodule outputs to be switched synchronously. This feature can be extremely useful in commutated motor applications where the next commutation state can be laid in ahead of time and then immediately switched to the outputs when the appropriate condition or time is reached. Not only do all the changes occur synchronously on all submodule outputs, but they occur IMMEDIATELY after the trigger event occurs eliminating any interrupt latency.

The synchronous output switching is accomplished via a signal called FORCE_OUT. This signal originates from the local FORCE bit within the submodule, from submodule0, or from external to the PWM module and, in most cases, is supplied from an external timer channel configured for output compare. In a typical application, software sets up the desired states of the output pins in preparation for the next FORCE_OUT event. This selection lays dormant until the FORCE_OUT signal transitions and then all outputs are

switched simultaneously. The signal switching is performed upstream from the deadtime generator so that any abrupt changes that might occur do not violate deadtime on the power stage when in complementary mode.

Figure 55-12 shows a popular application that can benefit from this feature. On a brushless DC motor it is desirable on many cases to spin the motor without need of hall-effect sensor feedback. Instead, the back EMF of the motor phases is monitored and this information is used to schedule the next commutation event. The top waveforms of Figure 55-12 are a simplistic representation of these back EMF signals. Timer compare events (represented by the long vertical lines in the diagram) are scheduled based on the zero crossings of the back-EMF waveforms. The PWM module is configured via software ahead of time with the next state of the PWM pins in anticipation of the compare event. When it happens, the output compare of the timer drives the FORCE_OUT signal which immediately changes the state of the PWM pins to the next commutation state with no software latency.

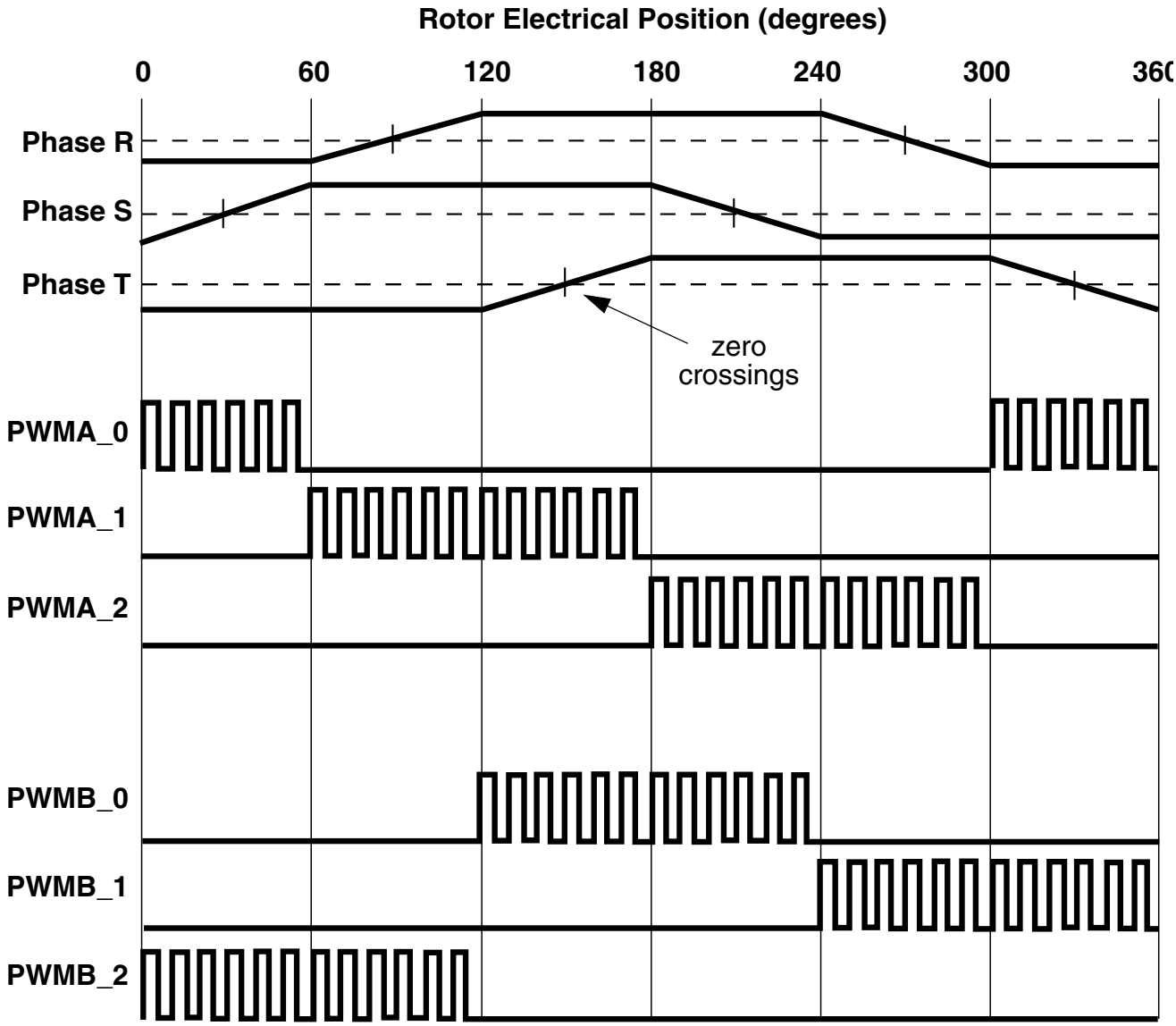


Figure 55-12. Sensorless BLDC Commutation Using the Force Out Function

55.3.2 Functional Details

This section describes the implementation of various sections of the PWM in greater detail.

The following figure is a high-level block diagram of output PWM generation.

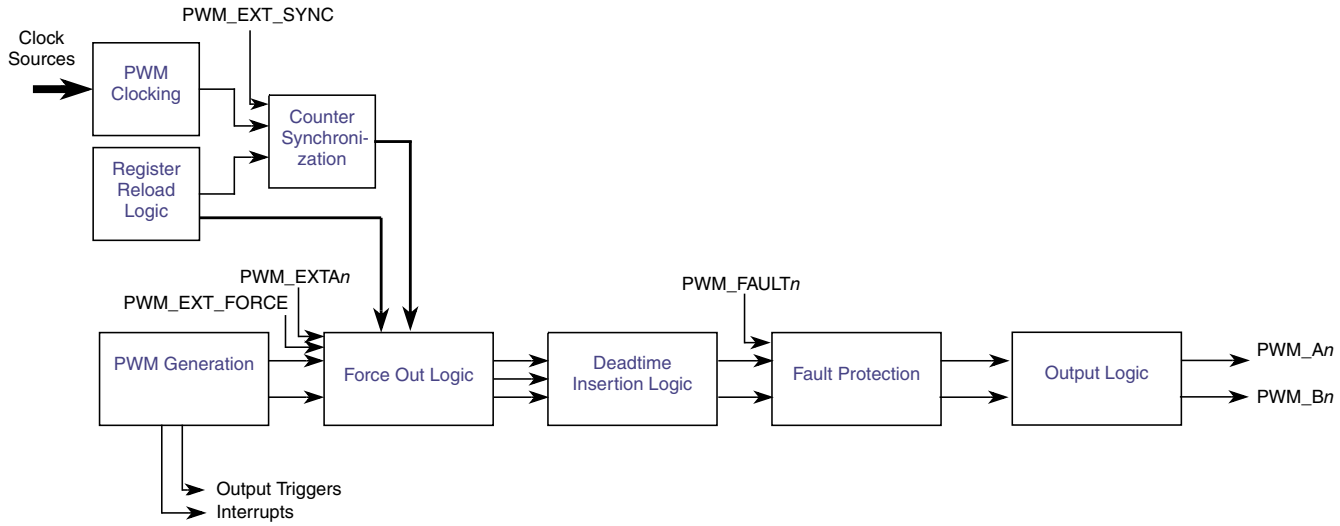


Figure 55-13. High-Level Output PWM Generation Block Diagram

55.3.2.1 PWM Clocking

Figure 55-14 shows the logic used to generate the main counter clock. Each submodule can select between three clock signals: the IPBus clock, EXT_CLK, and AUX_CLK. The EXT_CLK goes to all of the submodules. The AUX_CLK signal is broadcast from submodule0 and can be selected as the clock source by other submodules so that the 8-bit prescaler and MCTRL[RUN] from submodule0 can control all of the submodules.

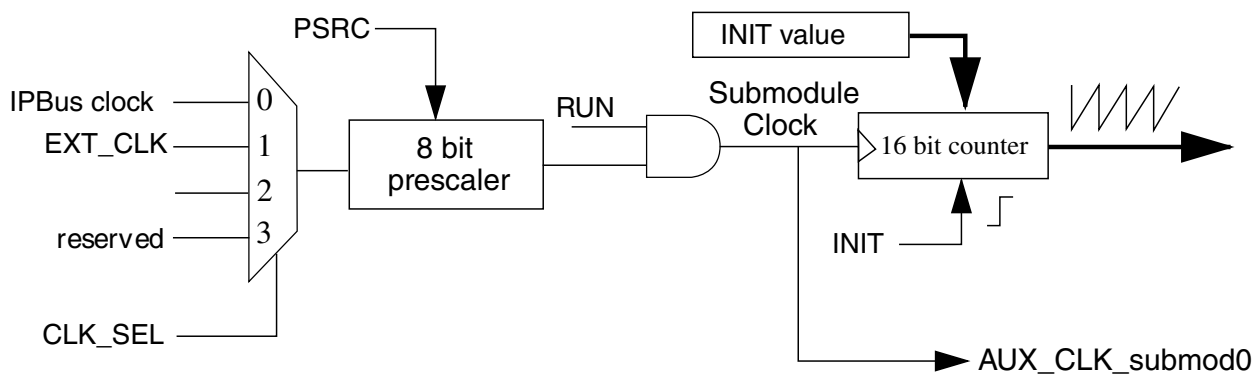


Figure 55-14. Clocking Block Diagram for PWM Submodule 0

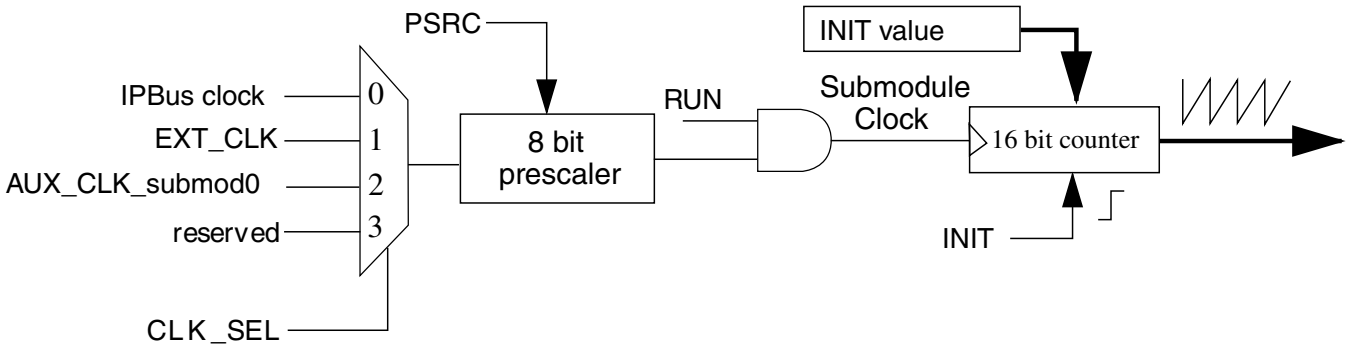


Figure 55-15. Clocking Block Diagram for PWM Submodule 1,2,3

To permit lower PWM frequencies, the prescaler produces the PWM clock frequency by dividing the IPBus clock frequency by 1-128. The prescaler bits, CTRL[PRSC], select the prescaler divisor. This prescaler is buffered and will not be used by the PWM generator until MCTRL[LDOK] is set and a new PWM reload cycle begins or CTRL[LDMOD] is set.

55.3.2.2 Register Reload Logic

The register reload logic is used to determine when the outer set of registers for all double buffered register pairs will be transferred to the inner set of registers. The register reload event can be scheduled to occur every "n" PWM cycles using CTRL[LDFQ] and CTRL[FULL]. A half cycle reload option is also supported (CTRL[HALF]) where the reload can take place in the middle of a PWM cycle. The half cycle point is defined by the VAL0 register and does not have to be exactly in the middle of the PWM cycle.

As illustrated in Figure 55-16 the reload signal from submodule0 can be broadcast as the Master Reload signal allowing the reload logic from submodule0 to control the reload of registers in other submodules.

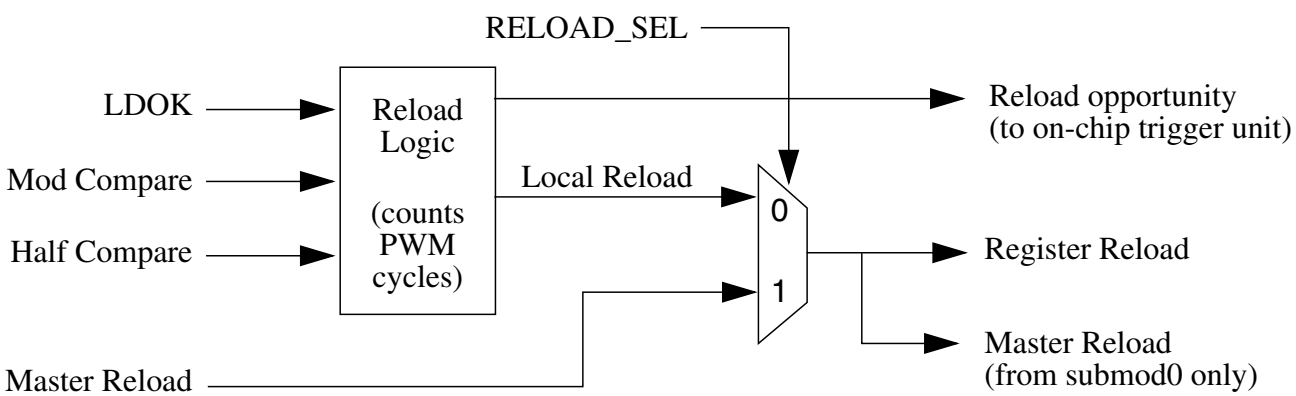


Figure 55-16. Register Reload Logic

55.3.2.3 Counter Synchronization

In the following figure, the 16 bit counter will count up until its output equals VAL1 which is used to specify the counter modulus value. The resulting compare causes a rising edge to occur on the Local Sync signal which is one of four possible sources used to cause the 16 bit counter to be initialized with INIT. If Local Sync is selected as the counter initialization signal, then VAL1 within the submodule effectively controls the timer period (and thus the PWM frequency generated by that submodule) and everything works on a local level.

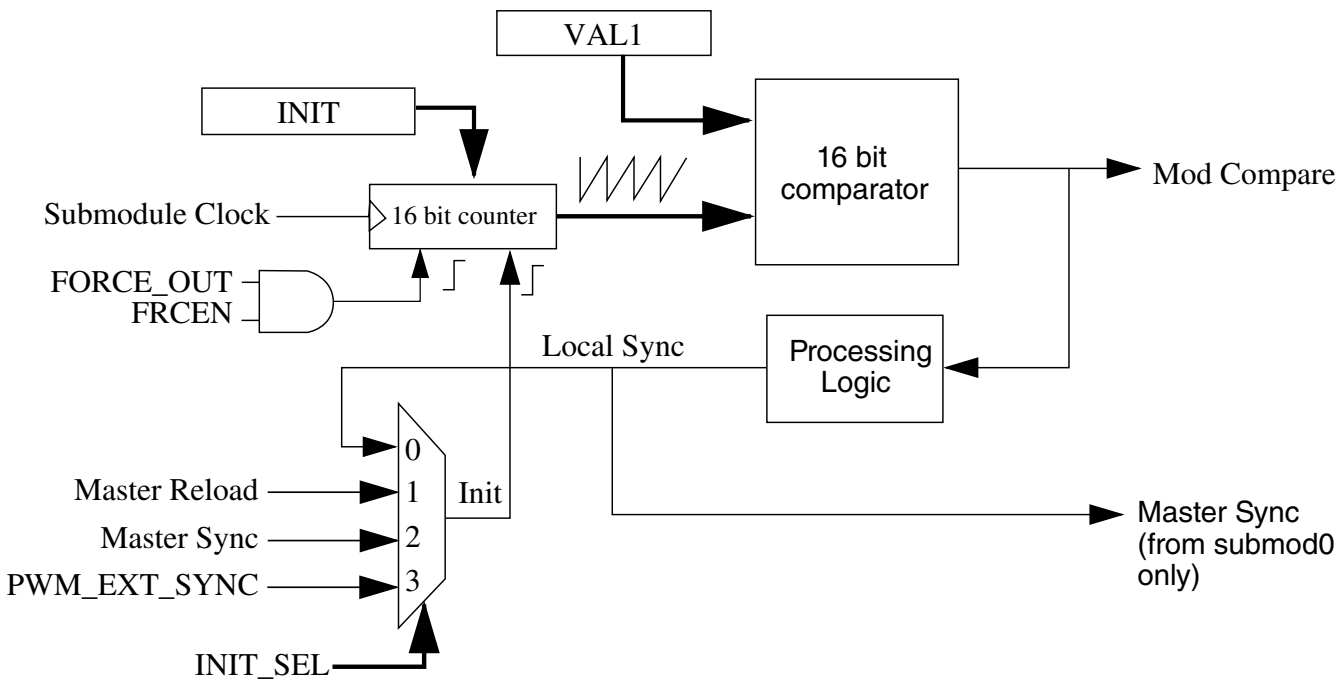


Figure 55-17. Submodule Timer Synchronization

The Master Sync signal originates as the Local Sync from submodule0. If configured to do so, the timer period of any submodule can be locked to the period of the timer in submodule0.

The PWM_EXT_SYNC signal originates on chip or off chip depending on the system architecture. This signal may be selected as the source for counter initialization so that an external source can control the period of all submodules.

If the Master Reload signal is selected as the source for counter initialization, then the period of the counter will be locked to the register reload frequency of submodule0. Since the reload frequency is usually commensurate to the sampling frequency of the software control algorithm, the submodule counter period will therefore equal the sampling period.

As a result, this timer can be used to generate output compares or output triggers over the entire sampling period which may consist of several PWM cycles. The Master Reload signal can only originate from submodule0.

The counter can optionally initialize upon the assertion of the FORCE_OUT signal assuming that CTRL2[FRCEN] is set. As indicated by the preceding figure, this constitutes a second init input into the counter which will cause the counter to initialize regardless of which signal is selected as the counter init signal. A forced initialization will also cause a register reload if MCTRL[LDOK] is set. The FORCE_OUT signal is provided mainly for commutated applications. When PWM signals are commutated on an inverter controlling a brushless DC motor, it is necessary to restart the PWM cycle at the beginning of the commutation interval. This action effectively resynchronizes the PWM waveform to the commutation timing. Otherwise, the average voltage applied to a motor winding integrated over the entire commutation interval will be a function of the timing between the asynchronous commutation event with respect to the PWM cycle. The effect is more critical at higher motor speeds where each commutation interval may consist of only a few PWM cycles. If the counter is not initialized at the start of each commutation interval, the result will be an oscillation caused by the beating between the PWM frequency and the commutation frequency.

55.3.2.4 PWM Generation

Figure 55-18 illustrates how PWM generation is accomplished in each submodule. In each case, two comparators and associated VALx registers are utilized for each PWM output signal. One comparator and VALx register are used to control the turn-on edge, while a second comparator and VALx register control the turn-off edge.

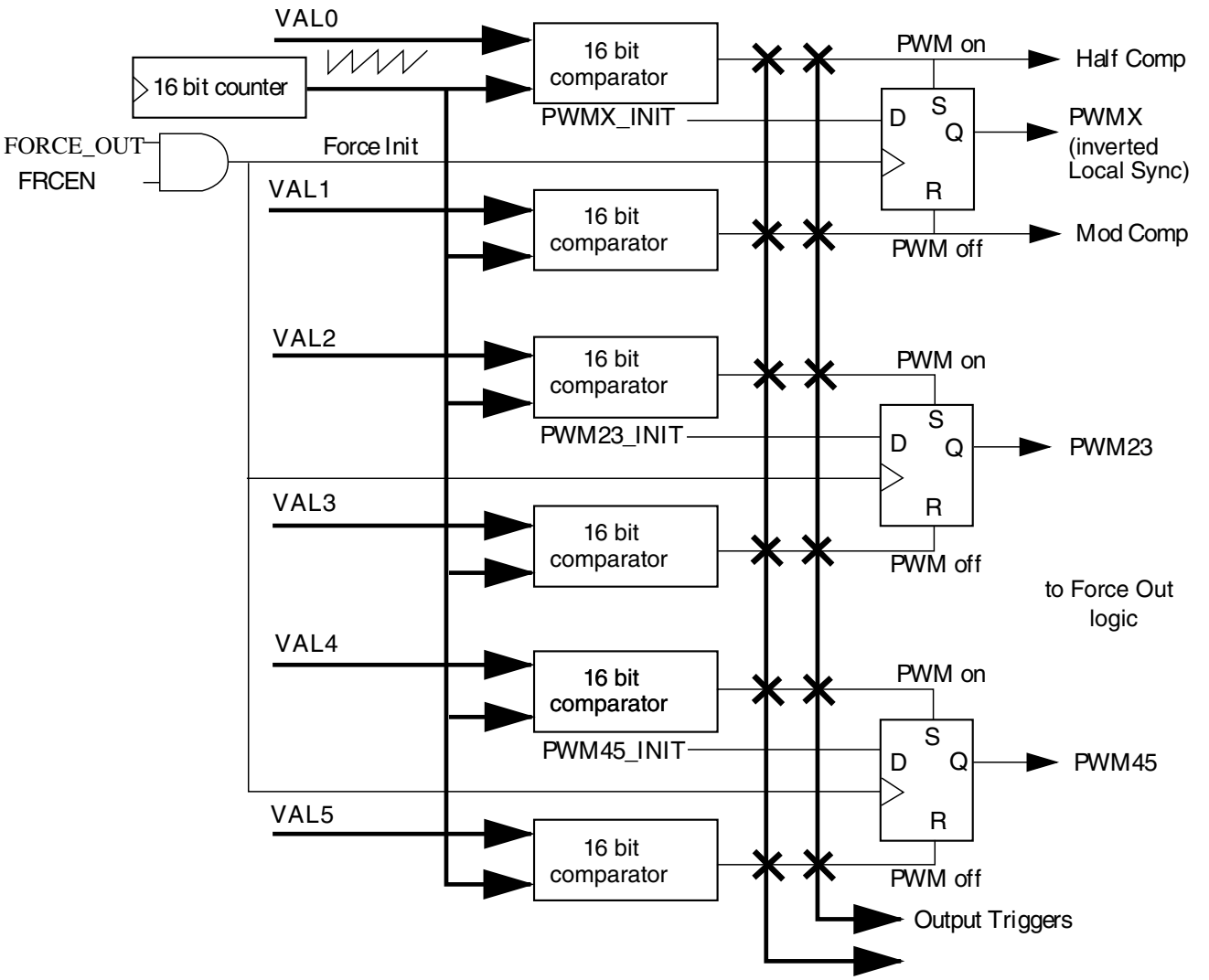


Figure 55-18. PWM Generation Hardware

The generation of the Local Sync signal is performed exactly the same way as the other PWM signals in the submodule. While comparator 0 causes a falling edge of the Local Sync signal, comparator 1 generates a rising edge. Comparator 1 is also hardwired to the reload logic to generate the full cycle reload indicator.

If VAL1 is controlling the modulus of the counter and VAL0 is half of the VAL1 register minus the INIT value, then the half cycle reload pulse will occur exactly half way through the timer count period and the Local Sync will have a 50% duty cycle. On the other hand, if the VAL1 and VAL0 registers are not required for register reloading or counter initialization, they can be used to modulate the duty cycle of the Local Sync signal, effectively turning it into an auxiliary PWM signal (PWM_X) assuming that the PWM_X pin is not being used for another function such as input capture or deadtime distortion correction. Including the Local Sync signal, each submodule is capable of generating three PWM signals where software has complete control over each edge of each of the signals.

If the comparators and edge value registers are not required for PWM generation, they can also be used for other functions such as output compares, generating output triggers, or generating interrupts at timed intervals.

The 16-bit comparators shown in Figure 55-18 are "equal to" comparators. In addition, if both the set and reset of the flip-flop are asserted, then the flop output goes to 0.

55.3.2.5 Output Compare Capabilities

By using the VALx registers in conjunction with the submodule timer and 16 bit comparators, buffered output compare functionality can be achieved with no additional hardware required. Specifically, the following output compare functions are possible:

- An output compare sets the output high
- An output compare sets the output low
- An output compare generates an interrupt
- An output compare generates an output trigger

In PWM generation, an output compare is initiated by programming a VALx register for a timer compare, which in turn causes the output of the D flip-flop to either set or reset. For example, if an output compare is desired on the PWM_A signal that sets it high, VAL2 would be programmed with the counter value where the output compare should take place. However, to prevent the D flip-flop from being reset again after the compare has occurred, the VAL3 register must be programmed to a value outside of the modulus range of the counter. Therefore, a compare that would result in resetting the D flip-flop output would never occur. Conversely, if an output compare is desired on the PWM_A signal that sets it low, the VAL3 register is programmed with the appropriate count value and the VAL2 register is programmed with a value outside the counter modulus range. Regardless of whether a high compare or low compare is programmed, an interrupt or output trigger can be generated when the compare event occurs.

55.3.2.6 Force Out Logic

For each submodule, software can select between eight signal sources for the FORCE_OUT signal: local CTRL2[FORCE], the Master Force signal from submodule0, the local Reload signal, the Master Reload signal from submodule0, the Local Sync signal, the Master Sync signal from submodule0, the EXT_SYNC signal from on- or off-chip, or the EXT_FORCE signal from on- or off-chip depending on the chip architecture. The local signals are used when the user simply wants to change the signals on the output

pins of the submodule without regard for synchronization with other submodules. However, if it is required that all signals on all submodule outputs change at the same time, the Master, EXT_SYNC, or EXT_FORCE signals should be selected.

Figure 55-19 illustrates the Force logic. The SEL23 and SEL45 fields each choose from one of four signals that can be supplied to the submodule outputs: the PWM signal, the inverted PWM signal, a binary level specified by software via the OUT23 and OUT45 bits, or the PWM_EXTA or PWM_EXTB alternate external control signals. The selection can be determined ahead of time and, when a FORCE_OUT event occurs, these values are presented to the signal selection mux that immediately switches the requested signal to the output of the mux for further processing downstream.

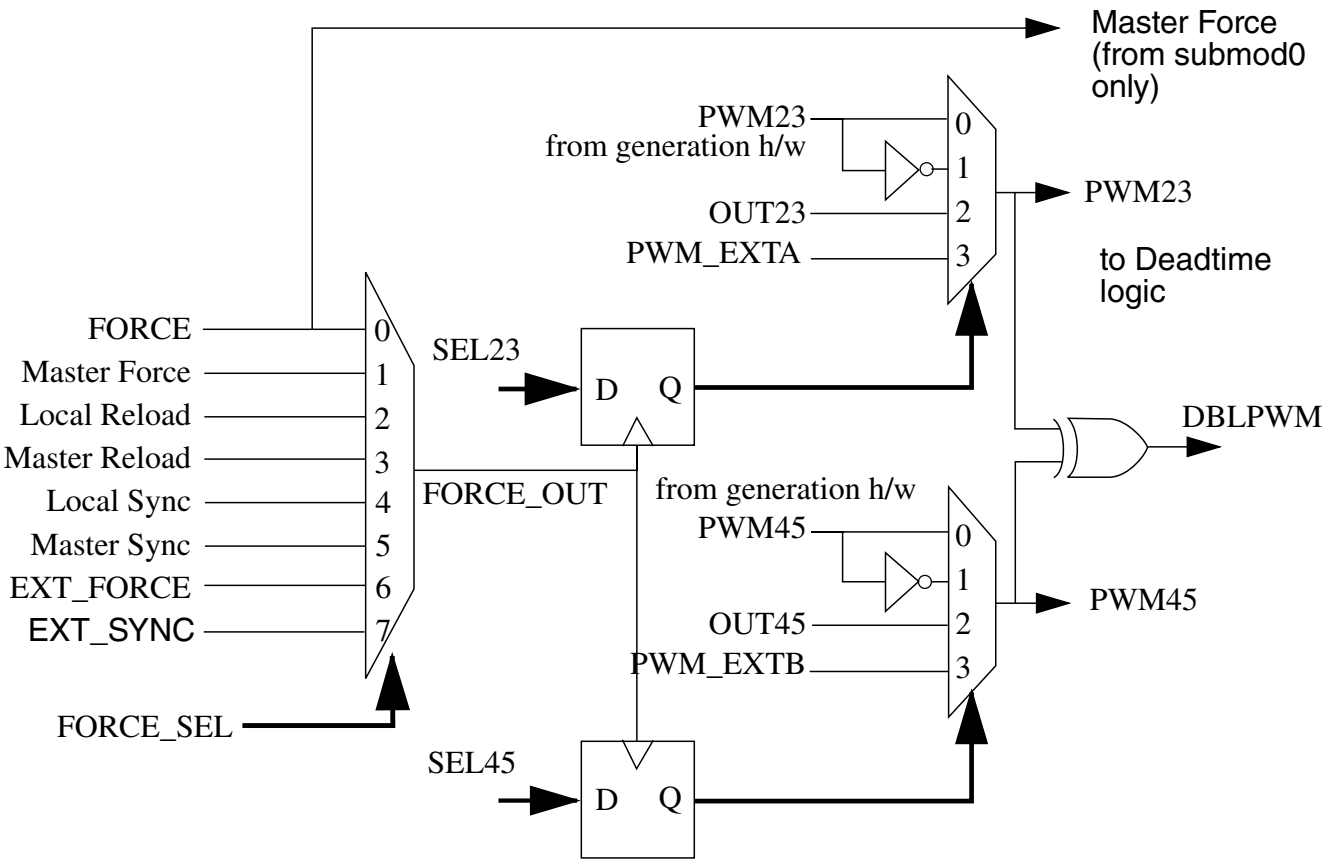


Figure 55-19. Force Out Logic

The local CTRL2[FORCE] signal of submodule0 can be broadcast as the Master Force signal to other submodules. This feature allows the CTRL2[FORCE] of submodule0 to synchronously update all of the submodule outputs at the same time. The EXT_FORCE signal originates from outside the PWM module from a source such as a timer or digital comparators in the Analog-to-Digital Converter.

55.3.2.7 Independent or Complementary Channel Operation

Writing a logic one to CTRL2[INDEP] configures the pair of PWM outputs as two independent PWM channels. Each PWM output is controlled by its own VALx pair operating independently of the other output.

Writing a logic zero to CTRL2[INDEP] configures the PWM output as a pair of complementary channels. The PWM pins are paired as shown in Figure 55-20 in complementary channel operation. Which signal is connected to the output pin (PWM23 or PWM45) is determined by MCTRL[IPOL].

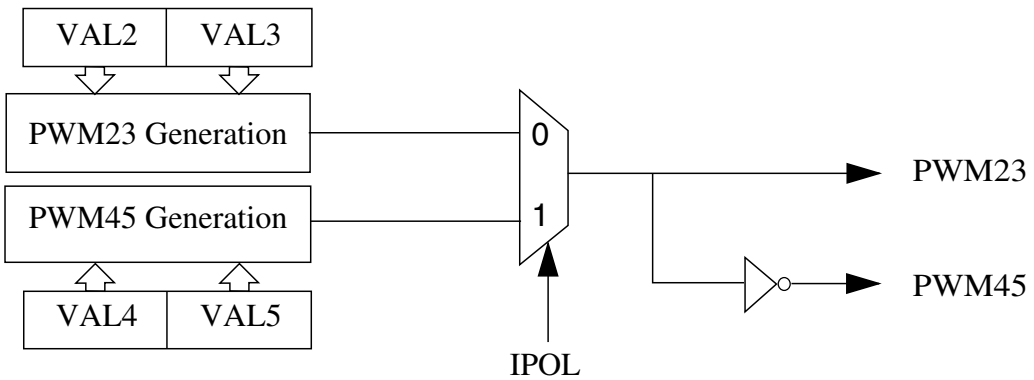


Figure 55-20. Complementary Channel Pair

The complementary channel operation is for driving top and bottom transistors in a motor drive circuit, such as the one in Figure 55-21.

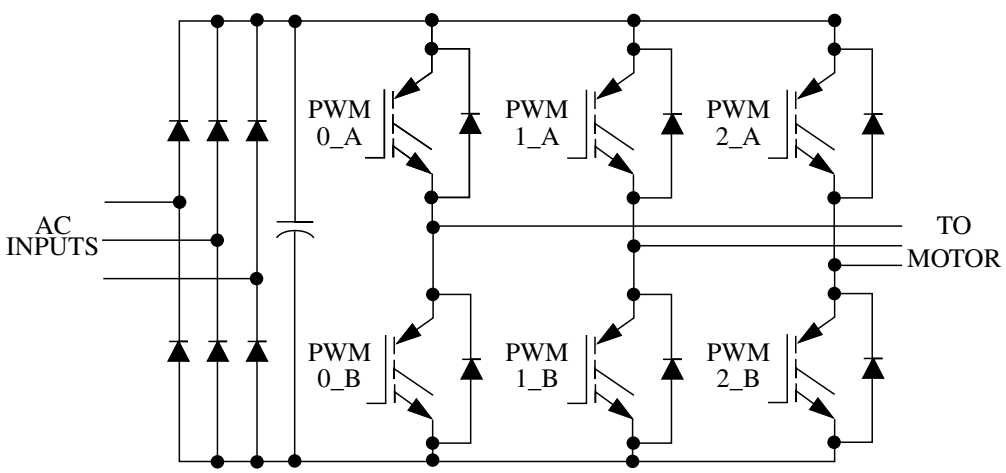


Figure 55-21. Typical 3 Phase AC Motor Drive

Complementary operation allows the use of the deadtime insertion feature.

55.3.2.8 Deadtime Insertion Logic

The following figure shows the deadtime insertion logic of each submodule which is used to create non-overlapping complementary signals when not in independent mode.

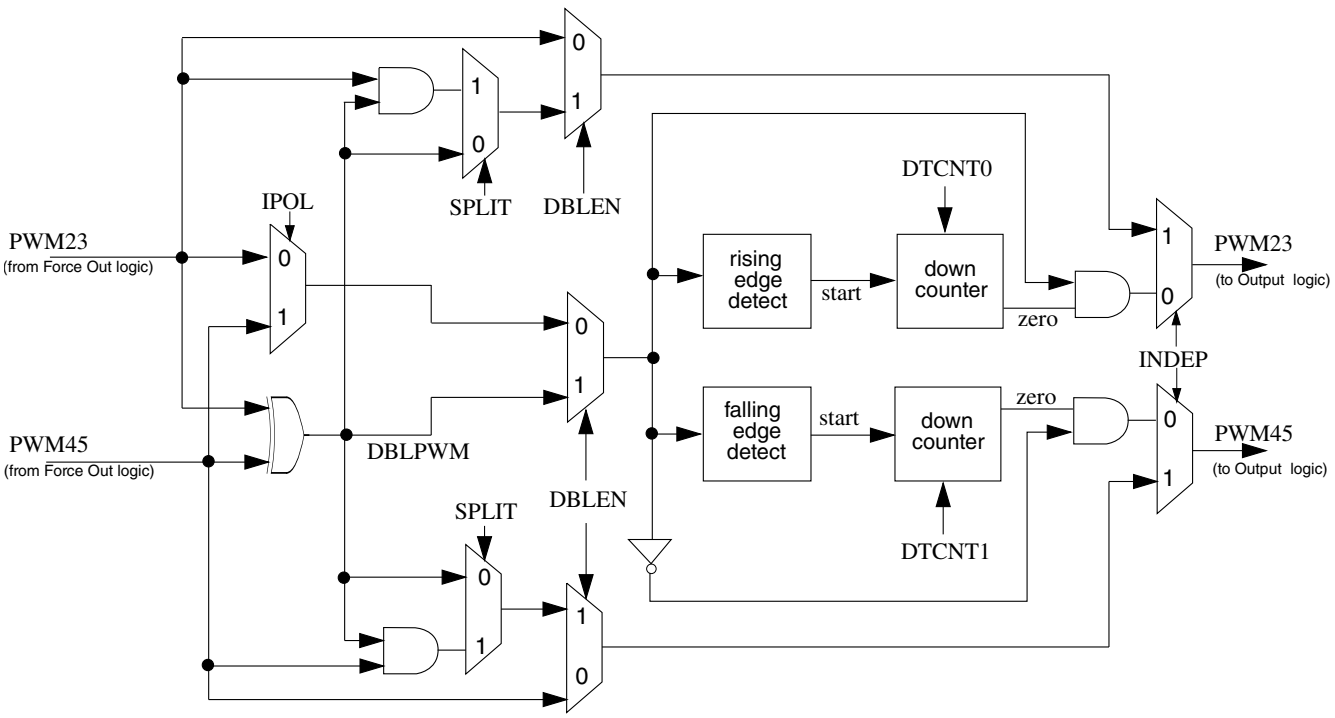


Figure 55-22. Deadtime Insertion Logic

While in the complementary mode, a PWM pair can be used to drive top/bottom transistors, as shown in the figure. When the top PWM channel is active, the bottom PWM channel is inactive, and vice versa.

Note

To avoid short circuiting the DC bus and endangering the transistor, there must be no overlap of conducting intervals between top and bottom transistor. But the transistor's characteristics may make its switching-off time longer than switching-on time. To avoid the conducting overlap of top and bottom transistors, deadtime needs to be inserted in the switching period, as illustrated in the following figure.

The deadtime generators automatically insert software-selectable activation delays into the pair of PWM outputs. The deadtime registers (DTCNT0 and DTCNT1) specify the number of IPBus clock cycles to use for deadtime delay. Every time the deadtime generator inputs change state, deadtime is inserted. Deadtime forces both PWM outputs in the pair to the inactive state.

When deadtime is inserted in complementary PWM signals connected to an inverter driving an inductive load, the PWM waveform on the inverter output will have a different duty cycle than what appears on the output pins of the PWM module. This results in a distortion in the voltage applied to the load. A method of correcting this, adding to or subtracting from the PWM value used, is discussed next.

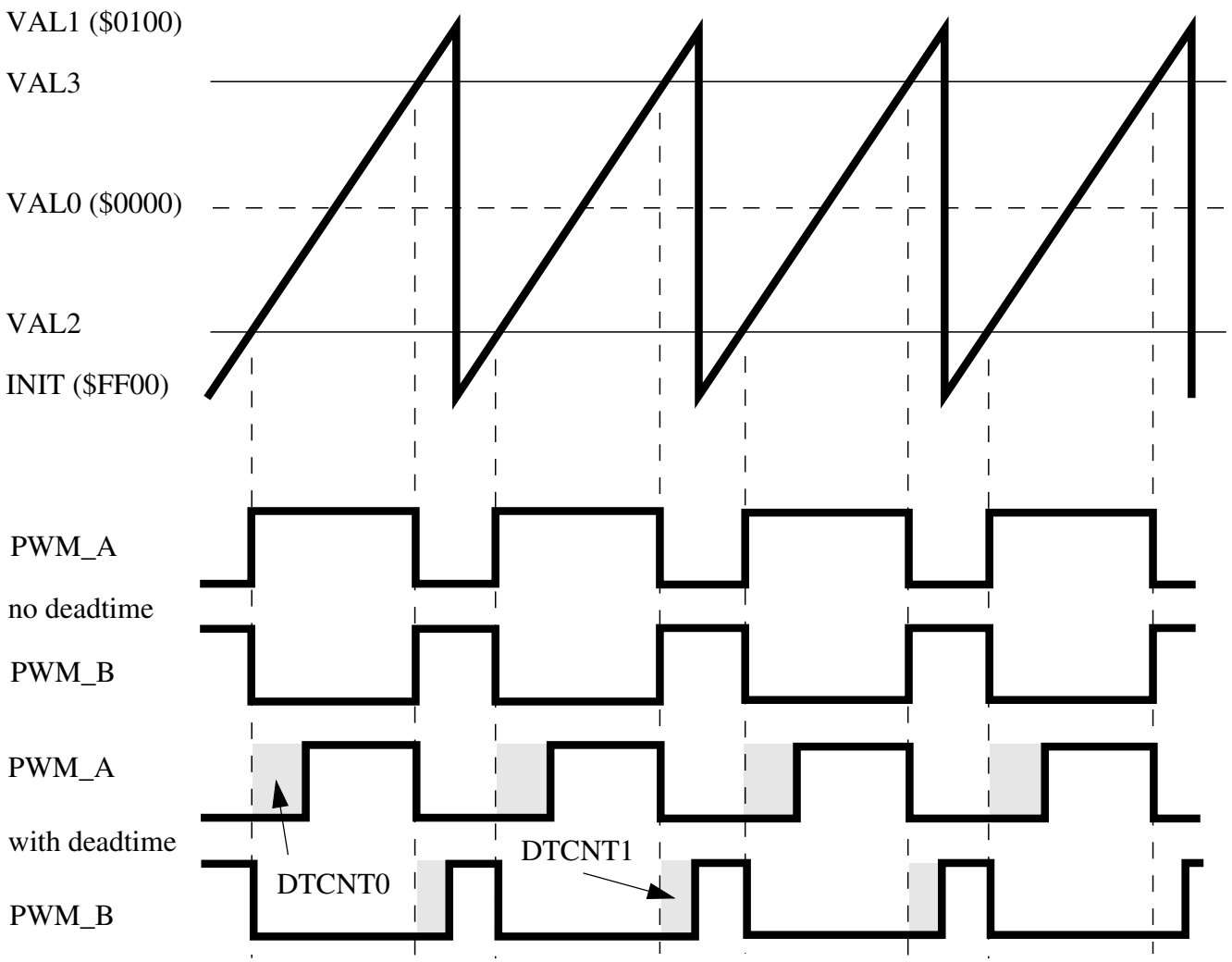


Figure 55-23. Deadtime Insertion

55.3.2.8.1 Top/Bottom Correction

In complementary mode, either the top or the bottom transistor controls the output voltage. However, deadtime has to be inserted to avoid overlap of conducting interval between the top and bottom transistor. Both transistors in complementary mode are off during deadtime, allowing the output voltage to be determined by the current status of load and introduce distortion in the output voltage. See the following figure. On AC induction motors running open-loop, the distortion typically manifests itself as poor low-speed performance, such as torque ripple and rough operation.

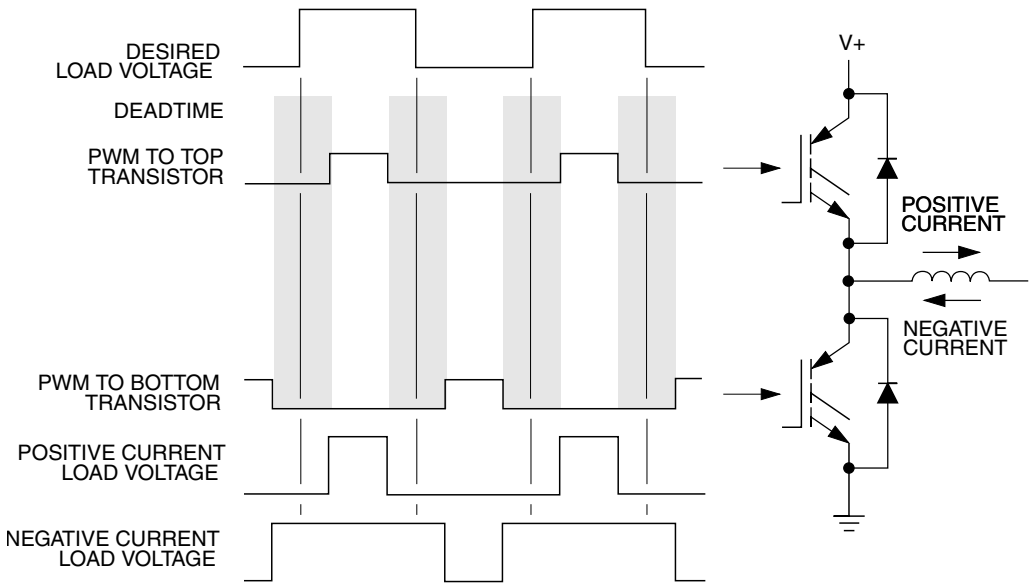


Figure 55-24. Deadtime Distortion

During deadtime, load inductance distorts output voltage by keeping current flowing through the diodes. This deadtime current flow creates a load voltage that varies with current direction. With a positive current flow, the load voltage during deadtime is equal to the bottom supply, putting the top transistor in control. With a negative current flow, the load voltage during deadtime is equal to the top supply putting the bottom transistor in control.

Remembering that the original PWM pulse widths where shortened by deadtime insertion, the averaged sinusoidal output will be less than the desired value. However, when deadtime is inserted, it creates a distortion in the motor current waveform inverter outputs. This distortion is aggravated by dissimilar turn-on and turn-off delays of each of the transistors. By giving the PWM module information on which transistor is controlling at a given time this distortion can be corrected.

For a typical circuit in complementary channel operation, only one of the transistors will be effective in controlling the output voltage at any given time. This depends on the direction of the motor inverter current for that pair, as the preceding figure shows. To correct distortion one of two different factors must be added to the desired PWM value, depending on whether the top or bottom transistor is controlling the output voltage. Therefore, the software is responsible for calculating both compensated PWM values prior to placing them in the VALx registers. Either the VAL2/VAL3 or the VAL4/VAL5 register pair controls the pulse width at any given time. For a given PWM pair, whether the VAL2/VAL3 or VAL4/VAL5 pair is active depends on either:

- The state of the current status pin, PWMX, for that driver
- The state of the odd/even correction bit, MCTRL[IPOL], for that driver

To correct deadtime distortion, software can decrease or increase the value in the appropriate VALx register.

- In edge-aligned operation, decreasing or increasing the PWM value by a correction value equal to the deadtime typically compensates for deadtime distortion.
- In center-aligned operation, decreasing or increasing the PWM value by a correction value equal to one-half the deadtime typically compensates for deadtime distortion.

55.3.2.8.2 Manual Correction

To detect the current status, the voltage on each PWMX pin is sampled twice in a PWM period, at the end of each deadtime. The value is stored in CTRL[DT]. CTRL[DT] is a timing marker especially indicating when to toggle between PWM value registers. Software can then set MCTRL[IPOL] to switch between VAL2/VAL3 and VAL4/VAL5 register pairs according to CTRL[DT] values.

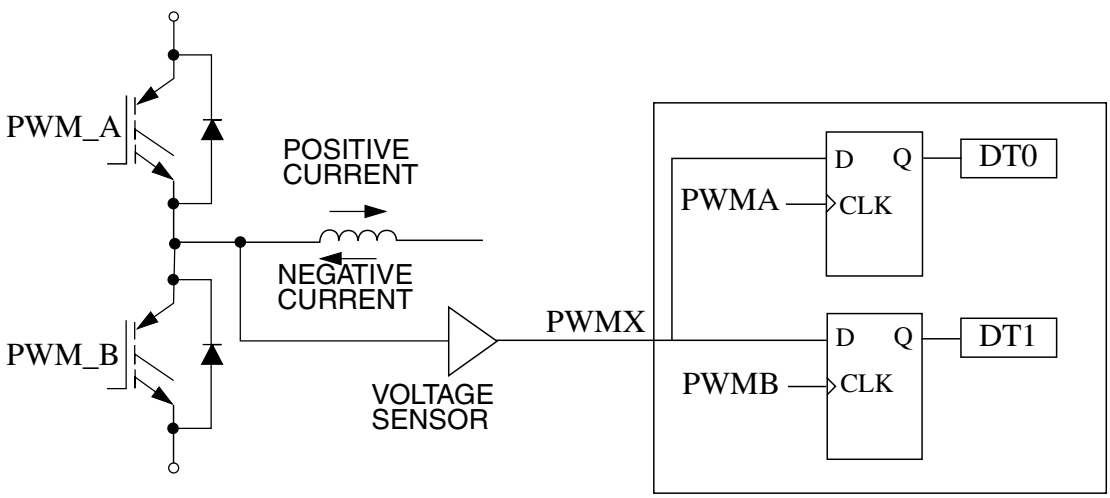


Figure 55-25. Current-status Sense Scheme for Deadtime Correction

Both D flip-flops latch low, CTRL[DT] = 00, during deadtime periods if current is large and flowing out of the complementary circuit. See the preceding figure. Both D flip-flops latch the high, CTRL[DT] = 11, during deadtime periods if current is also large and flowing into the complementary circuit.

However, under low-current, the output voltage of the complementary circuit during deadtime is somewhere between the high and low levels. The current cannot free-wheel through the opposition anti-body diode, regardless of polarity, giving additional distortion when the current crosses zero. **Sampled results will be CTRL[DT] = b10. Thus, the best time to change one PWM value register to another is just before the current zero crossing.**

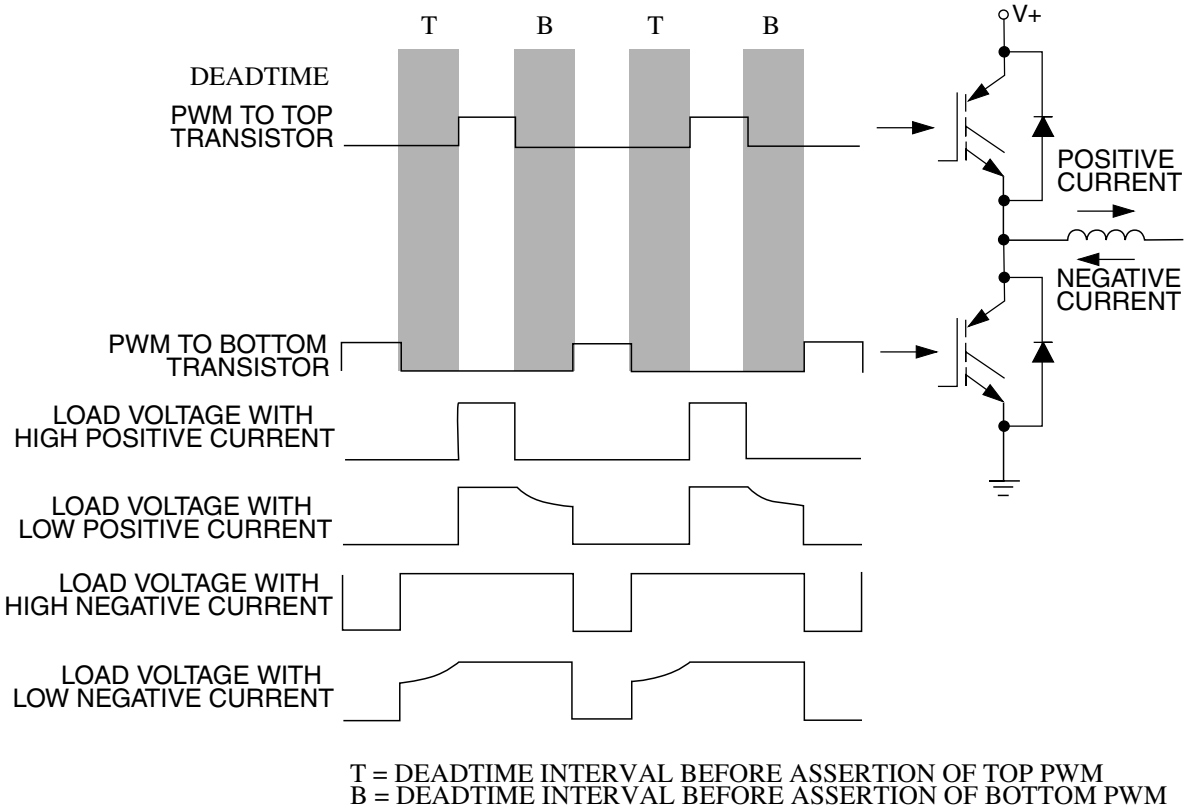


Figure 55-26. Output Voltage Waveforms

55.3.2.9 Fractional Delay Logic

For applications where more resolution than a single IPBus clock period is needed, the fractional delay logic can be used to achieve fine resolution on the rising and falling edges of the PWM_A and PWM_B outputs and fine resolution for the PWM period. Enable the use of the fractional delay logic by setting FRCTRL[FRACx_EN]. The FRACVALx registers act as a fractional clock cycle addition to the turn on and turn off count specified by the VAL2, VAL3, VAL4, or VAL5 registers. The FRACVAL1 register acts as a fractional increase in the PWM period as defined by VAL1. If FRACVAL1 is programmed to a non-zero value, then the largest value for the VAL1 register is 0xFFFE for unsigned usage or 0x7FFE for signed usage. This limit is needed in order to avoid counter rollovers when accumulating the fractional additional period.

The results of the fractional delay logic depend on whether or not the PWM submodule has an analog NanoEdge placer block available.

55.3.2.9.1 Fractional Delay Logic with NanoEdge Placement Block

Using the NanoEdge placer block requires that the IPBus clock to the PWM be set at a defined frequency. The NanoEdge placer is powered up by setting FRCTRL[FRAC_PU]. Enable fine edge control on the various PWM edges by setting FRCTRL[FRACx_EN]. The fractional values in the FRACVALx registers allow placing the PWM edge or PWM period to a granularity of 1/32 of the IPBus clock period. For example, if you desire the rising edge of the PWMA output to occur at a count of 12.25, then program VAL2 with 0x000C and FRACVAL2 with 0x4000. Using FRACVAL1 will adjust the PWM period with the same granularity of 1/32 of a clock period.

If the FRCTRL[FRAC_PU] bits in all of the submodules are clear, then the NanoEdge placer is powered down, and alternate clock frequencies can be used without the NanoEdge placement feature.

55.3.2.10 Output Logic

The following figure shows the output logic of each submodule including how each PWM output has individual fault disabling, polarity control, and output enable. This allows for maximum flexibility when interfacing to the external circuitry.

The PWM23 and PWM45 signals which are output from the deadtime logic (refer to the figure) are positive true signals. In other words, a high level on these signals should result in the corresponding transistor in the PWM inverter being turned ON. The voltage level required at the PWM output pin to turn the transistor ON or OFF is a function of the logic between the pin and the transistor. Therefore, it is imperative that the user program OCTRL[POLA] and OCTRL[POLB] before enabling the output pins. A fault condition can result in the PWM output being tristated, forced to a logic 1, or forced to a logic 0 depending on the values programmed into the OCTRL[PWMxFS] fields.

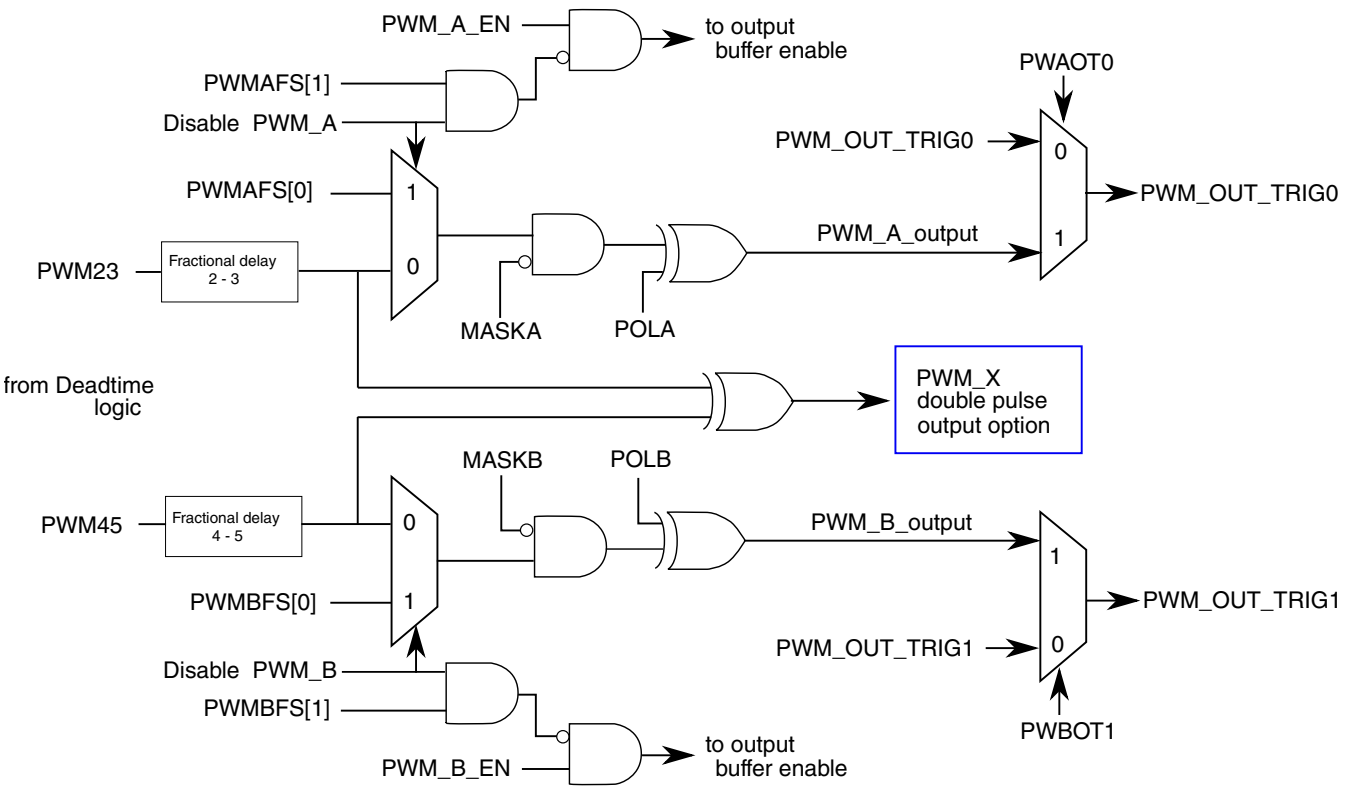


Figure 55-27. Output Logic

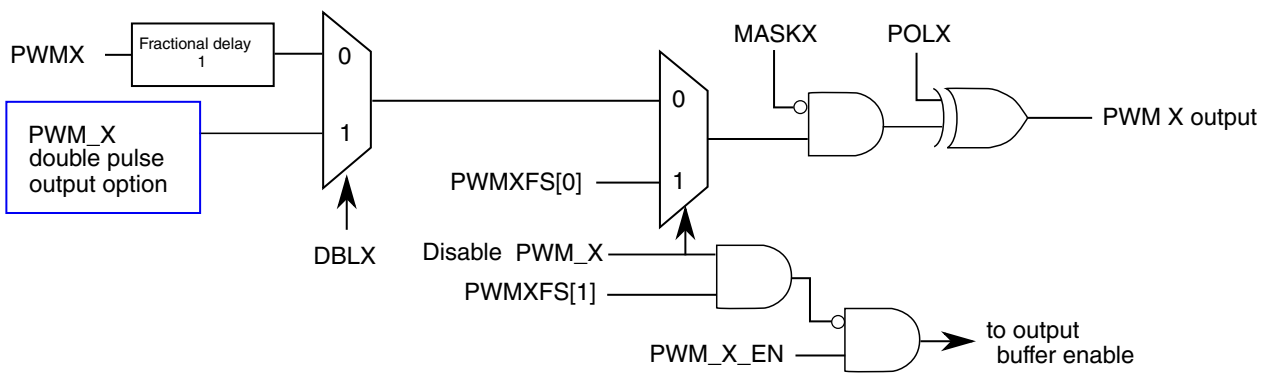


Figure 55-28. Output Logic continued

55.3.2.11 E-Capture

Commensurate with the idea of controlling both edges of an output signal, the Enhanced Capture (E-Capture) logic is designed to measure both edges of an input signal. As a result, when a submodule pin is configured for input capture, the CVALx registers associated with that pin are used to record the edge values.

The following figure is a block diagram of the E-Capture circuit. Upon entering the pin input, the signal is split into two paths. One goes straight to a mux input where software can select to pass the signal directly to the capture logic for processing. The other path connects the signal to an 8 bit counter which counts both the rising and falling edges of the signal. The output of this counter is compared to an 8 bit value that is specified by the user (EDGCMPx) and when the two values are equal, the comparator generates a pulse that resets the counter. This pulse is also supplied to the mux input where software can select it to be processed by the capture logic. This feature permits the E-Capture circuit to count up to 256 edge events before initiating a capture event. this feature is useful for dividing down high frequency signals for capture processing so that capture interrupts don't overwhelm the CPU. Also, this feature can be used to simply generate an interrupt after "n" events have been counted.

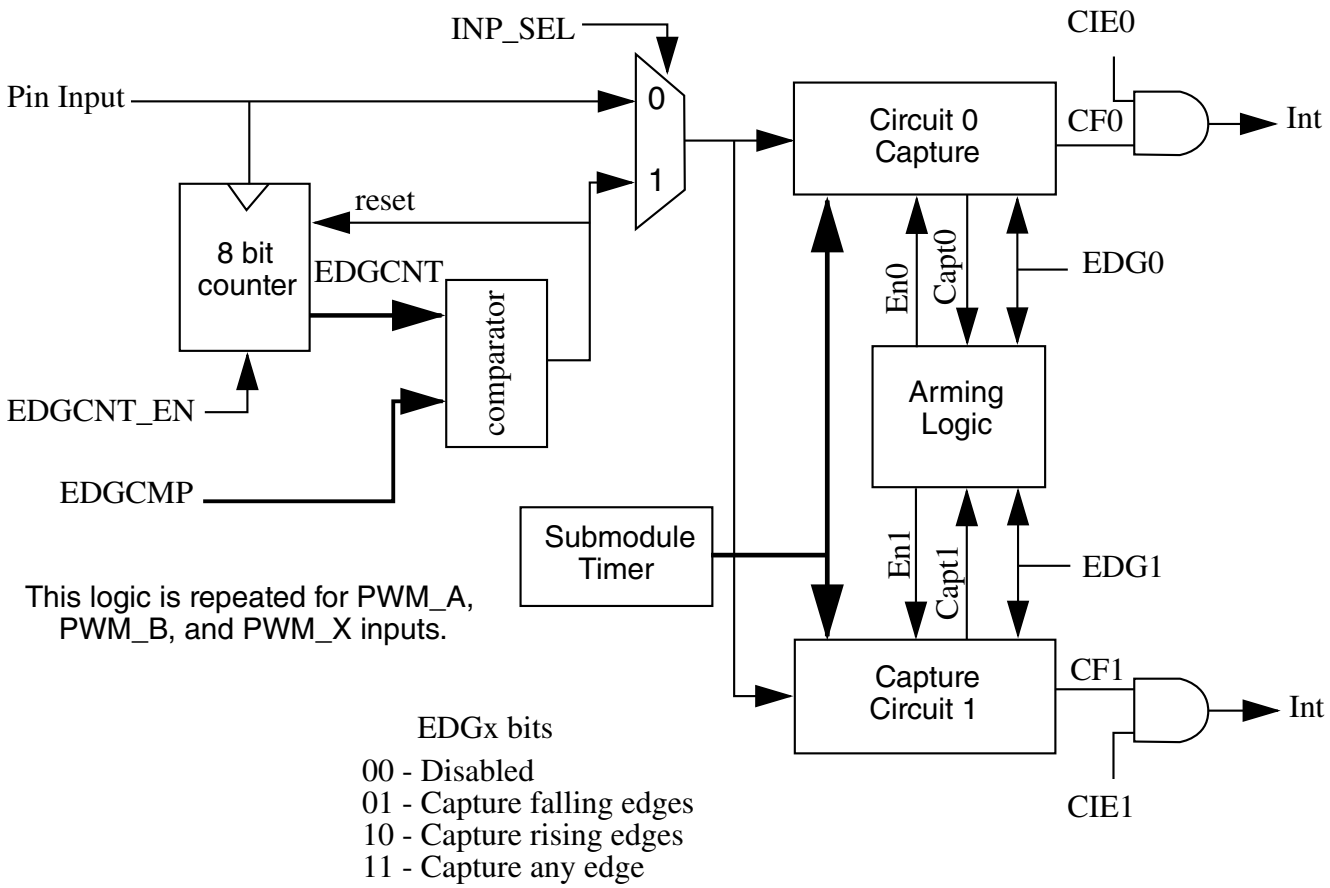


Figure 55-29. Enhanced Capture (E-Capture) Logic

Based on the mode selection, the mux selects either the pin input or the compare output from the count/compare circuit to be processed by the capture logic. The selected signal is routed to two separate capture circuits which work in tandem to capture sequential edges of the signal. The type of edge to be captured by each circuit is determined by CAPTCTRLx[EDGx1] and CAPTCTRLx[EDGx0], whose functionality is listed in the preceding figure. Also, controlling the operation of the capture circuits is the arming

logic which allows captures to be performed in a free running (continuous) or one shot fashion. In free running mode, the capture sequences will be performed indefinitely. If both capture circuits are enabled, they will work together in a ping-pong style where a capture event from one circuit leads to the arming of the other and vice versa. In one shot mode, only one capture sequence will be performed. If both capture circuits are enabled, capture circuit 0 is first armed and when a capture event occurs, capture circuit 1 is armed. Once the second capture occurs, further captures are disabled until another capture sequence is initiated. Both capture circuits are also capable of generating an interrupt to the CPU.

55.3.2.12 Fault Protection

Fault protection can control any combination of PWM output pins. Faults are generated by a logic one on any of the FAULTx pins. This polarity can be changed via FCTRL[FLVL]. Each FAULTx pin can be mapped arbitrarily to any of the PWM outputs. When fault protection hardware disables PWM outputs, the PWM generator continues to run, only the output pins are forced to logic 0, logic 1, or tristated depending the values of OCTRL[PWMxFS].

The fault decoder disables PWM pins selected by the fault logic and the disable mapping (DISMAPn) registers. The following figure shows an example of the fault disable logic. Each bank of bits in DISMAPn control the mapping for a single PWM pin. See the following table.

The fault protection is enabled even when the PWM module is not enabled; therefore, a fault will be latched in and must be cleared in order to prevent an interrupt when the PWM is enabled.

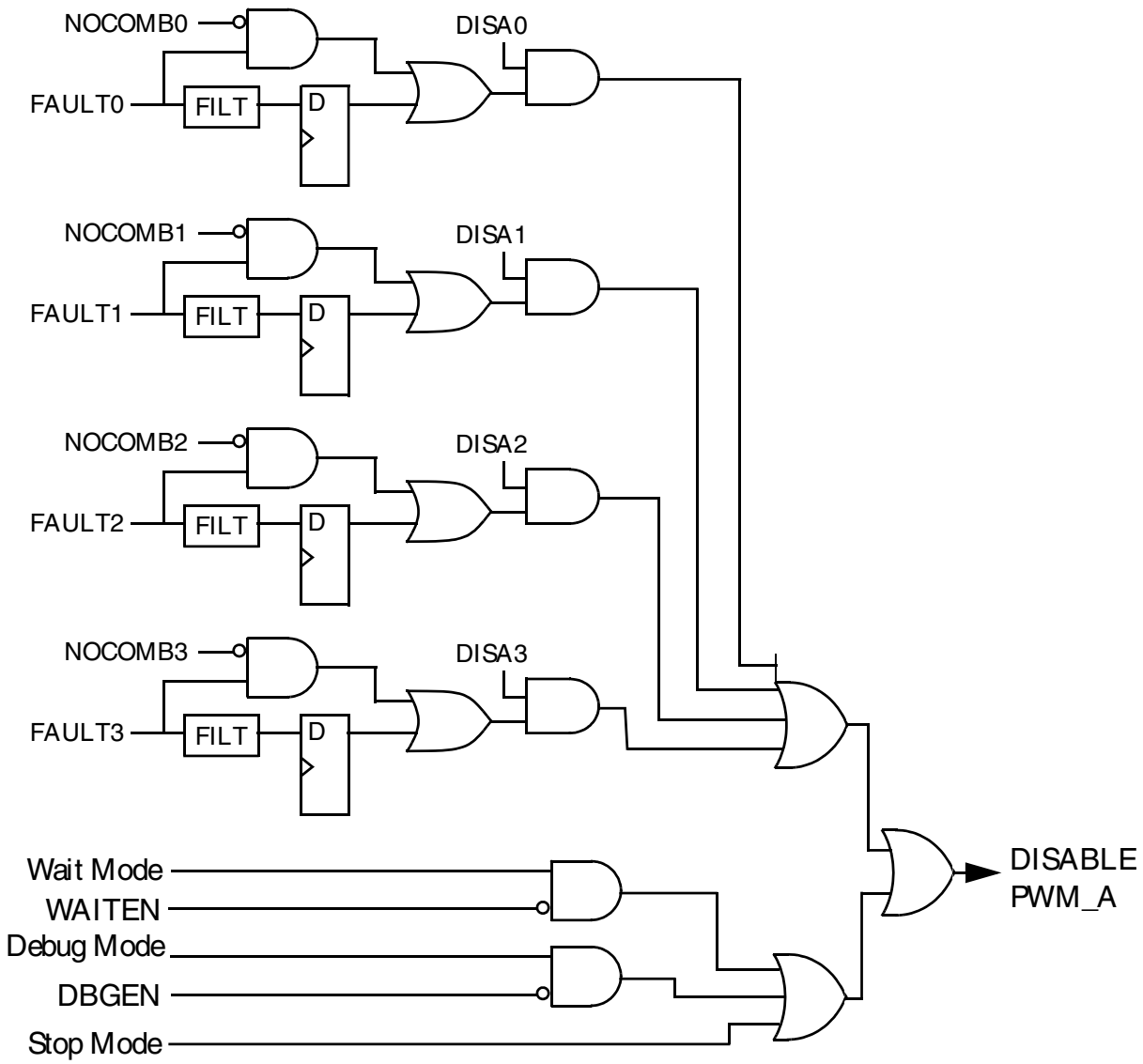


Figure 55-30. Fault Decoder for PWM_A

Table 55-3. Fault Mapping

PWM Pin	Controlling Register Bits
PWM_A	DISMAP0[DIS0A]
PWM_B	DISMAP0[DIS0B]
PWM_X	DISMAP0[DIS0X]

55.3.2.12.1 Fault Pin Filter

Each fault pin has a programmable filter that can be bypassed. The sampling period of the filter can be adjusted with FFILT[FILT_PER]. The number of consecutive samples that must agree before an input transition is recognized can be adjusted using FFILT[FILT_CNT]. Setting FFILT[FILT_PER] to all 0 disables the input filter for a given FAULTx pin.

Upon detecting a logic 0 on the filtered FAULTx pin (or a logic 1 if FCTRL[FLVLx] is set), the corresponding FSTS[FFPINx] and fault flag, FSTS[FFLAGx], bits are set. FSTS[FFPINx] remains set as long as the filtered FAULTx pin is zero. Clear FSTS[FFLAGx] by writing a logic 1 to FSTS[FFLAGx].

If the FIEx, FAULTx pin interrupt enable bit is set, FSTS[FFLAGx] generates a CPU interrupt request. The interrupt request latch remains set until:

- Software clears FSTS[FFLAGx] by writing a logic one to the bit
- Software clears the FIEx bit by writing a logic zero to it
- A reset occurs

Even with the filter enabled, there is a combinational path from the FAULTx inputs to the PWM pins. This logic is also capable of holding a fault condition in the event of loss of clock to the PWM module.

55.3.2.12.2 Automatic Fault Clearing

Setting an automatic clearing mode bit, FCTRL[FAUTOx], configures faults from the FAULTx pin for automatic clearing.

When FCTRL[FAUTOx] is set, disabled PWM pins are enabled when the FAULTx pin returns to logic one and a new PWM full or half cycle begins. See the following figure. If FSTS[FFULLx] is set, then the disabled PWM pins are enabled at the start of a full cycle. If FSTS[FHALFx] is set, then the disabled PWM pins are enabled at the start of a half cycle. Clearing FSTS[FFLAGx] does not affect disabled PWM pins when FCTRL[FAUTOx] is set.

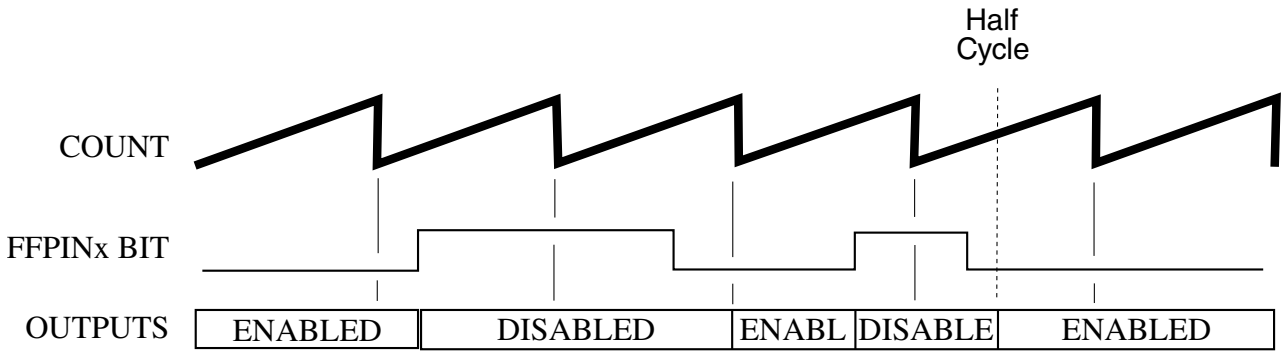


Figure 55-31. Automatic Fault Clearing

55.3.2.12.3 Manual Fault Clearing

Clearing the automatic clearing mode bit, FCTRL[FAUTOx], configures faults from the FAULTx pin for manual clearing:

- If the fault safety mode bits, FCTRL[FSAFEx], are clear, then PWM pins disabled by the FAULTx pins are enabled when:
  - Software clears the corresponding FSTS[FFLAGx] flag
  - The pins are enabled when the next PWM full or half cycle begins regardless of the logic level detected by the filter at the FAULTx pin. See the first following figure. If FSTS[FFULLx] is set, then the disabled PWM pins are enabled at the start of a full cycle. If FSTS[FHALFx] is set, then the disabled PWM pins are enabled at the start of a half cycle.
- If the fault safety mode bits, FCTRL[FSAFEx], are set, then PWM pins disabled by the FAULTx pins are enabled when:
  - Software clears the corresponding FSTS[FFLAGx] flag
  - The filter detects a logic zero on the FAULTx pin at the start of the next PWM full or half cycle boundary. See the second following figure. If FSTS[FFULLx] is set, then the disabled PWM pins are enabled at the start of a full cycle. If FSTS[FHALFx] is set, then the disabled PWM pins are enabled at the start of a half cycle.

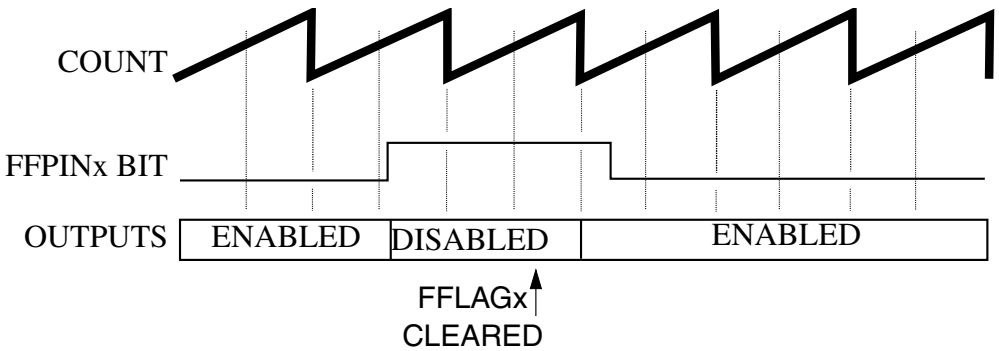


Figure 55-32. Manual Fault Clearing (FCTRL[FSAFE]=0)

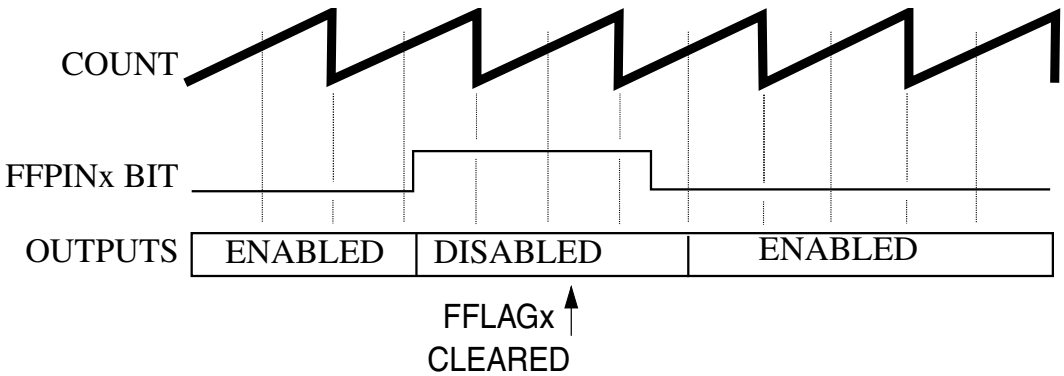


Figure 55-33. Manual Fault Clearing (FCTRL[FSAFE]=1)

Note

Fault protection also applies during software output control when the SEL23 and SEL45 fields are set to select OUT23 and OUT45 bits or PWM_EXTA and PWM_EXTB. Fault clearing still occurs at half PWM cycle boundaries while the PWM generator is engaged, MCTRL[RUN] equals one. But the OUTx bits can control the PWM pins while the PWM generator is off, MCTRL[RUN] equals zero. Thus, fault clearing occurs at IPBus cycles while the PWM generator is off and at the start of PWM cycles when the generator is engaged.

55.3.2.12.4 Fault Testing

FTST[FTEST] is used to simulate a fault condition on each of the fault inputs within that fault channel.

55.3.3 PWM Generator Loading

55.3.3.1 Load Enable

MCTRL[LDOK] enables loading of the following PWM generator parameters:

- The prescaler divisor—from CTRL[PRSC]
- The PWM period and pulse width—from the INIT and VALx registers

MCTRL[LDOK] allows software to finish calculating all of these PWM parameters so they can be synchronously updated. The CTRL[PRSC], INIT, and VALx registers are loaded by software into a set of outer buffers. When MCTRL[LDOK] is set, these values are transferred to an inner set of registers at the beginning of the next PWM reload cycle to be used by the PWM generator. These values can be transferred to the inner set of registers immediately upon setting MCTRL[LDOK] if CTRL[LDMOD] is set. Set MCTRL[LDOK] by reading it when it is a logic zero and then writing a logic one to it. After loading, MCTRL[LDOK] is automatically cleared.

55.3.3.2 Load Frequency

CTRL[LDFQ] selects an integral loading frequency of one to 16 PWM reload opportunities. CTRL[LDFQ] takes effect at every PWM reload opportunity, regardless the state of MCTRL[LDOK]. CTRL[HALF] and CTRL[FULL] control reload timing. If CTRL[FULL] is set, a reload opportunity occurs at the end of every PWM cycle when the count equals VAL1. If CTRL[HALF] is set, a reload opportunity occurs at the half cycle when the count equals VAL0. If both CTRL[HALF] and CTRL[FULL] are set, a reload opportunity occurs twice per PWM cycle when the count equals VAL1 and when it equals VAL0.

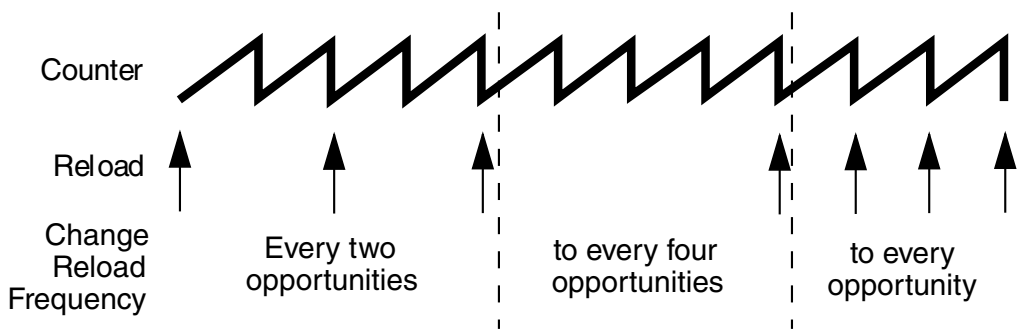


Figure 55-34. Full Cycle Reload Frequency Change

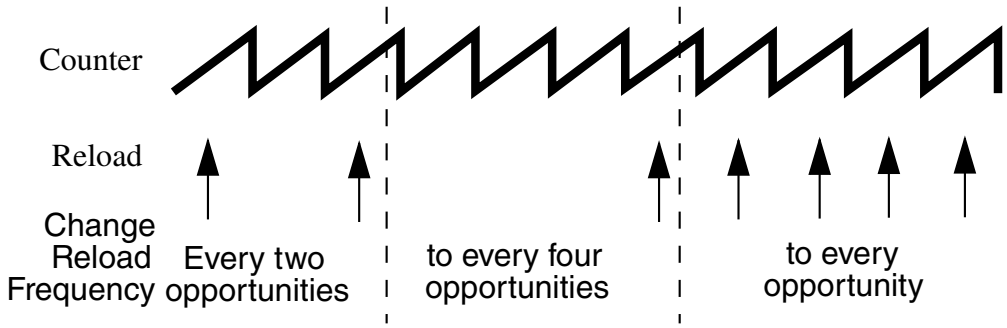


Figure 55-35. Half Cycle Reload Frequency Change

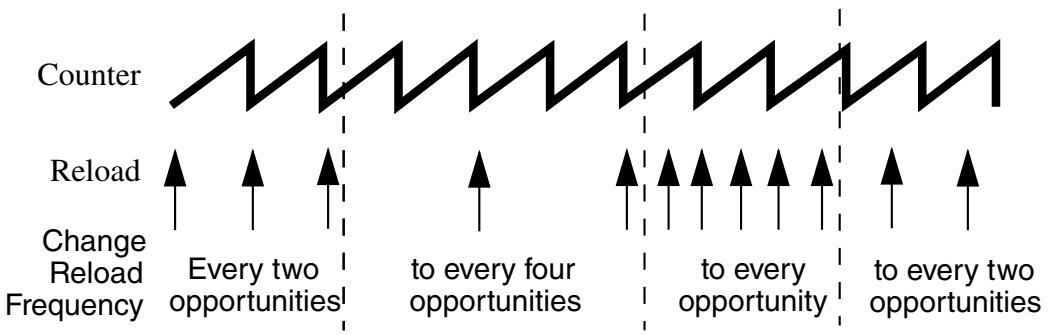


Figure 55-36. Full and Half Cycle Reload Frequency Change

55.3.3.3 Reload Flag

At every reload opportunity the PWM Reload Flag (STS[RF]) is set. Setting STS[RF] happens even if an actual reload is prevented by MCTRL[LDOK]. If the PWM reload interrupt enable bit, INTEN[RIE], is set, the STS[RF] flag generates CPU interrupt requests allowing software to calculate new PWM parameters in real time. When INTEN[RIE] is not set, reloads still occur at the selected reload rate without generating CPU interrupt requests.

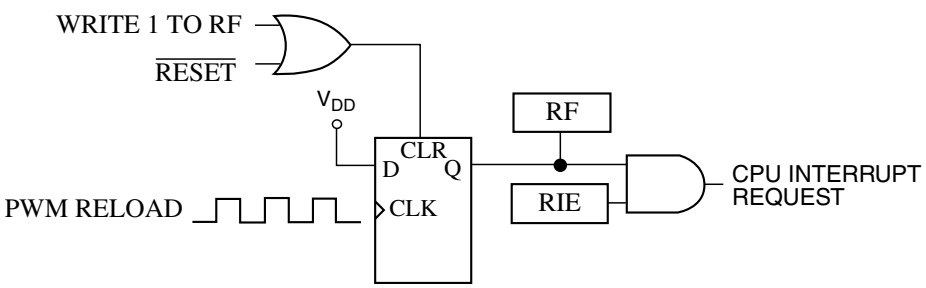


Figure 55-37. PWMF Reload Interrupt Request

55.3.3.4 Reload Errors

Whenever one of the VALx, FRACVALx, or CTRL[PRSC] registers is updated, the STS[RUF] flag is set to indicate that the data is not coherent. STS[RUF] will be cleared by a successful reload which consists of the reload signal while MCTRL[LDOK] is set. If STS[RUF] is set and MCTRL[LDOK] is clear when the reload signal occurs, a reload error has taken place and STS[REF] is set. If STS[RUF] is clear when a reload signal asserts, then the data is coherent and no error will be flagged.

55.3.3.5 Initialization

Initialize all registers and set MCTRL[LDOK] before setting MCTRL[RUN].

Note

Even if MCTRL[LDOK] is not set, setting MCTRL[RUN] also sets the STS[RF] flag. To prevent a CPU interrupt request, clear INTEN[RIE] before setting MCTRL[RUN].

The PWM generator uses the last values loaded if MCTRL[RUN] is cleared and then set while MCTRL[LDOK] equals zero.

When MCTRL[RUN] is cleared:

- The STS[RF] flag and pending CPU interrupt requests are not cleared
- All fault circuitry remains active
- Software/external output control remains active
- Deadtime insertion continues during software/external output control

55.4 External Signals

The PWM has pins named PWM[n]_A, PWM[n]_B, PWM[n]_X, FAULT[n], PWM[n]_EXT_SYNC, EXT_FORCE, PWM[n]_EXTA, and PWM[n]_EXTB. The PWM also has an on-chip input called EXT_CLK and an output signal called PWM[n]_OUT_TRIGx.

55.4.1 PWM[n]_A and PWM[n]_B - External PWM Output Pair

These pins are the output pins of the PWM channels. These pins can be independent PWM signals or a complementary pair. When not needed as an output, they can be used as inputs to the input capture circuitry.

55.4.2 PWM[n]_X - Auxiliary PWM Output signal

These pins are the auxiliary output pins of the PWM channels. They can be independent PWM signals. When not needed as an output, they can be used as inputs to the input capture circuitry or used to detect the polarity of the current flowing through the complementary circuit at deadtime correction.

X pins have limited usability as input capture - see note on pin 3103

55.4.3 FAULT[n] - Fault Inputs

These are input pins for disabling selected PWM outputs.

55.4.4 PWM[n]_EXT_SYNC - External Synchronization Signal

These input signals allow a source external to the PWM to initialize the PWM counter. In this manner, the PWM can be synchronized to external circuitry.

55.4.5 EXT_FORCE - External Output Force Signal

This input signal allows a source external to the PWM to force an update of the PWM outputs. In this manner, the PWM can be synchronized to external circuitry.

55.4.6 PWM[n]_EXTA and PWM[n]_EXTB - Alternate PWM Control Signals

These pins allow an alternate source to control the PWM_A and PWM_B outputs. Typically, either the PWM_EXTA or PWM_EXTB input (depending on the state of MCTRL[IPOL]) is used for the generation of a complementary pair. Typical control signals include ADC conversion high/low limits, TMR outputs, GPIO inputs, and comparator outputs.

55.4.7 PWM[n]_OUT_TRIG0 and PWM[n]_OUT_TRIG1 - Output Triggers

These outputs allow the PWM submodules to control timing of ADC conversions. See the description of the OUT_TRIG_EN bits in the Output Trigger Control Register for information about how to enable these outputs and how the compare registers match up to the output triggers.

55.4.8 EXT_CLK - External Clock Signal

This signal allows a source external to the PWM (typically a timer or an off-chip source) to control the PWM clocking. In this manner, the PWM can be synchronized to the timer, or multiple chips can be synchronized to each other.

55.5 Resets

All PWM registers are reset to their default values upon any system reset.

The reset forces all registers to their reset states and tri-states the PWM outputs.

55.6 Interrupts

Each of the submodules within the eFlexPWM module can generate an interrupt from several sources. The fault logic can also generate interrupts. The interrupt service routine (ISR) must check the related interrupt enables and interrupt flags to determine the actual cause of the interrupt.

Table 55-4. Interrupt Summary

Core Interrupt	Interrupt Flag	Interrupt Enable	Name	Description
PWM_CMP0	SM0STS[CMPF]	SM0INTEN[CMPIE]	Submodule 0 compare interrupt	Compare event has occurred
PWM_CAP0	SM0STS[CFA1], SM0STS[CFA0], SM0STS[CFB1], SM0STS[CFB0], SM0STS[CFX1],	SM0INTEN[CFA1IE], SM0INTEN[CFA0IE], SM0INTEN[CFB1IE], SM0INTEN[CFB0IE], SM0INTEN[CFX1IE],	Submodule 0 input capture interrupt	Input capture event has occurred

Table continues on the next page...

Table 55-4. Interrupt Summary (continued)

Core Interrupt	Interrupt Flag	Interrupt Enable	Name	Description
	SM0STS[CFX0]	SM0INTEN[CFX0IE]		
PWM_RELOAD0	SM0STS[RF]	SM0INTEN[RIE]	Submodule 0 reload interrupt	Reload event has occurred
PWM_CMP1	SM1STS[CMPF]	SM1INTEN[CMPIE]	Submodule 1 compare interrupt	Compare event has occurred
PWM_CAP1	SM1STS[CFA1], SM1STS[CFA0], SM1STS[CFB1], SM1STS[CFB0], SM1STS[CFX1], SM1STS[CFX0]	SM1INTEN[CFA1IE], SM1INTEN[CFA0IE], SM1INTEN[CFB1IE], SM1INTEN[CFB0IE], SM1INTEN[CFX1IE], SM1INTEN[CFX0IE]	Submodule 1 input capture interrupt	Input capture event has occurred
PWM_RELOAD1	SM1STS[RF]	SM1INTEN[RIE]	Submodule 1 reload interrupt	Reload event has occurred
PWM_CMP2	SM2STS[CMPF]	SM2INTEN[CMPIE]	Submodule 2 compare interrupt	Compare event has occurred
PWM_CAP2	SM2STS[CFA1], SM2STS[CFA0], SM2STS[CFB1], SM2STS[CFB0], SM2STS[CFX1], SM2STS[CFX0]	SM2INTEN[CFA1IE], SM2INTEN[CFA0IE], SM2INTEN[CFB1IE], SM2INTEN[CFB0IE], SM2INTEN[CFX1IE], SM2INTEN[CFX0IE	Submodule 2 input capture interrupt	Input capture event has occurred
PWM_RELOAD2	SM2STS[RF]	SM2INTEN[RIE]	Submodule 2 reload interrupt	Reload event has occurred
PWM_CMP3	SM3STS[CMPF]	SM3INTEN[CMPIE]	Submodule 3 compare interrupt	Compare event has occurred
PWM_CAP3	SM3STS[CFA1], SM3STS[CFA0], SM3STS[CFB1], SM3STS[CFB0], SM3STS[CFX1], SM3STS[CFX0]	SM3INTEN[CFA1IE], SM3INTEN[CFA0IE], SM3INTEN[CFB1IE], SM3INTEN[CFB0IE], SM3INTEN[CFX1IE], SM3INTEN[CFX0IE	Submodule 3 input capture interrupt	Input capture event has occurred
PWM_RELOAD3	SM3STS[RF]	SM3INTEN[RIE]	Submodule 3 reload interrupt	Reload event has occurred
PWM_RERR	SM0STS[REF]	SM0INTEN[REIE]	Submodule 0 reload error interrupt	Reload error has occurred
	SM1STS[REF]	SM1INTEN[REIE]	Submodule 1 reload error interrupt	
	SM2STS[REF]	SM2INTEN[REIE]	Submodule 2 reload error interrupt	
	SM3STS[REF]	SM3INTEN[REIE]	Submodule 3 reload error interrupt	

Table continues on the next page...

Table 55-4. Interrupt Summary (continued)

Core Interrupt	Interrupt Flag	Interrupt Enable	Name	Description
PWM_FAULT	FSTS[FFLAG]	FCTRL[FIE]	Fault input interrupt	Fault condition has been detected

55.7 DMA

Each submodule can request a DMA read access for its capture FIFOs and a DMA write request for its double buffered VALx registers.

Table 55-5. DMA Summary

DMA Request	DMA Enable	Name	Description
Submodule 0 read request	SM0DMAEN[CX0DE]	SM0 Capture FIFO X0 read request	SM0CVAL0 contains a value to be read
	SM0DMAEN[CX1DE]	SM0 Capture FIFO X1 read request	SM0CVAL1 contains a value to be read
	SM0DMAEN[CA0DE]	SM0 Capture FIFO A0 read request	SM0CVAL2 contains a value to be read
	SM0DMAEN[CA1DE]	SM0 Capture FIFO A1 read request	SM0CVAL3 contains a value to be read
	SM0DMAEN[CB0DE]	SM0 Capture FIFO B0 read request	SM0CVAL4 contains a value to be read
	SM0DMAEN[CB1DE]	SM0 Capture FIFO B1 read request	SM0CVAL5 contains a value to be read
	SM0DMAEN[CAPTDE]	SM0 Capture FIFO read request source select	Selects source of submodule0 read DMA request
Submodule 0 write request	SM0DMAEN[VALDE]	SM0VALx write request	SM0VALx registers need to be updated
Submodule 1 read request	SM1DMAEN[CX0DE]	SM1 Capture FIFO X0 read request	SM1CVAL0 contains a value to be read
	SM1DMAEN[CX1DE]	SM1 Capture FIFO X! read request	SM1CVAL1 contains a value to be read
	SM1DMAEN[CA0DE]	SM1Capture FIFO A0 read request	SM1CVAL2 contains a value to be read
	SM1DMAEN[CA1DE]	SM1 Capture FIFO A1 read request	SM1CVAL3 contains a value to be read
	SM1DMAEN[CB0DE]	SM1 Capture FIFO B0 read request	SM1CVAL4 contains a value to be read
	SM1DMAEN[CB1DE]	SM1 Capture FIFO B1 read request	SM!CVAL5 contains a value to be read
	SM1DMAEN[CAPTDE]	SM1 Capture FIFO read request source select	Selects source of submodule1 read DMA request

Table continues on the next page...

Table 55-5. DMA Summary (continued)

DMA Request	DMA Enable	Name	Description
Submodule 1 write request	SM1DMAEN[VALDE]	SM1VALx write request	SM1VALx registers need to be updated
Submodule 2 read request	SM2DMAEN[CX0DE]	SM2 Capture FIFO X0 read request	SM2CVAL0 contains a value to be read
	SM2DMAEN[CX1DE]	SM2 Capture FIFO X! read request	SM2CVAL1 contains a value to be read
	SM2DMAEN[CA0DE]	SM2 Capture FIFO A0 read request	SM2CVAL2 contains a value to be read
	SM2DMAEN[CA1DE]	SM2 Capture FIFO A1 read request	SM2CVAL3 contains a value to be read
	SM2DMAEN[CB0DE]	SM2 Capture FIFO B0 read request	SM2CVAL4 contains a value to be read
	SM2DMAEN[CB1DE]	SM2 Capture FIFO B1 read request	SM2CVAL5 contains a value to be read
	SM2DMAEN[CAPTDE]	SM2 Capture FIFO read request source select	Selects source of submodule2 read DMA request
Submodule 2 write request	SM2DMAEN[VALDE]	SM2VALx write request	SM2VALx registers need to be updated
Submodule 3 read request	SM3DMAEN[CX0DE]	SM3 Capture FIFO X0 read request	SM3CVAL0 contains a value to be read
	SM3DMAEN[CX1DE]	SM3 Capture FIFO X! read request	SM3CVAL1 contains a value to be read
	SM3DMAEN[CA0DE]	SM3 Capture FIFO A0 read request	SM3CVAL2 contains a value to be read
	SM3DMAEN[CA1DE]	SM3 Capture FIFO A1 read request	SM3CVAL3 contains a value to be read
	SM3DMAEN[CB0DE]	SM3 Capture FIFO B0 read request	SM3CVAL4 contains a value to be read
	SM3DMAEN[CB1DE]	SM3 Capture FIFO B1 read request	SM3CVAL5 contains a value to be read
	SM3DMAEN[CAPTDE]	SM3 Capture FIFO read request source select	Selects source of submodule3 read DMA request
Submodule 3 write request	SM3DMAEN[VALDE]	SM3VALx write request	SM3VALx registers need to be updated

55.8 PWM register descriptions

The address of a register is the sum of a base address and an address offset. The base address is defined at the core level, and the address offset is defined at the module level. The PWM module has a set of registers for each PWM submodule, for the configuration logic, and for each fault channel.

NOTE

While the registers are 16-bit wide, they can be accessed in pairs as 32-bit registers, if the pairs are word-aligned.

Submodule registers are repeated for each PWM submodule. To designate which submodule they are in, register names are prefixed with SMx(x=0,1,2,3). The base address of submodule 0 is the same as the base address for the PWM module as a whole. The base address of submodule 1 is offset 0x60 from the base address for the PWM module as a whole. This 0x60 offset is based on the number of registers in a submodule.The base address of submodule 2 is equal to the base address of submodule 1 plus this same 0x60 offset.The pattern repeats for the base address of submodule 3.

The base address of the configuration registers is equal to the base address of the PWM module as a whole plus an offset of 0x180.

Fault channel registers are repeated for each fault channel. To designate which fault channel they are in, register names are prefixed with F0 and F1. The base address of fault channel 0 is equal to the base address of the PWM module as a whole plus an offset of 0x18C. The base address of fault channel 1 is the base address of fault channel 0 + 0x4. This 0x4 offset is based on the number of registers in a fault channel. Each of the four fields in the fault channel registers corresponds to fault inputs 3-0.

55.8.1 PWM memory map

PWM1 base address: 403D_C000h

PWM2 base address: 403E_0000h

PWM3 base address: 403E_4000h

PWM4 base address: 403E_8000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Counter Register (SM0CNT)	16	RO	0000h
2h	Initial Count Register (SM0INIT)	16	RW	0000h
4h	Control 2 Register (SM0CTRL2)	16	RW	0000h
6h	Control Register (SM0CTRL)	16	RW	0400h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
Ah	Value Register 0 (SM0VAL0)	16	RW	0000h
Ch	Fractional Value Register 1 (SM0FRACVAL1)	16	RW	0000h
Eh	Value Register 1 (SM0VAL1)	16	RW	0000h
10h	Fractional Value Register 2 (SM0FRACVAL2)	16	RW	0000h
12h	Value Register 2 (SM0VAL2)	16	RW	0000h
14h	Fractional Value Register 3 (SM0FRACVAL3)	16	RW	0000h
16h	Value Register 3 (SM0VAL3)	16	RW	0000h
18h	Fractional Value Register 4 (SM0FRACVAL4)	16	RW	0000h
1Ah	Value Register 4 (SM0VAL4)	16	RW	0000h
1Ch	Fractional Value Register 5 (SM0FRACVAL5)	16	RW	0000h
1Eh	Value Register 5 (SM0VAL5)	16	RW	0000h
20h	Fractional Control Register (SM0FRCTRL)	16	RW	0000h
22h	Output Control Register (SM0OCTRL)	16	RW	0000h
24h	Status Register (SM0STS)	16	W1C	0000h
26h	Interrupt Enable Register (SM0INTEN)	16	RW	0000h
28h	DMA Enable Register (SM0DMAEN)	16	RW	0000h
2Ah	Output Trigger Control Register (SM0TCTRL)	16	RW	0000h
2Ch	Fault Disable Mapping Register 0 (SM0DISMAP0)	16	RW	FFFFh
30h	Deadtime Count Register 0 (SM0DTCNT0)	16	RW	07FFh
32h	Deadtime Count Register 1 (SM0DTCNT1)	16	RW	07FFh
34h	Capture Control A Register (SM0CAPTCTRLA)	16	RW	0000h
36h	Capture Compare A Register (SM0CAPTCOMPA)	16	RW	0000h
38h	Capture Control B Register (SM0CAPTCTRLB)	16	RW	0000h
3Ah	Capture Compare B Register (SM0CAPTCOMPB)	16	RW	0000h
3Ch	Capture Control X Register (SM0CAPTCTRLX)	16	RW	0000h
3Eh	Capture Compare X Register (SM0CAPTCOMPX)	16	RW	0000h
40h	Capture Value 0 Register (SM0CVAL0)	16	RO	0000h
42h	Capture Value 0 Cycle Register (SM0CVAL0CYC)	16	RO	0000h
44h	Capture Value 1 Register (SM0CVAL1)	16	RO	0000h
46h	Capture Value 1 Cycle Register (SM0CVAL1CYC)	16	RO	0000h
48h	Capture Value 2 Register (SM0CVAL2)	16	RO	0000h
4Ah	Capture Value 2 Cycle Register (SM0CVAL2CYC)	16	RO	0000h
4Ch	Capture Value 3 Register (SM0CVAL3)	16	RO	0000h
4Eh	Capture Value 3 Cycle Register (SM0CVAL3CYC)	16	RO	0000h
50h	Capture Value 4 Register (SM0CVAL4)	16	RO	0000h
52h	Capture Value 4 Cycle Register (SM0CVAL4CYC)	16	RO	0000h
54h	Capture Value 5 Register (SM0CVAL5)	16	RO	0000h
56h	Capture Value 5 Cycle Register (SM0CVAL5CYC)	16	RO	0000h
60h	Counter Register (SM1CNT)	16	RO	0000h
62h	Initial Count Register (SM1INIT)	16	RW	0000h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
64h	Control 2 Register (SM1CTRL2)	16	RW	0000h
66h	Control Register (SM1CTRL)	16	RW	0400h
6Ah	Value Register 0 (SM1VAL0)	16	RW	0000h
6Ch	Fractional Value Register 1 (SM1FRACVAL1)	16	RW	0000h
6Eh	Value Register 1 (SM1VAL1)	16	RW	0000h
70h	Fractional Value Register 2 (SM1FRACVAL2)	16	RW	0000h
72h	Value Register 2 (SM1VAL2)	16	RW	0000h
74h	Fractional Value Register 3 (SM1FRACVAL3)	16	RW	0000h
76h	Value Register 3 (SM1VAL3)	16	RW	0000h
78h	Fractional Value Register 4 (SM1FRACVAL4)	16	RW	0000h
7Ah	Value Register 4 (SM1VAL4)	16	RW	0000h
7Ch	Fractional Value Register 5 (SM1FRACVAL5)	16	RW	0000h
7Eh	Value Register 5 (SM1VAL5)	16	RW	0000h
80h	Fractional Control Register (SM1FRCTRL)	16	RW	0000h
82h	Output Control Register (SM1OCTRL)	16	RW	0000h
84h	Status Register (SM1STS)	16	W1C	0000h
86h	Interrupt Enable Register (SM1INTEN)	16	RW	0000h
88h	DMA Enable Register (SM1DMAEN)	16	RW	0000h
8Ah	Output Trigger Control Register (SM1TCTRL)	16	RW	0000h
8Ch	Fault Disable Mapping Register 0 (SM1DISMAP0)	16	RW	FFFFh
90h	Deadtime Count Register 0 (SM1DTCNT0)	16	RW	07FFh
92h	Deadtime Count Register 1 (SM1DTCNT1)	16	RW	07FFh
94h	Capture Control A Register (SM1CAPTCTRLA)	16	RW	0000h
96h	Capture Compare A Register (SM1CAPTCOMPA)	16	RW	0000h
98h	Capture Control B Register (SM1CAPTCTRLB)	16	RW	0000h
9Ah	Capture Compare B Register (SM1CAPTCOMPB)	16	RW	0000h
9Ch	Capture Control X Register (SM1CAPTCTRLX)	16	RW	0000h
9Eh	Capture Compare X Register (SM1CAPTCOMPX)	16	RW	0000h
A0h	Capture Value 0 Register (SM1CVAL0)	16	RO	0000h
A2h	Capture Value 0 Cycle Register (SM1CVAL0CYC)	16	RO	0000h
A4h	Capture Value 1 Register (SM1CVAL1)	16	RO	0000h
A6h	Capture Value 1 Cycle Register (SM1CVAL1CYC)	16	RO	0000h
A8h	Capture Value 2 Register (SM1CVAL2)	16	RO	0000h
AAh	Capture Value 2 Cycle Register (SM1CVAL2CYC)	16	RO	0000h
ACh	Capture Value 3 Register (SM1CVAL3)	16	RO	0000h
AEh	Capture Value 3 Cycle Register (SM1CVAL3CYC)	16	RO	0000h
B0h	Capture Value 4 Register (SM1CVAL4)	16	RO	0000h
B2h	Capture Value 4 Cycle Register (SM1CVAL4CYC)	16	RO	0000h
B4h	Capture Value 5 Register (SM1CVAL5)	16	RO	0000h
B6h	Capture Value 5 Cycle Register (SM1CVAL5CYC)	16	RO	0000h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
C0h	Counter Register (SM2CNT)	16	RO	0000h
C2h	Initial Count Register (SM2INIT)	16	RW	0000h
C4h	Control 2 Register (SM2CTRL2)	16	RW	0000h
C6h	Control Register (SM2CTRL)	16	RW	0400h
CAh	Value Register 0 (SM2VAL0)	16	RW	0000h
CCh	Fractional Value Register 1 (SM2FRACVAL1)	16	RW	0000h
CEh	Value Register 1 (SM2VAL1)	16	RW	0000h
D0h	Fractional Value Register 2 (SM2FRACVAL2)	16	RW	0000h
D2h	Value Register 2 (SM2VAL2)	16	RW	0000h
D4h	Fractional Value Register 3 (SM2FRACVAL3)	16	RW	0000h
D6h	Value Register 3 (SM2VAL3)	16	RW	0000h
D8h	Fractional Value Register 4 (SM2FRACVAL4)	16	RW	0000h
DAh	Value Register 4 (SM2VAL4)	16	RW	0000h
DCh	Fractional Value Register 5 (SM2FRACVAL5)	16	RW	0000h
DEh	Value Register 5 (SM2VAL5)	16	RW	0000h
E0h	Fractional Control Register (SM2FRCTRL)	16	RW	0000h
E2h	Output Control Register (SM2OCTRL)	16	RW	0000h
E4h	Status Register (SM2STS)	16	W1C	0000h
E6h	Interrupt Enable Register (SM2INTEN)	16	RW	0000h
E8h	DMA Enable Register (SM2DMAEN)	16	RW	0000h
EAh	Output Trigger Control Register (SM2TCTRL)	16	RW	0000h
ECh	Fault Disable Mapping Register 0 (SM2DISMAP0)	16	RW	FFFFh
F0h	Deadtime Count Register 0 (SM2DTCNT0)	16	RW	07FFh
F2h	Deadtime Count Register 1 (SM2DTCNT1)	16	RW	07FFh
F4h	Capture Control A Register (SM2CAPTCTRLA)	16	RW	0000h
F6h	Capture Compare A Register (SM2CAPTCOMPA)	16	RW	0000h
F8h	Capture Control B Register (SM2CAPTCTRLB)	16	RW	0000h
FAh	Capture Compare B Register (SM2CAPTCOMPB)	16	RW	0000h
FCh	Capture Control X Register (SM2CAPTCTRLX)	16	RW	0000h
FEh	Capture Compare X Register (SM2CAPTCOMPX)	16	RW	0000h
100h	Capture Value 0 Register (SM2CVAL0)	16	RO	0000h
102h	Capture Value 0 Cycle Register (SM2CVAL0CYC)	16	RO	0000h
104h	Capture Value 1 Register (SM2CVAL1)	16	RO	0000h
106h	Capture Value 1 Cycle Register (SM2CVAL1CYC)	16	RO	0000h
108h	Capture Value 2 Register (SM2CVAL2)	16	RO	0000h
10Ah	Capture Value 2 Cycle Register (SM2CVAL2CYC)	16	RO	0000h
10Ch	Capture Value 3 Register (SM2CVAL3)	16	RO	0000h
10Eh	Capture Value 3 Cycle Register (SM2CVAL3CYC)	16	RO	0000h
110h	Capture Value 4 Register (SM2CVAL4)	16	RO	0000h
112h	Capture Value 4 Cycle Register (SM2CVAL4CYC)	16	RO	0000h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
114h	Capture Value 5 Register (SM2CVAL5)	16	RO	0000h
116h	Capture Value 5 Cycle Register (SM2CVAL5CYC)	16	RO	0000h
120h	Counter Register (SM3CNT)	16	RO	0000h
122h	Initial Count Register (SM3INIT)	16	RW	0000h
124h	Control 2 Register (SM3CTRL2)	16	RW	0000h
126h	Control Register (SM3CTRL)	16	RW	0400h
12Ah	Value Register 0 (SM3VAL0)	16	RW	0000h
12Ch	Fractional Value Register 1 (SM3FRACVAL1)	16	RW	0000h
12Eh	Value Register 1 (SM3VAL1)	16	RW	0000h
130h	Fractional Value Register 2 (SM3FRACVAL2)	16	RW	0000h
132h	Value Register 2 (SM3VAL2)	16	RW	0000h
134h	Fractional Value Register 3 (SM3FRACVAL3)	16	RW	0000h
136h	Value Register 3 (SM3VAL3)	16	RW	0000h
138h	Fractional Value Register 4 (SM3FRACVAL4)	16	RW	0000h
13Ah	Value Register 4 (SM3VAL4)	16	RW	0000h
13Ch	Fractional Value Register 5 (SM3FRACVAL5)	16	RW	0000h
13Eh	Value Register 5 (SM3VAL5)	16	RW	0000h
140h	Fractional Control Register (SM3FRCTRL)	16	RW	0000h
142h	Output Control Register (SM3OCTRL)	16	RW	0000h
144h	Status Register (SM3STS)	16	W1C	0000h
146h	Interrupt Enable Register (SM3INTEN)	16	RW	0000h
148h	DMA Enable Register (SM3DMAEN)	16	RW	0000h
14Ah	Output Trigger Control Register (SM3TCTRL)	16	RW	0000h
14Ch	Fault Disable Mapping Register 0 (SM3DISMAP0)	16	RW	FFFFh
150h	Deadtime Count Register 0 (SM3DTCNT0)	16	RW	07FFh
152h	Deadtime Count Register 1 (SM3DTCNT1)	16	RW	07FFh
154h	Capture Control A Register (SM3CAPTCTRLA)	16	RW	0000h
156h	Capture Compare A Register (SM3CAPTCOMPA)	16	RW	0000h
158h	Capture Control B Register (SM3CAPTCTRLB)	16	RW	0000h
15Ah	Capture Compare B Register (SM3CAPTCOMPB)	16	RW	0000h
15Ch	Capture Control X Register (SM3CAPTCTRLX)	16	RW	0000h
15Eh	Capture Compare X Register (SM3CAPTCOMPX)	16	RW	0000h
160h	Capture Value 0 Register (SM3CVAL0)	16	RO	0000h
162h	Capture Value 0 Cycle Register (SM3CVAL0CYC)	16	RO	0000h
164h	Capture Value 1 Register (SM3CVAL1)	16	RO	0000h
166h	Capture Value 1 Cycle Register (SM3CVAL1CYC)	16	RO	0000h
168h	Capture Value 2 Register (SM3CVAL2)	16	RO	0000h
16Ah	Capture Value 2 Cycle Register (SM3CVAL2CYC)	16	RO	0000h
16Ch	Capture Value 3 Register (SM3CVAL3)	16	RO	0000h
16Eh	Capture Value 3 Cycle Register (SM3CVAL3CYC)	16	RO	0000h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
170h	Capture Value 4 Register (SM3CVAL4)	16	RO	0000h
172h	Capture Value 4 Cycle Register (SM3CVAL4CYC)	16	RO	0000h
174h	Capture Value 5 Register (SM3CVAL5)	16	RO	0000h
176h	Capture Value 5 Cycle Register (SM3CVAL5CYC)	16	RO	0000h
180h	Output Enable Register (OUTEN)	16	RW	0000h
182h	Mask Register (MASK)	16	RW	0000h
184h	Software Controlled Output Register (SWCOUT)	16	RW	0000h
186h	PWM Source Select Register (DTSRCSEL)	16	RW	0000h
188h	Master Control Register (MCTRL)	16	RW	0000h
18Ah	Master Control 2 Register (MCTRL2)	16	RW	0000h
18Ch	Fault Control Register (FCTRL0)	16	RW	0000h
18Eh	Fault Status Register (FSTS0)	16	RW	0000h
190h	Fault Filter Register (FFILT0)	16	RW	0000h
192h	Fault Test Register (FTST0)	16	RW	0000h
194h	Fault Control 2 Register (FCTRL20)	16	RW	0000h

55.8.2 Counter Register (SM0CNT - SM3CNT)

55.8.2.1 Offset

Register	Offset
SM0CNT	0h
SM1CNT	60h
SM2CNT	C0h
SM3CNT	120h

55.8.2.2 Function

This read-only register displays the state of the signed 16-bit submodule counter. This register is not byte accessible.

55.8.2.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CNT															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

55.8.2.4 Fields

Field	Function
15-0 CNT	Counter Register Bits

55.8.3 Initial Count Register (SM0INIT - SM3INIT)

55.8.3.1 Offset

Register	Offset
SM0INIT	2h
SM1INIT	62h
SM2INIT	C2h
SM3INIT	122h

55.8.3.2 Function

The 16-bit signed value in this buffered, read/write register defines the initial count value for the PWM in PWM clock periods. This is the value loaded into the submodule counter when local sync, master sync, or master reload is asserted (based on the value of CTRL2[INIT_SEL]) or when CTRL2[FORCE] is asserted and force init is enabled. For PWM operation, the buffered contents of this register are loaded into the counter at the start of every PWM cycle. This register is not byte accessible.

NOTE

The INIT register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. This register cannot be written when MCTRL[LDOK] is set. Reading INIT reads the value in a buffer and not necessarily the value the PWM generator is currently using.

55.8.3.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	INIT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

55.8.3.4 Fields

Field	Function
15-0 INIT	Initial Count Register Bits

55.8.4 Control 2 Register (SM0CTRL2 - SM3CTRL2)

55.8.4.1 Offset

Register	Offset
SM0CTRL2	4h
SM1CTRL2	64h
SM2CTRL2	C4h
SM3CTRL2	124h

55.8.4.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	DBGEN	WAITEN	INDEP	PWM23_INIT	PWM45_INIT	PWMX_INIT	INIT_SEL		FRCEN	0	FORCE_SEL			RELOAD_SEL	CLK_SEL	
W										FORCE						
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

55.8.4.3 Fields

Field	Function
15 DBGEN	Debug Enable When set to one, the PWM will continue to run while the chip is in debug mode. If the device enters debug mode and this bit is zero, then the PWM outputs will be disabled until debug mode is exited. At that point the PWM pins will resume operation as programmed in the PWM registers.
14 WAITEN	WAIT Enable When set to one, the PWM will continue to run while the chip is in WAIT mode. In this mode, the peripheral clock continues to run but the CPU clock does not. If the device enters WAIT mode and this bit is zero, then the PWM outputs will be disabled until WAIT mode is exited. At that point the PWM pins will resume operation as programmed in the PWM registers.
13 INDEP	Independent or Complementary Pair Operation This bit determines if the PWM_A and PWM_B channels will be independent PWMs or a complementary PWM pair. 0b - PWM_A and PWM_B form a complementary PWM pair. 1b - PWM_A and PWM_B outputs are independent PWMs.
12 PWM23_INIT	PWM23 Initial Value This read/write bit determines the initial value for PWM23 and the value to which it is forced when FORCE_INIT is asserted.
11 PWM45_INIT	PWM45 Initial Value This read/write bit determines the initial value for PWM45 and the value to which it is forced when FORCE_INIT is asserted.
10 PWMX_INIT	PWM_X Initial Value This read/write bit determines the initial value for PWM_X and the value to which it is forced when FORCE_INIT is asserted.
9-8 INIT_SEL	Initialization Control Select These read/write bits control the source of the INIT signal which goes to the counter. 00b - Local sync (PWM_X) causes initialization. 01b - Master reload from submodule 0 causes initialization. This setting should not be used in submodule 0 as it will force the INIT signal to logic 0. The submodule counter will only reinitialize when a master reload occurs. 10b - Master sync from submodule 0 causes initialization. This setting should not be used in submodule 0 as it will force the INIT signal to logic 0.

Table continues on the next page...

Field	Function
	11b - EXT_SYNC causes initialization.
7 FRCEN	FRCEN This bit allows the CTRL2[FORCE] signal to initialize the counter without regard to the signal selected by CTRL2[INIT_SEL]. This is a software controlled initialization. A forced initialization will also assert the register reload if MCTRL[LDOK] is set. 0b - Initialization from a FORCE_OUT is disabled. 1b - Initialization from a FORCE_OUT is enabled.
6 FORCE	Force Initialization If CTRL2[FORCE_SEL] is set to 000, writing a 1 to this bit results in a FORCE_OUT event. This causes the following actions to be taken: • The PWM_A and PWM_B output pins will assume values based on DTSRCSEL[SMxSEL23] and DTSRCSEL[SMxSEL45]. • If CTRL2[FRCEN] is set, the counter value will be initialized with the INIT register value.
5-3 FORCE_SEL	This read/write bit determines the source of the FORCE OUTPUT signal for this submodule. 000b - The local force signal, CTRL2[FORCE], from this submodule is used to force updates. 001b - The master force signal from submodule 0 is used to force updates. This setting should not be used in submodule 0 as it will hold the FORCE OUTPUT signal to logic 0. 010b - The local reload signal from this submodule is used to force updates without regard to the state of LDOK. 011b - The master reload signal from submodule0 is used to force updates if LDOK is set. This setting should not be used in submodule0 as it will hold the FORCE OUTPUT signal to logic 0. 100b - The local sync signal from this submodule is used to force updates. 101b - The master sync signal from submodule0 is used to force updates. This setting should not be used in submodule0 as it will hold the FORCE OUTPUT signal to logic 0. 110b - The external force signal, EXT_FORCE, from outside the PWM module causes updates. 111b - The external sync signal, EXT_SYNC, from outside the PWM module causes updates.
2 RELOAD_SEL	Reload Source Select This read/write bit determines the source of the RELOAD signal for this submodule. When this bit is set, MCTRL[LDOK[0]] for submodule 0 should be used since the local MCTRL[LDOK] will be ignored. 0b - The local RELOAD signal is used to reload registers. 1b - The master RELOAD signal (from submodule 0) is used to reload registers. This setting should not be used in submodule 0 as it will force the RELOAD signal to logic 0.
1-0 CLK_SEL	Clock Source Select These read/write bits determine the source of the clock signal for this submodule. 00b - The IPBus clock is used as the clock for the local prescaler and counter. 01b - EXT_CLK is used as the clock for the local prescaler and counter. 10b - Submodule 0's clock (AUX_CLK) is used as the source clock for the local prescaler and counter. This setting should not be used in submodule 0 as it will force the clock to logic 0. 11b - reserved

55.8.5 Control Register (SM0CTRL - SM3CTRL)

55.8.5.1 Offset

Register	Offset
SM0CTRL	6h
SM1CTRL	66h
SM2CTRL	C6h
SM3CTRL	126h

55.8.5.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	LDFQ				HALF	FULL	DT		COMPMODE	PRSC			SPLIT	LDMOD	DBLX	DBLEN
W																
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0

55.8.5.3 Fields

Field	Function
15-12 LDFQ	Load Frequency These buffered read/write bits select the PWM load frequency. Reset clears LDFQ, selecting loading every PWM opportunity. A PWM opportunity is determined by HALF and FULL. **NOTE:** LDFQ takes effect when the current load cycle is complete, regardless of the state of MCTRL[LDOK]. Reading LDFQ reads the buffered values and not necessarily the values currently in effect. 0000b - Every PWM opportunity 0001b - Every 2 PWM opportunities 0010b - Every 3 PWM opportunities 0011b - Every 4 PWM opportunities 0100b - Every 5 PWM opportunities 0101b - Every 6 PWM opportunities 0110b - Every 7 PWM opportunities 0111b - Every 8 PWM opportunities 1000b - Every 9 PWM opportunities 1001b - Every 10 PWM opportunities 1010b - Every 11 PWM opportunities 1011b - Every 12 PWM opportunities 1100b - Every 13 PWM opportunities 1101b - Every 14 PWM opportunities 1110b - Every 15 PWM opportunities 1111b - Every 16 PWM opportunities

Table continues on the next page...

Field	Function
11 HALF	Half Cycle Reload This read/write bit enables half-cycle reloads. A half cycle is defined by when the submodule counter matches the VAL0 register and does not have to be half way through the PWM cycle. 0b - Half-cycle reloads disabled. 1b - Half-cycle reloads enabled.
10 FULL	Full Cycle Reload This read/write bit enables full-cycle reloads. A full cycle is defined by when the submodule counter matches the VAL1 register. Either CTRL[HALF] or CTRL[FULL] must be set in order to move the buffered data into the registers used by the PWM generators or CTRL[LDMOD] must be set. If both CTRL[HALF] and CTRL[FULL] are set, then reloads can occur twice per cycle. 0b - Full-cycle reloads disabled. 1b - Full-cycle reloads enabled.
9-8 DT	Deadtime These read only bits reflect the sampled values of the PWM_X input at the end of each deadtime. Sampling occurs at the end of deadtime 0 for DT[0] and the end of deadtime 1 for DT[1]. Reset clears these bits.
7 COMPMODE	Compare Mode This bit controls how comparisons are made between the VAL* registers and the PWM submodule counter. This bit can only be written one time after which it requires a reset to release the bit for writing again. 0b - The VAL* registers and the PWM counter are compared using an "equal to" method. This means that PWM edges are only produced when the counter is equal to one of the VAL* register values. This implies that a PWMA output that is high at the end of a period will maintain this state until a match with VAL3 clears the output in the following period. 1b - The VAL* registers and the PWM counter are compared using an "equal to or greater than" method. This means that PWM edges are produced when the counter is equal to or greater than one of the VAL* register values. This implies that a PWMA output that is high at the end of a period could go low at the start of the next period if the starting counter value is greater than (but not necessarily equal to) the new VAL3 value.
6-4 PRSC	Prescaler These buffered read/write bits select the divide ratio of the PWM clock frequency selected by CTRL2[CLK_SEL]. **NOTE:** Reading CTRL[PRSC] reads the buffered values and not necessarily the values currently in effect. CTRL[PRSC] takes effect at the beginning of the next PWM cycle and only when the load okay bit, MCTRL[LDOK], is set or CTRL[LDMOD] is set. This field cannot be written when MCTRL[LDOK] is set. 000b - Prescaler 1. PWM clock frequency = f_{clk} 001b - Prescaler 2. PWM clock frequency = f_{clk} /2 010b - Prescaler 4. PWM clock frequency = f_{clk} /4 011b - Prescaler 8. PWM clock frequency = f_{clk} /8 100b - Prescaler 16. PWM clock frequency = f_{clk} /16 101b - Prescaler 32. PWM clock frequency = f_{clk} /32 110b - Prescaler 64. PWM clock frequency = f_{clk} /64 111b - Prescaler 128. PWM clock frequency = f_{clk} /128
3 SPLIT	Split the DBLPWM signal to PWMA and PWMB This read/write bit is only used in independent mode when DBLEN is set. This bit allows the two PWM pulses generated by DBLEN to be split with one pulse on PWMA and one on PWMB. The two pulses within the same PWM period are created by an XOR function of the PWMA and PWMB sources. The

Table continues on the next page...

Field	Function
	splitting function causes PWMA to output the pulse that occurs when the PWMA source is 1 and the PWMB source is 0. The PWMB output occurs when the PWMB source is 1 and the PWMA source is 0. (See Double Switching PWMs .) 0b - DBLPWM is not split. PWMA and PWMB each have double pulses. 1b - DBLPWM is split to PWMA and PWMB.
2 LDMOD	Load Mode Select This read/write bit selects the timing of loading the buffered registers for this submodule. 0b - Buffered registers of this submodule are loaded and take effect at the next PWM reload if MCTRL[LDOK] is set. 1b - Buffered registers of this submodule are loaded and take effect immediately upon MCTRL[LDOK] being set. In this case it is not necessary to set CTRL[FULL] or CTRL[HALF].
1 DBLX	PWMX Double Switching Enable This read/write bit enables the double switching behavior on PWMX. When this bit is set, the PWMX output shall be the exclusive OR combination of PWMA and PWMB prior to polarity and masking considerations. 0b - PWMX double pulse disabled. 1b - PWMX double pulse enabled.
0 DBLEN	Double Switching Enable This read/write bit enables the double switching PWM behavior(See Double Switching PWMs). Double switching is not compatible with fractional PWM clock generation. Make sure this bit is clear when setting FRCTRL[FRAC23_EN], FRCTRL[FRAC45_EN], or FRCTRL[FRAC1_EN]. 0b - Double switching disabled. 1b - Double switching enabled.

55.8.6 Value Register 0 (SM0VAL0 - SM3VAL0)

55.8.6.1 Offset

Register	Offset
SM0VAL0	Ah
SM1VAL0	6Ah
SM2VAL0	CAh
SM3VAL0	12Ah

55.8.6.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	VAL0															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

55.8.6.3 Fields

Field	Function
15-0 VAL0	Value Register 0 The 16-bit signed value in this buffered, read/write register defines the mid-cycle reload point for the PWM in PWM clock periods. This value also defines when the PWM_X signal is set and the local sync signal is reset. This register is not byte accessible. **NOTE:** The VAL0 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. VAL0 cannot be written when MCTRL[LDOK] is set. Reading VAL0 reads the value in a buffer. It is not necessarily the value the PWM generator is currently using.

55.8.7 Fractional Value Register 1 (SM0FRACVAL1 - SM3FRACVAL1)

55.8.7.1 Offset

Register	Offset
SM0FRACVAL1	Ch
SM1FRACVAL1	6Ch
SM2FRACVAL1	CCh
SM3FRACVAL1	12Ch

55.8.7.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FRACVAL1					0										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

55.8.7.3 Fields

Field	Function
15-11 FRACVAL1	Fractional Value 1 Register These bits act as a fractional addition to the value in the VAL1 register which controls the PWM period width. The PWM period is computed in terms of IPBus clock cycles. This fractional portion is accumulated at the end of every cycle until an additional whole IPBus cycle is reached. At this time the value being used for VAL1 is temporarily incremented and the PWM cycle is extended by one clock period to compensate for the accumulated fractional values. **NOTE:** The FRACVAL1 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. FRACVAL1 cannot be written when MCTRL[LDOK] is set. Reading FRACVAL1 reads the value in a buffer and not necessarily the value the PWM generator is currently using.
10-0 —	RESERVED

55.8.8 Value Register 1 (SM0VAL1 - SM3VAL1)

55.8.8.1 Offset

Register	Offset
SM0VAL1	Eh
SM1VAL1	6Eh
SM2VAL1	CEh
SM3VAL1	12Eh

55.8.8.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	VAL1															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

55.8.8.3 Fields

Field	Function
15-0 VAL1	Value Register 1 The 16-bit signed value written to this buffered, read/write register defines the modulo count value (maximum count) for the submodule counter. Upon reaching this count value, the counter reloads itself with the contents of the INIT register and asserts the local sync signal while resetting PWM_X. This register is not byte accessible. **NOTE:** The VAL1 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. VAL1 cannot be written when MCTRL[LDOK] is set. Reading VAL1 reads the value in a buffer. It is not necessarily the value the PWM generator is currently using. **NOTE:** When using FRACVAL1, limit the maximum value of VAL1 to 0xFFFE for unsigned applications or to 0x7FFE for signed applications, to avoid counter rollovers caused by accumulating the fractional period defined by FRACVAL1. **NOTE:** If the VAL1 register defines the timer period (Local Sync is selected as the counter initialization signal), a 100% duty cycle cannot be achieved on the PWMX output. After the count reaches VAL1, the PWMX output is low for a minimum of one count every cycle. When the Master Sync signal (only originated by the Local Sync from sub-module 0) is used to control the timer period, the VAL1 register can be free for other functions such as PWM generation without the duty cycle limitation.

55.8.9 Fractional Value Register 2 (SM0FRACVAL2 - SM3FRACVAL2)

55.8.9.1 Offset

Register	Offset
SM0FRACVAL2	10h
SM1FRACVAL2	70h
SM2FRACVAL2	D0h
SM3FRACVAL2	130h

55.8.9.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FRACVAL2					0										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

55.8.9.3 Fields

Field	Function
15-11 FRACVAL2	Fractional Value 2 These bits act as a fractional addition to the value in the VAL2 register which controls the PWM_A turn on timing. It is also used to control the fractional addition to the turn off delay of PWM_B when MCTRL[IPOLx]=0 in complementary mode, CTRL2[INDEP]=0. **NOTE:** The FRACVAL2 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. FRACVAL2 cannot be written when MCTRL[LDOK] is set. Reading FRACVAL2 reads the value in a buffer and not necessarily the value the PWM generator is currently using. **NOTE:** FRCTRL[FRAC23_EN] should be set to 0 when the values of VAL2 and VAL3 cause the high or low time of the PWM output to be 3 cycles or less.
10-0 —	RESERVED

55.8.10 Value Register 2 (SM0VAL2 - SM3VAL2)

55.8.10.1 Offset

Register	Offset
SM0VAL2	12h
SM1VAL2	72h
SM2VAL2	D2h
SM3VAL2	132h

55.8.10.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	VAL2															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

55.8.10.3 Fields

Field	Function
15-0 VAL2	Value Register 2 The 16-bit signed value in this buffered, read/write register defines the count value to set PWM23 high. This register is not byte accessible. **NOTE:** The VAL2 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. VAL2 cannot be written when MCTRL[LDOK] is set. Reading VAL2 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

55.8.11 Fractional Value Register 3 (SM0FRACVAL3 - SM3FRACVAL3)

55.8.11.1 Offset

Register	Offset
SM0FRACVAL3	14h
SM1FRACVAL3	74h
SM2FRACVAL3	D4h
SM3FRACVAL3	134h

55.8.11.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FRACVAL3					0										
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

55.8.11.3 Fields

Field	Function
15-11 FRACVAL3	Fractional Value 3 These bits act as a fractional addition to the value in the VAL3 register which controls the PWM_A turn off timing. It is also used to control the fractional addition to the turn on delay of PWM_B when MCTRL[IPOLx]=0 in complementary mode, CTRL2[INDEP]=0. **NOTE:** The FRACVAL3 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. FRACVAL3 cannot be written when MCTRL[LDOK] is set. Reading FRACVAL3 reads the value in a buffer and not necessarily the value the PWM generator is currently using. **NOTE:** FRCTRL[FRAC23_EN] should be set to 0 when the values of VAL2 and VAL3 cause the high or low time of the PWM output to be 3 cycles or less.
10-0 —	RESERVED

55.8.12 Value Register 3 (SM0VAL3 - SM3VAL3)

55.8.12.1 Offset

Register	Offset
SM0VAL3	16h
SM1VAL3	76h
SM2VAL3	D6h
SM3VAL3	136h

55.8.12.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	VAL3															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

55.8.12.3 Fields

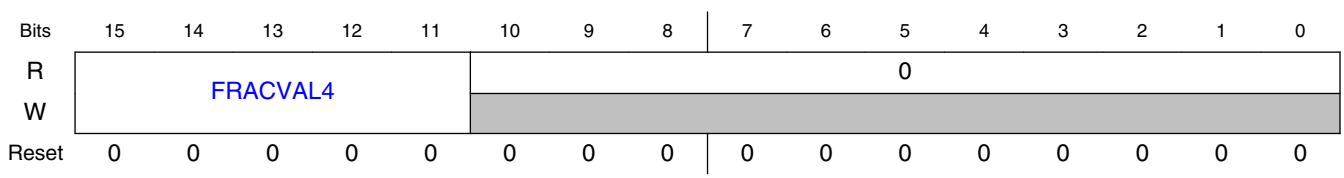
Field	Function
15-0 VAL3	Value Register 3 The 16-bit signed value in this buffered, read/write register defines the count value to set PWM23 low. This register is not byte accessible. **NOTE:** The VAL3 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. VAL3 cannot be written when MCTRL[LDOK] is set. Reading VAL3 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

55.8.13 Fractional Value Register 4 (SM0FRACVAL4 - SM3FRACVAL4)

55.8.13.1 Offset

Register	Offset
SM0FRACVAL4	18h
SM1FRACVAL4	78h
SM2FRACVAL4	D8h
SM3FRACVAL4	138h

55.8.13.2 Diagram



55.8.13.3 Fields

Field	Function
15-11 FRACVAL4	Fractional Value 4

Table continues on the next page...

Field	Function
	These bits act as a fractional addition to the value in the VAL4 register which controls the PWM_B turn on timing. It is also used to control the fractional addition to the turn off delay of PWM_A when MCTRL[IPOLx]=1 in complementary mode, CTRL2[INDEP]=0. **NOTE:** The FRACVAL4 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. FRACVAL4 cannot be written when MCTRL[LDOK] is set. Reading FRACVAL4 reads the value in a buffer and not necessarily the value the PWM generator is currently using. **NOTE:** FRCTRL[FRAC45_EN] should be set to 0 when the values of VAL4 and VAL5 cause the high or low time of the PWM output to be 3 cycles or less.
10-0 —	RESERVED

55.8.14 Value Register 4 (SM0VAL4 - SM3VAL4)

55.8.14.1 Offset

Register	Offset
SM0VAL4	1Ah
SM1VAL4	7Ah
SM2VAL4	DAh
SM3VAL4	13Ah

55.8.14.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	VAL4															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

55.8.14.3 Fields

Field	Function
15-0 VAL4	Value Register 4 The 16-bit signed value in this buffered, read/write register defines the count value to set PWM45 high. This register is not byte accessible.

Field	Function
	NOTE: The VAL4 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. VAL4 cannot be written when MCTRL[LDOK] is set. Reading VAL4 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

55.8.15 Fractional Value Register 5 (SM0FRACVAL5 - SM3FRACVAL5)

55.8.15.1 Offset

Register	Offset
SM0FRACVAL5	1Ch
SM1FRACVAL5	7Ch
SM2FRACVAL5	DCh
SM3FRACVAL5	13Ch

55.8.15.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FRACVAL5								0							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

55.8.15.3 Fields

Field	Function
15-11 FRACVAL5	Fractional Value 5 These bits act as a fractional addition to the value in the VAL5 register which controls the PWM_B turn off timing. It is also used to control the fractional addition to the turn on delay of PWM_A when MCTRL[IPOLx]=1 in complementary mode, CTRL2[INDEP]=0. **NOTE:** The FRACVAL5 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. FRACVAL5 cannot be written when MCTRL[LDOK] is set. Reading FRACVAL5 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

Table continues on the next page...

Field	Function
	NOTE: FRCTRL[FRAC45_EN] should be set to 0 when the values of VAL4 and VAL5 cause the high or low time of the PWM output to be 3 cycles or less.
10-0 —	RESERVED

55.8.16 Value Register 5 (SM0VAL5 - SM3VAL5)

55.8.16.1 Offset

Register	Offset
SM0VAL5	1Eh
SM1VAL5	7Eh
SM2VAL5	DEh
SM3VAL5	13Eh

55.8.16.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	VAL5															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

55.8.16.3 Fields

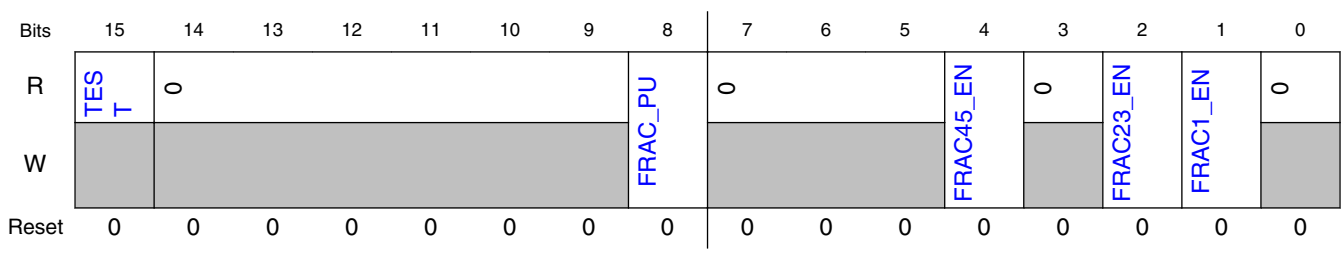
Field	Function
15-0 VAL5	Value Register 5 The 16-bit signed value in this buffered, read/write register defines the count value to set PWM45 low. This register is not byte accessible. **NOTE:** The VAL5 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. VAL5 cannot be written when MCTRL[LDOK] is set. Reading VAL5 reads the value in a buffer and not necessarily the value the PWM generator is currently using.

55.8.17 Fractional Control Register (SM0FRCTRL - SM3FRCTRL)

55.8.17.1 Offset

Register	Offset
SM0FRCTRL	20h
SM1FRCTRL	80h
SM2FRCTRL	E0h
SM3FRCTRL	140h

55.8.17.2 Diagram



55.8.17.3 Fields

Field	Function
15 TEST	Test Status Bit This is a read only test bit for factory use. This bit will reset to 0 but may be either 0 or 1 during PWM operation.
14-9 —	RESERVED
8 FRAC_PU	Fractional Delay Circuit Power Up This bit is used to power up the fractional delay analog block. The fractional delay block takes 25 us to power up after the first FRAC_PU bit in any submodule is set. The fractional delay block only powers down when the FRAC_PU bits in all submodules are 0. The fractional delay logic can only be used when the IPBus clock is running at 100 MHz. When turned off, fractional placement is disabled. After setting this bit and waiting the 25usec, load the PWM VAL* registers with values to create a PWM output with greater than 0% duty cycle and run for at least one PWM period. This can be done without the outputs enabled and is used to clear the state of the analog block that produces the fractional delays. 0b - Turn off fractional delay logic.

Table continues on the next page...

Field	Function
	1b - Power up fractional delay logic.
7-5 —	RESERVED
4 FRAC45_EN	Fractional Cycle Placement Enable for PWM_B This bit is used to enable the fractional cycle edge placement of PWM_B using the FRACVAL4 and FRACVAL5 registers. When disabled, the fractional cycle edge placement of PWM_B is bypassed. **NOTE:** The FRAC45_EN bit is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. FRAC45_EN cannot be written when MCTRL[LDOK] is set. Reading FRAC45_EN reads the value in a buffer and not necessarily the value the PWM generator is currently using. 0b - Disable fractional cycle placement for PWM_B. 1b - Enable fractional cycle placement for PWM_B.
3 —	RESERVED
2 FRAC23_EN	Fractional Cycle Placement Enable for PWM_A This bit is used to enable the fractional cycle edge placement of PWM_A using the FRACVAL2 and FRACVAL3 registers. When disabled, the fractional cycle edge placement of PWM_A is bypassed. **NOTE:** The FRAC23_EN bit is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. FRAC23_EN cannot be written when MCTRL[LDOK] is set. Reading FRAC23_EN reads the value in a buffer and not necessarily the value the PWM generator is currently using. 0b - Disable fractional cycle placement for PWM_A. 1b - Enable fractional cycle placement for PWM_A.
1 FRAC1_EN	Fractional Cycle PWM Period Enable This bit is used to enable the fractional cycle length of the PWM period using the FRACVAL1 register. When disabled, the fractional cycle length of the PWM period is bypassed. **NOTE:** The FRAC1_EN bit is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. FRAC1_EN cannot be written when MCTRL[LDOK] is set. Reading FRAC1_EN reads the value in a buffer and not necessarily the value the PWM generator is currently using. 0b - Disable fractional cycle length for the PWM period. 1b - Enable fractional cycle length for the PWM period.
0 —	RESERVED

55.8.18 Output Control Register (SM0OCTRL - SM3OCTRL)

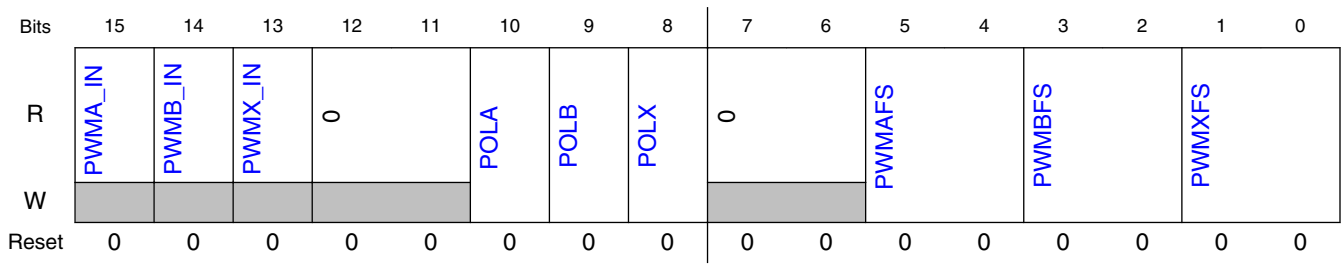
55.8.18.1 Offset

Register	Offset
SM0OCTRL	22h
SM1OCTRL	82h

Table continues on the next page...

Register	Offset
SM2OCTRL	E2h
SM3OCTRL	142h

55.8.18.2 Diagram



55.8.18.3 Fields

Field	Function
15 PWMA_IN	PWM_A Input This read only bit shows the logic value currently being driven into the PWM_A input. The bit's reset state is undefined.
14 PWMB_IN	PWM_B Input This read only bit shows the logic value currently being driven into the PWM_B input. The bit's reset state is undefined.
13 PWMX_IN	PWM_X Input This read only bit shows the logic value currently being driven into the PWM_X input. The bit's reset state is undefined.
12-11 —	RESERVED
10 POLA	PWM_A Output Polarity This bit inverts the PWM_A output polarity. 0b - PWM_A output not inverted. A high level on the PWM_A pin represents the "on" or "active" state. 1b - PWM_A output inverted. A low level on the PWM_A pin represents the "on" or "active" state.
9 POLB	PWM_B Output Polarity This bit inverts the PWM_B output polarity. 0b - PWM_B output not inverted. A high level on the PWM_B pin represents the "on" or "active" state. 1b - PWM_B output inverted. A low level on the PWM_B pin represents the "on" or "active" state.
8	PWM_X Output Polarity

Table continues on the next page...

Field	Function
POLX	This bit inverts the PWM_X output polarity. 0b - PWM_X output not inverted. A high level on the PWM_X pin represents the "on" or "active" state. 1b - PWM_X output inverted. A low level on the PWM_X pin represents the "on" or "active" state.
7-6 —	RESERVED
5-4 PWMAFS	PWM_A Fault State These bits determine the fault state for the PWM_A output during fault conditions and STOP mode. It may also define the output state during WAIT and DEBUG modes depending on the settings of CTRL2[WAITEN] and CTRL2[DBGEN]. 00b - Output is forced to logic 0 state prior to consideration of output polarity control. 01b - Output is forced to logic 1 state prior to consideration of output polarity control. 10, 11b - Output is tristated.
3-2 PWMBFS	PWM_B Fault State These bits determine the fault state for the PWM_B output during fault conditions and STOP mode. It may also define the output state during WAIT and DEBUG modes depending on the settings of CTRL2[WAITEN] and CTRL2[DBGEN]. 00b - Output is forced to logic 0 state prior to consideration of output polarity control. 01b - Output is forced to logic 1 state prior to consideration of output polarity control. 10, 11b - Output is tristated.
1-0 PWMXFS	PWM_X Fault State These bits determine the fault state for the PWM_X output during fault conditions and STOP mode. It may also define the output state during WAIT and DEBUG modes depending on the settings of CTRL2[WAITEN] and CTRL2[DBGEN]. 00b - Output is forced to logic 0 state prior to consideration of output polarity control. 01b - Output is forced to logic 1 state prior to consideration of output polarity control. 10, 11b - Output is tristated.

55.8.19 Status Register (SM0STS - SM3STS)

55.8.19.1 Offset

Register	Offset
SM0STS	24h
SM1STS	84h
SM2STS	E4h
SM3STS	144h

55.8.19.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	RUF	REF	RF	CFA1	CFA0	CFB1	CFB0	CFX1	CFX0	CMPF					
W			W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C	W1C					
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

55.8.19.3 Fields

Field	Function
15 —	RESERVED
14 RUF	Registers Updated Flag This read-only flag is set when one of the INIT, VALx, FRACVALx, or CTRL[PRSC] registers has been written, which indicates potentially non-coherent data in the set of double buffered registers. Clear this bit by a proper reload sequence consisting of a reload signal while MCTRL[LDOK] = 1. Reset clears this bit. 0b - No register update has occurred since last reload. 1b - At least one of the double buffered registers has been updated since the last reload.
13 REF	Reload Error Flag This read/write flag is set when a reload cycle occurs while MCTRL[LDOK] is 0 and the double buffered registers are in a non-coherent state (STS[RUF] = 1). Clear this bit by writing a logic one to this location. Reset clears this bit. 0b - No reload error occurred. 1b - Reload signal occurred with non-coherent data and MCTRL[LDOK] = 0.
12 RF	Reload Flag This read/write flag is set at the beginning of every reload cycle regardless of the state of MCTRL[LDOK]. Clear this bit by writing a logic one to this location when DMAEN[VALDE] is clear (non-DMA mode). This flag can also be cleared by the DMA done signal when DMAEN[VALDE] is set (DMA mode) . Reset clears this bit. 0b - No new reload cycle since last STS[RF] clearing 1b - New reload cycle since last STS[RF] clearing
11 CFA1	Capture Flag A1 This bit is set when a capture event occurs on the Capture A1 circuit . This bit is cleared by writing a one to this bit position if DMAEN[CA1DE] is clear (non-DMA mode) or by the DMA done signal if DMAEN[CA1DE] is set (DMA mode) . Reset clears this bit.
10 CFA0	Capture Flag A0 This bit is set when a capture event occurs on the Capture A0 circuit . This bit is cleared by writing a one to this bit position if DMAEN[CA0DE] is clear (non-DMA mode) or by the DMA done signal if DMAEN[CA0DE] is set (DMA mode) . Reset clears this bit.
9 CFB1	Capture Flag B1

Table continues on the next page...

Field	Function
	This bit is set when a capture event occurs on the Capture B1 circuit . This bit is cleared by writing a one to this bit position if DMAEN[CB1DE] is clear (non-DMA mode) or by the DMA done signal if DMAEN[CB1DE] is set (DMA mode) . Reset clears this bit.
8 CFB0	Capture Flag B0 This bit is set when a capture event occurs on the Capture B0 circuit . This bit is cleared by writing a one to this bit position if DMAEN[CB0DE] is clear (non-DMA mode) or by the DMA done signal if DMAEN[CB0DE] is set (DMA mode) . Reset clears this bit.
7 CFX1	Capture Flag X1 This bit is set when a capture event occurs on the Capture X1 circuit . This bit is cleared by writing a one to this bit position if DMAEN[CX1DE] is clear (non-DMA mode) or by the DMA done signal if DMAEN[CX1DE] is set (DMA mode) . Reset clears this bit.
6 CFX0	Capture Flag X0 This bit is set when a capture event occurs on the Capture X0 circuit . This bit is cleared by writing a one to this bit position if DMAEN[CX0DE] is clear (non-DMA mode) or by the DMA done signal if DMAEN[CX0DE] is set (DMA mode) . Reset clears this bit.
5-0 CMPF	Compare Flags These bits are set when the submodule counter value matches the value of one of the VALx registers. Clear these bits by writing a 1 to a bit position. 000000b - No compare event has occurred for a particular VALx value. 000001b - A compare event has occurred for a particular VALx value.

55.8.20 Interrupt Enable Register (SM0INTEN - SM3INTEN)

55.8.20.1 Offset

Register	Offset
SM0INTEN	26h
SM1INTEN	86h
SM2INTEN	E6h
SM3INTEN	146h

55.8.20.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0		REIE	RIE	CA1IE	CA0IE	CB1IE	CB0IE	CX1IE	CX0IE	CMPIE					
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

55.8.20.3 Fields

Field	Function
15-14 —	RESERVED
13 REIE	Reload Error Interrupt Enable This read/write bit enables the reload error flag, STS[REF], to generate CPU interrupt requests. Reset clears this bit. 0b - STS[REF] CPU interrupt requests disabled 1b - STS[REF] CPU interrupt requests enabled
12 RIE	Reload Interrupt Enable This read/write bit enables the reload flag, STS[RF], to generate CPU interrupt requests. Reset clears this bit. 0b - STS[RF] CPU interrupt requests disabled 1b - STS[RF] CPU interrupt requests enabled
11 CA1IE	Capture A 1 Interrupt Enable This bit allows the STS[CFA1] flag to create an interrupt request to the CPU. Do not set both this bit and DMAEN[CA1DE]. 0b - Interrupt request disabled for STS[CFA1]. 1b - Interrupt request enabled for STS[CFA1].
10 CA0IE	Capture A 0 Interrupt Enable This bit allows the STS[CFA0] flag to create an interrupt request to the CPU. Do not set both this bit and DMAEN[CA0DE]. 0b - Interrupt request disabled for STS[CFA0]. 1b - Interrupt request enabled for STS[CFA0].
9 CB1IE	Capture B 1 Interrupt Enable This bit allows the STS[CFB1] flag to create an interrupt request to the CPU. Do not set both this bit and DMAEN[CB1DE]. 0b - Interrupt request disabled for STS[CFB1]. 1b - Interrupt request enabled for STS[CFB1].
8 CB0IE	Capture B 0 Interrupt Enable This bit allows the STS[CFB0] flag to create an interrupt request to the CPU. Do not set both this bit and DMAEN[CB0DE]. 0b - Interrupt request disabled for STS[CFB0]. 1b - Interrupt request enabled for STS[CFB0].
7 CX1IE	Capture X 1 Interrupt Enable This bit allows the STS[CFX1] flag to create an interrupt request to the CPU. Do not set both this bit and DMAEN[CX1DE]. 0b - Interrupt request disabled for STS[CFX1]. 1b - Interrupt request enabled for STS[CFX1].
6 CX0IE	Capture X 0 Interrupt Enable This bit allows the STS[CFX0] flag to create an interrupt request to the CPU. Do not set both this bit and DMAEN[CX0DE].

Table continues on the next page...

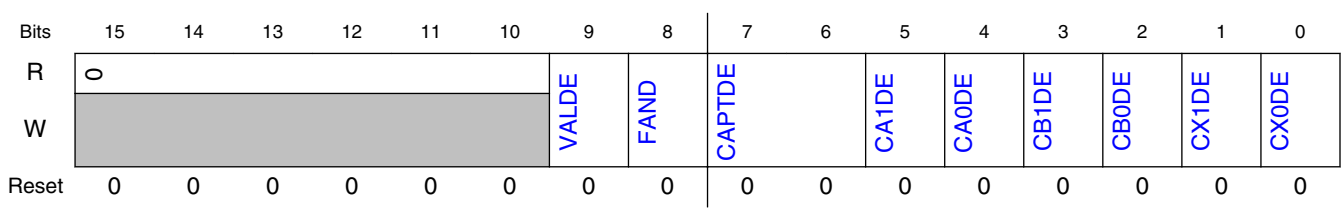
Field	Function
	0b - Interrupt request disabled for STS[CFX0]. 1b - Interrupt request enabled for STS[CFX0].
5-0 CMPIE	Compare Interrupt Enables These bits enable the STS[CMPF] flags to cause a compare interrupt request to the CPU. 000000b - The corresponding STS[CMPF] bit will not cause an interrupt request. 000001b - The corresponding STS[CMPF] bit will cause an interrupt request.

55.8.21 DMA Enable Register (SM0DMAEN - SM3DMAEN)

55.8.21.1 Offset

Register	Offset
SM0DMAEN	28h
SM1DMAEN	88h
SM2DMAEN	E8h
SM3DMAEN	148h

55.8.21.2 Diagram



55.8.21.3 Fields

Field	Function
15-10 —	RESERVED
9 VALDE	Value Registers DMA Enable This read/write bit enables DMA write requests for the VALx and FRACVALx registers when STS[RF] is set. Reset clears this bit. 0b - DMA write requests disabled

Table continues on the next page...

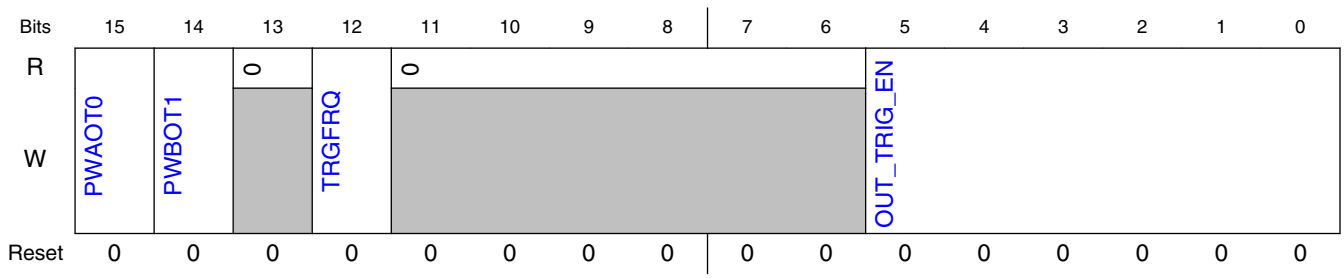
Field	Function
	1b - Enabled. DMA write requests for the VALx and FRACVALx registers enabled
8 FAND	FIFO Watermark AND Control This read/write bit works in conjunction with the DMAEN[CAPTDE] field when it is set to watermark mode (DMAEN[CAPTDE] = 01). While DMAEN[CAxDE], DMAEN[CBxDE], and DMAEN[CXxDE] determine which FIFO watermarks the DMA read request is sensitive to, this bit determines if the selected watermarks are AND'ed together or OR'ed together in order to create the request. 0b - Selected FIFO watermarks are OR'ed together. 1b - Selected FIFO watermarks are AND'ed together.
7-6 CAPTDE	Capture DMA Enable Source Select These read/write bits select the source of enabling the DMA read requests for the capture FIFOs. Reset clears these bits. 00b - Read DMA requests disabled. 01b - Exceeding a FIFO watermark sets the DMA read request. This requires at least one of DMAEN[CA1DE], DMAEN[CA0DE], DMAEN[CB1DE], DMAEN[CB0DE], DMAEN[CX1DE], or DMAEN[CX0DE] to also be set in order to determine to which watermark(s) the DMA request is sensitive. 10b - A local sync (VAL1 matches counter) sets the read DMA request. 11b - A local reload (STS[RF] being set) sets the read DMA request.
5 CA1DE	Capture A1 FIFO DMA Enable This read/write bit enables DMA read requests for the Capture A1 FIFO data when STS[CFA1] is set. Reset clears this bit. Do not set both this bit and INTEN[CA1IE].
4 CA0DE	Capture A0 FIFO DMA Enable This read/write bit enables DMA read requests for the Capture A0 FIFO data when STS[CFA0] is set. Reset clears this bit. Do not set both this bit and INTEN[CA0IE].
3 CB1DE	Capture B1 FIFO DMA Enable This read/write bit enables DMA read requests for the Capture B1 FIFO data when STS[CFB1] is set. Reset clears this bit. Do not set both this bit and INTEN[CB1IE].
2 CB0DE	Capture B0 FIFO DMA Enable This read/write bit enables DMA read requests for the Capture B0 FIFO data when STS[CFB0] is set. Reset clears this bit. Do not set both this bit and INTEN[CB0IE].
1 CX1DE	Capture X1 FIFO DMA Enable This read/write bit enables DMA read requests for the Capture X1 FIFO data when STS[CFX1] is set. Reset clears this bit. Do not set both this bit and INTEN[CX1IE].
0 CX0DE	Capture X0 FIFO DMA Enable This read/write bit enables DMA read requests for the Capture X0 FIFO data when STS[CFX0] is set. Reset clears this bit. Do not set both this bit and INTEN[CX0IE].

55.8.22 Output Trigger Control Register (SM0TCTRL - SM3TCTRL)

55.8.22.1 Offset

Register	Offset
SM0TCTRL	2Ah
SM1TCTRL	8Ah
SM2TCTRL	EAh
SM3TCTRL	14Ah

55.8.22.2 Diagram



55.8.22.3 Fields

Field	Function
15 PWAOT0	Output Trigger 0 Source Select This bit selects which signal to bring out on the PWM's PWM_OUT_TRIG0 port. The output trigger port is often connected to routing logic on the chip. This control bit allows the PWMA output signal to be driven onto the output trigger port so it can be sent to the chip routing logic. 0b - Route the PWM_OUT_TRIG0 signal to PWM_OUT_TRIG0 port. 1b - Route the PWMA output to the PWM_OUT_TRIG0 port.
14 PWBOT1	Output Trigger 1 Source Select This bit selects which signal to bring out on the PWM's PWM_OUT_TRIG1 port. The output trigger port is often connected to routing logic on the chip. This control bit allows the PWMB output signal to be driven onto the output trigger port so it can be sent to the chip routing logic. 0b - Route the PWM_OUT_TRIG1 signal to PWM_OUT_TRIG1 port. 1b - Route the PWMB output to the PWM_OUT_TRIG1 port.
13 —	RESERVED
12 TRGFRQ	Trigger frequency This read/write bit allows control over the frequency of the trigger outputs when using non-zero values of CTRL[LDFQ].

Table continues on the next page...

Field	Function
	0b - Trigger outputs are generated during every PWM period even if the PWM is not reloaded every period due to CTRL[LDFQ] being non-zero. 1b - Trigger outputs are generated only during the final PWM period prior to a reload opportunity when the PWM is not reloaded every period due to CTRL[LDFQ] being non-zero.
11-6 —	RESERVED
5-0 OUT_TRIG_EN	Output Trigger Enables These bits enable the generation of PWM_OUT_TRIG0 and PWM_OUT_TRIG1 outputs based on the counter value matching the value in one or more of the VAL0-5 registers. **NOTE:** Due to delays in creating the PWM outputs, the output trigger signals will lead the PWM output edges by 2-3 clock cycles depending on the fractional cycle value being used. 1xxxxxb - PWM_OUT_TRIG1 will set when the counter value matches the VAL5 value. x1xxxxb - PWM_OUT_TRIG0 will set when the counter value matches the VAL4 value. xx1xxxb - PWM_OUT_TRIG1 will set when the counter value matches the VAL3 value. xxx1xxb - PWM_OUT_TRIG0 will set when the counter value matches the VAL2 value. xxxx1xb - PWM_OUT_TRIG1 will set when the counter value matches the VAL1 value. xxxxx1b - PWM_OUT_TRIG0 will set when the counter value matches the VAL0 value.

55.8.23 Fault Disable Mapping Register 0 (SM0DISMAP0 - SM3DISMAP0)

55.8.23.1 Offset

Register	Offset
SM0DISMAP0	2Ch
SM1DISMAP0	8Ch
SM2DISMAP0	ECh
SM3DISMAP0	14Ch

55.8.23.2 Function

This register determines which PWM pins are disabled by the fault protection inputs. Reset sets all of the bits in the fault disable mapping register.

55.8.23.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	1				DIS0X				DIS0B				DIS0A			
W																
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

55.8.23.4 Fields

Field	Function
15-12 —	RESERVED
11-8 DIS0X	PWM_X Fault Disable Mask 0 Each of the four bits of this read/write field is one-to-one associated with the four FAULTx inputs of fault channel 0. The PWM_X output is turned off if there is a logic 1 on a FAULTx input and a 1 in the corresponding bit of this field. A reset sets all bits in this field.
7-4 DIS0B	PWM_B Fault Disable Mask 0 Each of the four bits of this read/write field is one-to-one associated with the four FAULTx inputs of fault channel 0. The PWM_B output is turned off if there is a logic 1 on a FAULTx input and a 1 in the corresponding bit of this field. A reset sets all bits in this field.
3-0 DIS0A	PWM_A Fault Disable Mask 0 Each of the four bits of this read/write field is one-to-one associated with the four FAULTx inputs of fault channel 0. The PWM_A output is turned off if there is a logic 1 on a FAULTx input and a 1 in the corresponding bit of this field. A reset sets all bits in this field.

55.8.24 Deadtime Count Register 0 (SM0DTCNT0 - SM3DTCNT0)

55.8.24.1 Offset

Register	Offset
SM0DTCNT0	30h
SM1DTCNT0	90h
SM2DTCNT0	F0h
SM3DTCNT0	150h

55.8.24.2 Function

Deadtime operation applies only to complementary channel operation. The values written to the DTCNTx registers are in terms of IPBus clock cycles regardless of the setting of CTRL[PRSC] and/or CTRL2[CLK_SEL]. Reset sets the deadtime count registers to a default value of 0x07FF, selecting a deadtime of 2047 IPBus clock cycles. The DTCNTx registers are not byte accessible.

55.8.24.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DTCNT0															
Reset	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1

55.8.24.4 Fields

Field	Function
15-0 DTCNT0	DTCNT0 The DTCNT0 field is interpreted differently depending on whether or not the fractional delays are being used (Set FRCTRL[FRAC23_EN] to 1 when MCTRL[IPOL]=0, or set FRCTRL[FRAC45_EN] to 1 when MCTRL[IPOL]=1). If the fractional delays are off, then the upper 5 bits of DTCNT0 are ignored and the remaining 11 bits are used to specify the number of cycles of deadtime. In this case the maximum value is 0x07FF which indicates 2047 cycles of deadtime. If the fractional delays are being used, then the upper 11 bits of DTCNT0 represent the number of whole cycles of deadtime and the lower 5 bits of each register represent the fractional cycle added to the whole number. In this case the maximum value is 0xFFFF which represents 2047 31/32 cycles of deadtime. The DTCNT0 field is used to control the deadtime during 0 to 1 transitions of the PWM_A output (assuming normal polarity).

55.8.25 Deadtime Count Register 1 (SM0DTCNT1 - SM3DTCNT1)

55.8.25.1 Offset

Register	Offset
SM0DTCNT1	32h

Table continues on the next page...

Register	Offset
SM1DTCNT1	92h
SM2DTCNT1	F2h
SM3DTCNT1	152h

55.8.25.2 Function

Deadtime operation applies only to complementary channel operation. The values written to the DTCNTx registers are in terms of IPBus clock cycles regardless of the setting of CTRL[PRSC] and/or CTRL2[CLK_SEL]. Reset sets the deadtime count registers to a default value of 0x07FF, selecting a deadtime of 2047 IPBus clock cycles. The DTCNTx registers are not byte accessible.

55.8.25.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	DTCNT1															
Reset	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1

55.8.25.4 Fields

Field	Function
15-0 DTCNT1	DTCNT1 The DTCNT1 field is interpreted differently depending on whether or not the fractional delays are being used (Set FRCTRL[FRAC23_EN] to 1 when MCTRL[IPOL]=0, or set FRCTRL[FRAC45_EN] to 1 when MCTRL[IPOL]=1). If the fractional delays are off, then the upper 5 bits of DTCNT1 are ignored and the remaining 11 bits are used to specify the number of cycles of deadtime. In this case the maximum value is 0x07FF which indicates 2047 cycles of deadtime. If the fractional delays are being used, then the upper 11 bits of DTCNT1 represent the number of whole cycles of deadtime and the lower 5 bits of each register represent the fractional cycle added to the whole number. In this case the maximum value is 0xFFFF which represents 2047 31/32 cycles of deadtime. The DTCNT1 field is used to control the deadtime during 0 to 1 transitions of the PWM_B output (assuming normal polarity).

55.8.26 Capture Control A Register (SM0CAPTCTRLA - SM3CAPTCTRLA)

55.8.26.1 Offset

Register	Offset
SM0CAPTCTRLA	34h
SM1CAPTCTRLA	94h
SM2CAPTCTRLA	F4h
SM3CAPTCTRLA	154h

55.8.26.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CA1CNT			CA0CNT			CFAWM		EDGCNTA_EN	INP_SEL A	EDGA1		EDGA0		ONESHOTA	ARMA
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

55.8.26.3 Fields

Field	Function
15-13 CA1CNT	Capture A1 FIFO Word Count This field reflects the number of words in the Capture A1 FIFO. (FIFO depth is 1.)
12-10 CA0CNT	Capture A0 FIFO Word Count This field reflects the number of words in the Capture A0 FIFO. (FIFO depth is 1.)
9-8 CFAWM	Capture A FIFOs Water Mark This field represents the water mark level for capture A FIFOs. The capture flags, STS[CFA1] and STS[CFA0], are not set until the word count of the corresponding FIFO is greater than this water mark level. (FIFO depth is 1.)
7 EDGCNTA_EN	Edge Counter A Enable This bit enables the edge counter which counts rising and falling edges on the PWM_A input signal. 0b - Edge counter disabled and held in reset 1b - Edge counter enabled
6	Input Select A

Table continues on the next page...

Field	Function
INP_SELA	This bit selects between the raw PWM_A input signal and the output of the edge counter/compare circuitry as the source for the input capture circuit. 0b - Raw PWM_A input signal selected as source. 1b - Edge Counter. Output of edge counter/compare selected as source. Note that when this bitfield is set to 1, the internal edge counter is enabled and the rising and/or falling edges specified by the CAPTCTRLA[EDGA0] and CAPTCTRLA[EDGA1] fields are ignored. The software must still place a value other than 00 in either or both of the CAPTCTLRA[EDGA0] and/or CAPTCTRLA[EDGA1] fields in order to enable one or both of the capture registers.
5-4 EDGA1	Edge A 1 These bits control the input capture 1 circuitry by determining which input edges cause a capture event. 00b - Disabled 01b - Capture falling edges 10b - Capture rising edges 11b - Capture any edge
3-2 EDGA0	Edge A 0 These bits control the input capture 0 circuitry by determining which input edges cause a capture event. 00b - Disabled 01b - Capture falling edges 10b - Capture rising edges 11b - Capture any edge
1 ONESHOTA	One Shot Mode A This bit selects between free running and one shot mode for the input capture circuitry. 0b - Free Running. Free running mode is selected. If both capture circuits are enabled, then capture circuit 0 is armed first after CAPTCTRLA[ARMA] is set. Once a capture occurs, capture circuit 0 is disarmed and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed and capture circuit 0 is re-armed. The process continues indefinitely. If only one of the capture circuits is enabled, then captures continue indefinitely on the enabled capture circuit. 1b - One Shot. One shot mode is selected. If both capture circuits are enabled, then capture circuit 0 is armed first after CAPTCTRLA[ARMA] is set. Once a capture occurs, capture circuit 0 is disarmed and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed and CAPTCTRLA[ARMA] is cleared. No further captures will be performed until CAPTCTRLA[ARMA] is set again. If only one of the capture circuits is enabled, then a single capture will occur on the enabled capture circuit and CAPTCTRLA[ARMA] is then cleared.
0 ARMA	Arm A Setting this bit high starts the input capture process. This bit can be cleared at any time to disable input capture operation. This bit is self-cleared when in one shot mode and one or more of the enabled capture circuits has had a capture event. 0b - Input capture operation is disabled. 1b - Input capture operation as specified by CAPTCTRLA[EDGAx] is enabled.

55.8.27 Capture Compare A Register (SM0CAPTCOMPA - SM3CAPTCOMPA)

55.8.27.1 Offset

Register	Offset
SM0CAPTCOMPA	36h
SM1CAPTCOMPA	96h
SM2CAPTCOMPA	F6h
SM3CAPTCOMPA	156h

55.8.27.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	EDGCNTA								EDGCMPA							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

55.8.27.3 Fields

Field	Function
15-8 EDGCNTA	Edge Counter A This read-only field contains the edge counter value for the PWM_A input capture circuitry.
7-0 EDGCMPA	Edge Compare A This read/write field is the compare value associated with the edge counter for the PWM_A input capture circuitry.

55.8.28 Capture Control B Register (SM0CAPTCTRLB - SM3CAPTCTRLB)

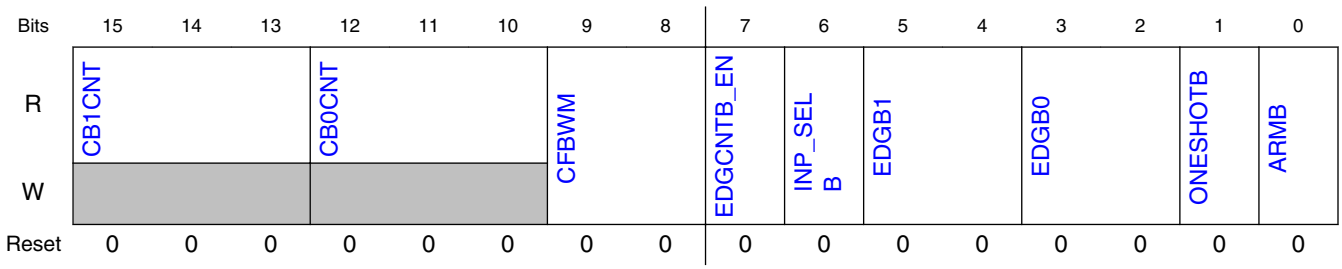
55.8.28.1 Offset

Register	Offset
SM0CAPTCTRLB	38h
SM1CAPTCTRLB	98h
SM2CAPTCTRLB	F8h

Table continues on the next page...

Register	Offset
SM3CAPTCTRLB	158h

55.8.28.2 Diagram



55.8.28.3 Fields

Field	Function
15-13 CB1CNT	Capture B1 FIFO Word Count This field reflects the number of words in the Capture B1 FIFO. (FIFO depth is 1.)
12-10 CB0CNT	Capture B0 FIFO Word Count This field reflects the number of words in the Capture B0 FIFO. (FIFO depth is 1.)
9-8 CFBWM	Capture B FIFOs Water Mark This field represents the water mark level for capture B FIFOs. The capture flags, STS[CFB1] and STS[CFB0], won't be set until the word count of the corresponding FIFO is greater than this water mark level. (FIFO depth is 1.)
7 EDGCNTB_EN	Edge Counter B Enable This bit enables the edge counter which counts rising and falling edges on the PWM_B input signal. 0b - Edge counter disabled and held in reset 1b - Edge counter enabled
6 INP_SELB	Input Select B This bit selects between the raw PWM_B input signal and the output of the edge counter/compare circuitry as the source for the input capture circuit. 0b - Raw PWM_B input signal selected as source. 1b - Edge Counter. Output of edge counter/compare selected as source. Note that when this bitfield is set to 1, the internal edge counter is enabled and the rising and/or falling edges specified by the CAPTCTRLB[EDGB0] and CAPTCTRLB[EDGB1] fields are ignored. The software must still place a value other than 00 in either or both of the CAPTCTLRB[EDGB0] and/or CAPTCTRLB[EDGB1] fields in order to enable one or both of the capture registers.
5-4 EDGB1	Edge B 1 These bits control the input capture 1 circuitry by determining which input edges cause a capture event. 00b - Disabled

Table continues on the next page...

Field	Function
	01b - Capture falling edges 10b - Capture rising edges 11b - Capture any edge
3-2 EDGB0	Edge B 0 These bits control the input capture 0 circuitry by determining which input edges cause a capture event. 00b - Disabled 01b - Capture falling edges 10b - Capture rising edges 11b - Capture any edge
1 ONESHOTB	One Shot Mode B This bit selects between free running and one shot mode for the input capture circuitry. 0b - Free Running. Free running mode is selected. If both capture circuits are enabled, then capture circuit 0 is armed first after CAPTCTRLB[ARMB] is set. Once a capture occurs, capture circuit 0 is disarmed and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed and capture circuit 0 is re-armed. The process continues indefinitely. If only one of the capture circuits is enabled, then captures continue indefinitely on the enabled capture circuit. 1b - One Shot. One shot mode is selected. If both capture circuits are enabled, then capture circuit 0 is armed first after CAPTCTRLB[ARMB] is set. Once a capture occurs, capture circuit 0 is disarmed and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed and CAPTCTRLB[ARMB] is cleared. No further captures will be performed until CAPTCTRLB[ARMB] is set again. If only one of the capture circuits is enabled, then a single capture will occur on the enabled capture circuit and CAPTCTRLB[ARMB] is then cleared.
0 ARMB	Arm B Setting this bit high starts the input capture process. This bit can be cleared at any time to disable input capture operation. This bit is self-cleared when in one shot mode and one or more of the enabled capture circuits has had a capture event. 0b - Input capture operation is disabled. 1b - Input capture operation as specified by CAPTCTRLB[EDGBx] is enabled.

55.8.29 Capture Compare B Register (SM0CAPTCOMPB - SM3CAPTCOMPB)

55.8.29.1 Offset

Register	Offset
SM0CAPTCOMPB	3Ah
SM1CAPTCOMPB	9Ah
SM2CAPTCOMPB	FAh
SM3CAPTCOMPB	15Ah

55.8.29.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	EDGCNTB								EDGCMPB							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

55.8.29.3 Fields

Field	Function
15-8 EDGCNTB	Edge Counter B This read-only field contains the edge counter value for the PWM_B input capture circuitry.
7-0 EDGCMPB	Edge Compare B This read/write field is the compare value associated with the edge counter for the PWM_B input capture circuitry.

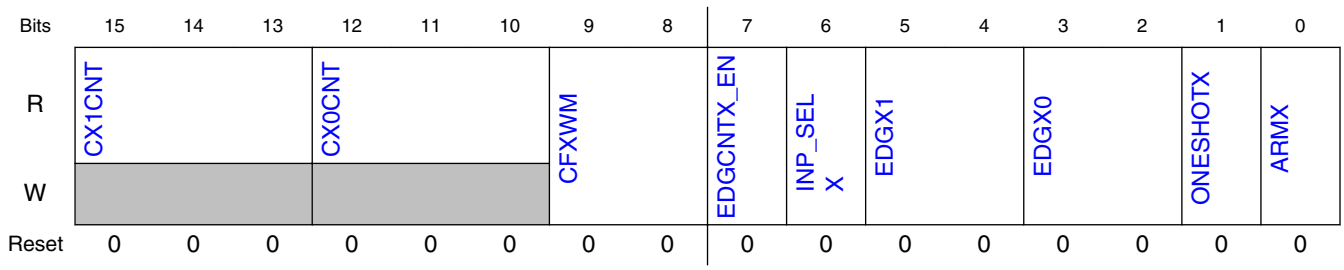
55.8.30 Capture Control X Register (SM0CAPTCTRLX - SM3CAPTCTRLX)

X pins have limited usability as input capture - see note on pin 3103

55.8.30.1 Offset

Register	Offset
SM0CAPTCTRLX	3Ch
SM1CAPTCTRLX	9Ch
SM2CAPTCTRLX	FCh
SM3CAPTCTRLX	15Ch

55.8.30.2 Diagram



X pins have limited usability as input capture - see note on pin 3103

55.8.30.3 Fields

Field	Function
15-13 CX1CNT	Capture X1 FIFO Word Count This field reflects the number of words in the Capture X1 FIFO. (FIFO depth is 1.)
12-10 CX0CNT	Capture X0 FIFO Word Count This field reflects the number of words in the Capture X0 FIFO. (FIFO depth is 1.)
9-8 CFXWM	Capture X FIFOs Water Mark This field represents the water mark level for capture X FIFOs. The capture flags, STS[CFX1] and STS[CFX0], won't be set until the word count of the corresponding FIFO is greater than this water mark level. (FIFO depth is 1.)
7 EDGCNTX_EN	Edge Counter X Enable This bit enables the edge counter which counts rising and falling edges on the PWM_X input signal. 0b - Edge counter disabled and held in reset 1b - Edge counter enabled
6 INP_SELX	Input Select X This bit selects between the raw PWM_X input signal and the output of the edge counter/compare circuitry as the source for the input capture circuit. 0b - Raw PWM_X input signal selected as source. 1b - Edge Counter. Output of edge counter/compare selected as source. Note that when this bitfield is set to 1, the internal edge counter is enabled and the rising and/or falling edges specified by the CAPTCTRLX[EDGX0] and CAPTCTRLX[EDGX1] fields are ignored. The software must still place a value other than 00 in either or both of the CAPTCTLRX[EDGX0] and/or CAPTCTRLX[EDGX1] fields in order to enable one or both of the capture registers.
5-4 EDGX1	Edge X 1 These bits control the input capture 1 circuitry by determining which input edges cause a capture event. 00b - Disabled 01b - Capture falling edges 10b - Capture rising edges 11b - Capture any edge
3-2 EDGX0	Edge X 0 These bits control the input capture 0 circuitry by determining which input edges cause a capture event. 00b - Disabled

Table continues on the next page...

Field	Function
	01b - Capture falling edges 10b - Capture rising edges 11b - Capture any edge
1 ONESHOTX	One Shot Mode Aux This bit selects between free running and one shot mode for the input capture circuitry. 0b - Free Running. Free running mode is selected. If both capture circuits are enabled, then capture circuit 0 is armed first after the ARMX bit is set. Once a capture occurs, capture circuit 0 is disarmed and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed and capture circuit 0 is re-armed. The process continues indefinitely. If only one of the capture circuits is enabled, then captures continue indefinitely on the enabled capture circuit. 1b - One Shot. One shot mode is selected. If both capture circuits are enabled, then capture circuit 0 is armed first after the ARMX bit is set. Once a capture occurs, capture circuit 0 is disarmed and capture circuit 1 is armed. After capture circuit 1 performs a capture, it is disarmed and the ARMX bit is cleared. No further captures will be performed until the ARMX bit is set again.If only one of the capture circuits is enabled, then a single capture will occur on the enabled capture circuit and the ARMX bit is then cleared.
0 ARMX	Arm X Setting this bit high starts the input capture process. This bit can be cleared at any time to disable input capture operation. This bit is self-cleared when in one shot mode and one or more of the enabled capture circuits has had a capture event. 0b - Input capture operation is disabled. 1b - Input capture operation as specified by CAPTCTRLX[EDGXx] is enabled.

55.8.31 Capture Compare X Register (SM0CAPTCOMPX - SM3CAPTCOMPX)

55.8.31.1 Offset

Register	Offset
SM0CAPTCOMPX	3Eh
SM1CAPTCOMPX	9Eh
SM2CAPTCOMPX	FEh
SM3CAPTCOMPX	15Eh

X pins have limited usability as input capture - see note on pin 3103

55.8.31.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	EDGCNTX								EDGCMPX							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

55.8.31.3 Fields

Field	Function
15-8 EDGCNTX	Edge Counter X This read-only field contains the edge counter value for the PWM_X input capture circuitry.
7-0 EDGCMPX	Edge Compare X This read/write field is the compare value associated with the edge counter for the PWM_X input capture circuitry.

55.8.32 Capture Value 0 Register (SM0CVAL0 - SM3CVAL0)

55.8.32.1 Offset

Register	Offset
SM0CVAL0	40h
SM1CVAL0	A0h
SM2CVAL0	100h
SM3CVAL0	160h

55.8.32.2 Function

Writing this register generates bus transfer error.

55.8.32.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CAPTVAL0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

55.8.32.4 Fields

Field	Function
15-0 CAPTVAL0	CAPTVAL0 This read-only register stores the value captured from the submodule counter. Exactly when this capture occurs is defined by CAPTCTRLX[EDGX0]. Each capture will increase the value of CAPTCTRLX[CX0CNT] by 1 until the maximum value is reached. Each read of this register will decrease the value of CAPTCTRLX[CX0CNT] by 1 until 0 is reached. This register is not byte accessible.

55.8.33 Capture Value 0 Cycle Register (SM0CVAL0CYC - SM3CVAL0CYC)

55.8.33.1 Offset

Register	Offset
SM0CVAL0CYC	42h
SM1CVAL0CYC	A2h
SM2CVAL0CYC	102h
SM3CVAL0CYC	162h

55.8.33.2 Function

Writing this register generates bus transfer error.

55.8.33.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0												CVAL0CYC			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

55.8.33.4 Fields

Field	Function
15-4 —	RESERVED
3-0 CVAL0CYC	CVAL0CYC This read-only register stores the cycle number corresponding to the value captured in CVAL0. This register is incremented each time the counter is loaded with the INIT value at the end of a PWM modulo cycle.

55.8.34 Capture Value 1 Register (SM0CVAL1 - SM3CVAL1)

55.8.34.1 Offset

Register	Offset
SM0CVAL1	44h
SM1CVAL1	A4h
SM2CVAL1	104h
SM3CVAL1	164h

55.8.34.2 Function

Writing this register generates bus transfer error.

55.8.34.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CAPTVAL1															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

55.8.34.4 Fields

Field	Function
15-0 CAPTVAL1	CAPTVAL1 This read-only register stores the value captured from the submodule counter. Exactly when this capture occurs is defined by CAPTCTRLX[EDGX1]. Each capture increases the value of CAPTCTRLX[CX1CNT] by 1 until the maximum value is reached. Each read of this register decreases the value of CAPTCTRLX[CX1CNT] by 1 until 0 is reached. This register is not byte accessible.

55.8.35 Capture Value 1 Cycle Register (SM0CVAL1CYC - SM3CVAL1CYC)

55.8.35.1 Offset

Register	Offset
SM0CVAL1CYC	46h
SM1CVAL1CYC	A6h
SM2CVAL1CYC	106h
SM3CVAL1CYC	166h

55.8.35.2 Function

Writing this register generates bus transfer error.

55.8.35.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0												CVAL1CYC			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

55.8.35.4 Fields

Field	Function
15-4 —	RESERVED
3-0 CVAL1CYC	CVAL1CYC This read-only register stores the cycle number corresponding to the value captured in CVAL1. This register is incremented each time the counter is loaded with the INIT value at the end of a PWM modulo cycle.

55.8.36 Capture Value 2 Register (SM0CVAL2 - SM3CVAL2)

55.8.36.1 Offset

Register	Offset
SM0CVAL2	48h
SM1CVAL2	A8h
SM2CVAL2	108h
SM3CVAL2	168h

55.8.36.2 Function

Writing this register generates bus transfer error.

55.8.36.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CAPTVAL2															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

55.8.36.4 Fields

Field	Function
15-0 CAPTVAL2	CAPTVAL2 This read-only register stores the value captured from the submodule counter. Exactly when this capture occurs is defined by CAPTCTRLA[EDGA0]. Each capture increases the value of CAPTCTRLA[CA0CNT] by 1 until the maximum value is reached. Each read of this register decreases the value of CAPTCTRLA[CA0CNT] by 1 until 0 is reached. This register is not byte accessible.

55.8.37 Capture Value 2 Cycle Register (SM0CVAL2CYC - SM3CVAL2CYC)

55.8.37.1 Offset

Register	Offset
SM0CVAL2CYC	4Ah
SM1CVAL2CYC	AAh
SM2CVAL2CYC	10Ah
SM3CVAL2CYC	16Ah

55.8.37.2 Function

Writing this register generates bus transfer error.

55.8.37.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0												CVAL2CYC			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

55.8.37.4 Fields

Field	Function
15-4 —	RESERVED
3-0 CVAL2CYC	CVAL2CYC This read-only register stores the cycle number corresponding to the value captured in CVAL2. This register is incremented each time the counter is loaded with the INIT value at the end of a PWM modulo cycle.

55.8.38 Capture Value 3 Register (SM0CVAL3 - SM3CVAL3)

55.8.38.1 Offset

Register	Offset
SM0CVAL3	4Ch
SM1CVAL3	ACh
SM2CVAL3	10Ch
SM3CVAL3	16Ch

55.8.38.2 Function

Writing this register generates bus transfer error.

55.8.38.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CAPTVAL3															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

55.8.38.4 Fields

Field	Function
15-0 CAPTVAL3	CAPTVAL3 This read-only register stores the value captured from the submodule counter. Exactly when this capture occurs is defined by CAPTCTRLA[EDGA1]. Each capture increases the value of CAPTCTRLA[CA1CNT] by 1 until the maximum value is reached. Each read of this register decreases the value of CAPTCTRLA[CA1CNT] by 1 until 0 is reached. This register is not byte accessible.

55.8.39 Capture Value 3 Cycle Register (SM0CVAL3CYC - SM3CVAL3CYC)

55.8.39.1 Offset

Register	Offset
SM0CVAL3CYC	4Eh
SM1CVAL3CYC	AEh
SM2CVAL3CYC	10Eh
SM3CVAL3CYC	16Eh

55.8.39.2 Function

Writing this register generates bus transfer error.

55.8.39.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0												CVAL3CYC			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

55.8.39.4 Fields

Field	Function
15-4 —	RESERVED
3-0 CVAL3CYC	CVAL3CYC This read-only register stores the cycle number corresponding to the value captured in CVAL3. This register is incremented each time the counter is loaded with the INIT value at the end of a PWM modulo cycle.

55.8.40 Capture Value 4 Register (SM0CVAL4 - SM3CVAL4)

55.8.40.1 Offset

Register	Offset
SM0CVAL4	50h
SM1CVAL4	B0h
SM2CVAL4	110h
SM3CVAL4	170h

55.8.40.2 Function

Writing this register generates bus transfer error.

55.8.40.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CAPTVAL4															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

55.8.40.4 Fields

Field	Function
15-0 CAPTVAL4	CAPTVAL4 This read-only register stores the value captured from the submodule counter. Exactly when this capture occurs is defined by CAPTCTRLB[EDGB0]. Each capture increases the value of CAPTCTRLB[CB0CNT] by 1 until the maximum value is reached. Each read of this register decreases the value of CAPTCTRLB[CB0CNT] by 1 until 0 is reached. This register is not byte accessible.

55.8.41 Capture Value 4 Cycle Register (SM0CVAL4CYC - SM3CVAL4CYC)

55.8.41.1 Offset

Register	Offset
SM0CVAL4CYC	52h
SM1CVAL4CYC	B2h
SM2CVAL4CYC	112h
SM3CVAL4CYC	172h

55.8.41.2 Function

Writing this register generates bus transfer error.

55.8.41.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0												CVAL4CYC			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

55.8.41.4 Fields

Field	Function
15-4 —	RESERVED
3-0 CVAL4CYC	CVAL4CYC This read-only register stores the cycle number corresponding to the value captured in CVAL4. This register is incremented each time the counter is loaded with the INIT value at the end of a PWM modulo cycle.

55.8.42 Capture Value 5 Register (SM0CVAL5 - SM3CVAL5)

55.8.42.1 Offset

Register	Offset
SM0CVAL5	54h
SM1CVAL5	B4h
SM2CVAL5	114h
SM3CVAL5	174h

55.8.42.2 Function

Writing this register generates bus transfer error.

55.8.42.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CAPTVAL5															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

55.8.42.4 Fields

Field	Function
15-0 CAPTVAL5	CAPTVAL5 This read-only register stores the value captured from the submodule counter. Exactly when this capture occurs is defined by CAPTCTRLB[EDGB1]. Each capture increases the value of CAPTCTRLB[CB1CNT] by 1 until the maximum value is reached. Each read of this register decreases the value of CAPTCTRLB[CB1CNT] by 1 until 0 is reached. This register is not byte accessible.

55.8.43 Capture Value 5 Cycle Register (SM0CVAL5CYC - SM3CVAL5CYC)

55.8.43.1 Offset

Register	Offset
SM0CVAL5CYC	56h
SM1CVAL5CYC	B6h
SM2CVAL5CYC	116h
SM3CVAL5CYC	176h

55.8.43.2 Function

Writing this register generates bus transfer error.

55.8.43.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0												CVAL5CYC			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

55.8.43.4 Fields

Field	Function
15-4 —	RESERVED
3-0 CVAL5CYC	CVAL5CYC This read-only register stores the cycle number corresponding to the value captured in CVAL5. This register is incremented each time the counter is loaded with the INIT value at the end of a PWM modulo cycle.

55.8.44 Output Enable Register (OUTEN)

55.8.44.1 Offset

Register	Offset
OUTEN	180h

55.8.44.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0				PWMA_EN				PWMB_EN				PWMX_EN			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

55.8.44.3 Fields

Field	Function
15-12 —	RESERVED
11-8 PWMA_EN	PWM_A Output Enables The four bits of this field enable the PWM_A outputs of submodules 3-0, respectively. These bits should be set to 0 (output disabled) when a PWM_A pin is being used for input capture. 0000b - PWM_A output disabled. 0001b - PWM_A output enabled.

Table continues on the next page...

Field	Function
7-4 PWMB_EN	PWM_B Output Enables The four bits of this field enable the PWM_B outputs of submodules 3-0, respectively. These bits should be set to 0 (output disabled) when a PWM_B pin is being used for input capture. 0000b - PWM_B output disabled. 0001b - PWM_B output enabled.
3-0 PWMX_EN	PWM_X Output Enables The four bits of this field enable the PWM_X outputs of submodules 3-0, respectively. These bits should be set to 0 (output disabled) when a PWM_X pin is being used for input capture or deadtime correction. 0000b - PWM_X output disabled. 0001b - PWM_X output enabled.

55.8.45 Mask Register (MASK)

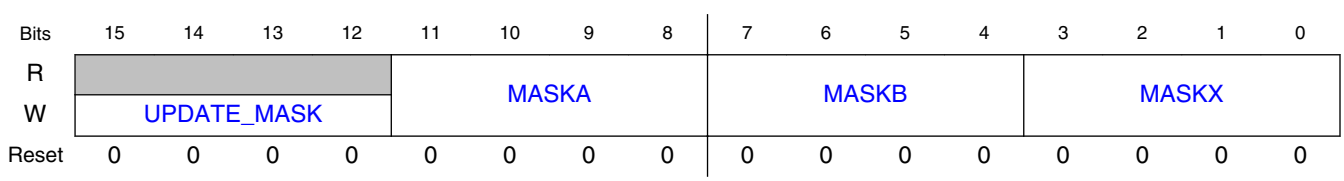
55.8.45.1 Offset

Register	Offset
MASK	182h

55.8.45.2 Function

MASK is double buffered and does not take effect until a FORCE_OUT event occurs within the appropriate submodule. Reading MASK reads the buffered values and not necessarily the values currently in effect. This double buffering can be overridden by setting the UPDATE_MASK bits.

55.8.45.3 Diagram



55.8.45.4 Fields

Field	Function
15-12 UPDATE_MASK	Update Mask Bits Immediately The four bits mask the PWM_X outputs of submodules 3-0, respectively, The four bits of this field force the MASK* bits to be immediately updated within submodules 3-0, respectively, without waiting for a FORCE_OUT event. These self-clearing bits always read as zero. Software may write to any or all of these bits and may set these bits in the same write operation that updates the MASKA, MASKB, and MASKX fields of this register. 0000b - Normal operation. MASK* bits within the corresponding submodule are not updated until a FORCE_OUT event occurs within the submodule. 0001b - Immediate operation. MASK* bits within the corresponding submodule are updated on the following clock edge after setting this bit.
11-8 MASKA	PWM_A Masks The four bits of this field mask the PWM_A outputs of submodules 3-0, respectively, forcing the output to logic 0 prior to consideration of the output polarity. 0000b - PWM_A output normal. 0001b - PWM_A output masked.
7-4 MASKB	PWM_B Masks The four bits of this field mask the PWM_B outputs of submodules 3-0, respectively, forcing the output to logic 0 prior to consideration of the output polarity. 0000b - PWM_B output normal. 0001b - PWM_B output masked.
3-0 MASKX	PWM_X Masks The four bits of this field mask the PWM_X outputs of submodules 3-0, respectively, forcing the output to logic 0 prior to consideration of the output polarity. 0000b - PWM_X output normal. 0001b - PWM_X output masked.

55.8.46 Software Controlled Output Register (SWCOUT)

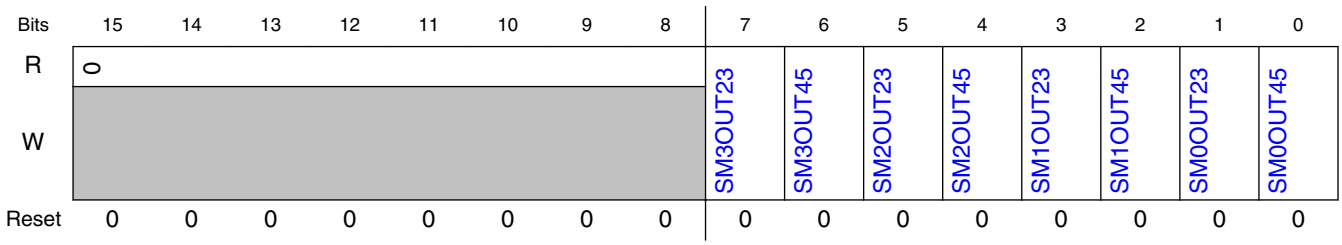
55.8.46.1 Offset

Register	Offset
SWCOUT	184h

55.8.46.2 Function

These bits are double buffered and do not take effect until a FORCE_OUT event occurs within the appropriate submodule. Reading these bits reads the buffered value and not necessarily the value currently in effect.

55.8.46.3 Diagram



55.8.46.4 Fields

Field	Function
15-8 —	RESERVED
7 SM3OUT23	Submodule 3 Software Controlled Output 23 This bit is only used when DTSRCSEL[SM3SEL23] is set to 0b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0b - A logic 0 is supplied to the deadtime generator of submodule 3 instead of PWM23. 1b - A logic 1 is supplied to the deadtime generator of submodule 3 instead of PWM23.
6 SM3OUT45	Submodule 3 Software Controlled Output 45 This bit is only used when DTSRCSEL[SM3SEL45] is set to b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0b - A logic 0 is supplied to the deadtime generator of submodule 3 instead of PWM45. 1b - A logic 1 is supplied to the deadtime generator of submodule 3 instead of PWM45.
5 SM2OUT23	Submodule 2 Software Controlled Output 23 This bit is only used when DTSRCSEL[SM2SEL23] is set to b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0b - A logic 0 is supplied to the deadtime generator of submodule 2 instead of PWM23. 1b - A logic 1 is supplied to the deadtime generator of submodule 2 instead of PWM23.
4 SM2OUT45	Submodule 2 Software Controlled Output 45 This bit is only used when DTSRCSEL[SM2SEL45] is set to b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0b - A logic 0 is supplied to the deadtime generator of submodule 2 instead of PWM45. 1b - A logic 1 is supplied to the deadtime generator of submodule 2 instead of PWM45.

Table continues on the next page...

Field	Function
3 SM1OUT23	Submodule 1 Software Controlled Output 23 This bit is only used when DTSRCSEL[SM1SEL23] is set to b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0b - A logic 0 is supplied to the deadtime generator of submodule 1 instead of PWM23. 1b - A logic 1 is supplied to the deadtime generator of submodule 1 instead of PWM23.
2 SM1OUT45	Submodule 1 Software Controlled Output 45 This bit is only used when DTSRCSEL[SM1SEL45] is set to b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0b - A logic 0 is supplied to the deadtime generator of submodule 1 instead of PWM45. 1b - A logic 1 is supplied to the deadtime generator of submodule 1 instead of PWM45.
1 SM0OUT23	Submodule 0 Software Controlled Output 23 This bit is only used when DTSRCSEL[SM0SEL23] is set to b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0b - A logic 0 is supplied to the deadtime generator of submodule 0 instead of PWM23. 1b - A logic 1 is supplied to the deadtime generator of submodule 0 instead of PWM23.
0 SM0OUT45	Submodule 0 Software Controlled Output 45 This bit is only used when DTSRCSEL[SM0SEL45] is set to b10. It allows software control of which signal is supplied to the deadtime generator of that submodule. 0b - A logic 0 is supplied to the deadtime generator of submodule 0 instead of PWM45. 1b - A logic 1 is supplied to the deadtime generator of submodule 0 instead of PWM45.

55.8.47 PWM Source Select Register (DTSRCSEL)

55.8.47.1 Offset

Register	Offset
DTSRCSEL	186h

55.8.47.2 Function

The PWM source select bits are double buffered and do not take effect until a FORCE_OUT event occurs within the appropriate submodule. Reading these bits reads the buffered value and not necessarily the value currently in effect.

55.8.47.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SM3SEL23		SM3SEL45		SM2SEL23		SM2SEL45		SM1SEL23		SM1SEL45		SM0SEL23		SM0SEL45	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

55.8.47.4 Fields

Field	Function
15-14 SM3SEL23	Submodule 3 PWM23 Control Select This field selects possible over-rides to the generated SM3PWM23 signal that will be passed to the deadtime logic upon the occurrence of a FORCE_OUT event in that submodule. 00b - Generated SM3PWM23 signal is used by the deadtime logic. 01b - Inverted generated SM3PWM23 signal is used by the deadtime logic. 10b - SWCOUT[SM3OUT23] is used by the deadtime logic. 11b - PWM3_EXTA signal is used by the deadtime logic.
13-12 SM3SEL45	Submodule 3 PWM45 Control Select This field selects possible over-rides to the generated SM3PWM45 signal that will be passed to the deadtime logic upon the occurrence of a FORCE_OUT event in that submodule. 00b - Generated SM3PWM45 signal is used by the deadtime logic. 01b - Inverted generated SM3PWM45 signal is used by the deadtime logic. 10b - SWCOUT[SM3OUT45] is used by the deadtime logic. 11b - PWM3_EXTB signal is used by the deadtime logic.
11-10 SM2SEL23	Submodule 2 PWM23 Control Select This field selects possible over-rides to the generated SM2PWM23 signal that will be passed to the deadtime logic upon the occurrence of a FORCE_OUT event in that submodule. 00b - Generated SM2PWM23 signal is used by the deadtime logic. 01b - Inverted generated SM2PWM23 signal is used by the deadtime logic. 10b - SWCOUT[SM2OUT23] is used by the deadtime logic. 11b - PWM2_EXTA signal is used by the deadtime logic.
9-8 SM2SEL45	Submodule 2 PWM45 Control Select This field selects possible over-rides to the generated SM2PWM45 signal that will be passed to the deadtime logic upon the occurrence of a FORCE_OUT event in that sudeadtime logic upon the occurrence of a FORCE_OUT event in that submodule. 00b - Generated SM2PWM45 signal is used by the deadtime logic. 01b - Inverted generated SM2PWM45 signal is used by the deadtime logic. 10b - SWCOUT[SM2OUT45] is used by the deadtime logic. 11b - PWM2_EXTB signal is used by the deadtime logic.
7-6 SM1SEL23	Submodule 1 PWM23 Control Select This field selects possible over-rides to the generated SM1PWM23 signal that will be passed to the deadtime logic upon the occurrence of a FORCE_OUT event in that submodule.

Table continues on the next page...

Field	Function
	00b - Generated SM1PWM23 signal is used by the deadtime logic. 01b - Inverted generated SM1PWM23 signal is used by the deadtime logic. 10b - SWCOUT[SM1OUT23] is used by the deadtime logic. 11b - PWM1_EXTA signal is used by the deadtime logic.
5-4 SM1SEL45	Submodule 1 PWM45 Control Select This field selects possible over-rides to the generated SM1PWM45 signal that will be passed to the deadtime logic upon the occurrence of a FORCE_OUT event in that submodule. 00b - Generated SM1PWM45 signal is used by the deadtime logic. 01b - Inverted generated SM1PWM45 signal is used by the deadtime logic. 10b - SWCOUT[SM1OUT45] is used by the deadtime logic. 11b - PWM1_EXTB signal is used by the deadtime logic.
3-2 SM0SEL23	Submodule 0 PWM23 Control Select This field selects possible over-rides to the generated SM0PWM23 signal that will be passed to the deadtime logic upon the occurrence of a FORCE_OUT event in that submodule. 00b - Generated SM0PWM23 signal is used by the deadtime logic. 01b - Inverted generated SM0PWM23 signal is used by the deadtime logic. 10b - SWCOUT[SM0OUT23] is used by the deadtime logic. 11b - PWM0_EXTA signal is used by the deadtime logic.
1-0 SM0SEL45	Submodule 0 PWM45 Control Select This field selects possible over-rides to the generated SM0PWM45 signal that will be passed to the deadtime logic upon the occurrence of a FORCE_OUT event in that submodule. 00b - Generated SM0PWM45 signal is used by the deadtime logic. 01b - Inverted generated SM0PWM45 signal is used by the deadtime logic. 10b - SWCOUT[SM0OUT45] is used by the deadtime logic. 11b - PWM0_EXTB signal is used by the deadtime logic.

55.8.48 Master Control Register (MCTRL)

55.8.48.1 Offset

Register	Offset
MCTRL	188h

55.8.48.2 Function

In every 4-bit field in this register, each bit acts on a separate submodule. Accordingly, the description of every bitfield refers to the effect of an individual bit.

55.8.48.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	IPOL				RUN				0				LDOK			
W									CLDOK							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

55.8.48.4 Fields

Field	Function
15-12 IPOL	Current Polarity The four buffered read/write bits of this field correspond to submodules 3-0, respectively. Each bit selects between PWM23 and PWM45 as the source for the generation of the complementary PWM pair output for the corresponding submodule. MCTRL[IPOL] is ignored in independent mode. MCTRL[IPOL] does not take effect until a FORCE_OUT event takes place in the appropriate submodule. Reading MCTRL[IPOL] reads the buffered value and not necessarily the value currently in effect. 0000b - PWM23 is used to generate complementary PWM pair in the corresponding submodule. 0001b - PWM45 is used to generate complementary PWM pair in the corresponding submodule.
11-8 RUN	Run The four read/write bits of this field enable the clocks to the PWM generator of submodules 3-0, respectively. The corresponding MCTRL[RUN] bit must be set for each submodule that is using its input capture functions or is using the local reload as its reload source. When this bit equals zero, the submodule counter is reset and PWM outputs are held. A reset clears this field. 0000b - PWM counter is stopped, but PWM outputs will hold the current state. 0001b - PWM counter is started in the corresponding submodule.
7-4 CLDOK	Clear Load Okay The 4 bits of CLDOK field correspond to submodules 3-0, respectively. Each write-only bit is used to clear the corresponding bit of MCTRL[LDOK]. Write a 1 to CLDOK to clear the corresponding MCTRL[LDOK] bit. If a reload occurs within a submodule with the corresponding MCTRL[LDOK] bit set at the same time that MCTRL[CLDOK] is written, then the reload in that submodule will not be performed and MCTRL[LDOK] will be cleared. CLDOK bit is self-clearing and always reads as a 0.
3-0 LDOK	Load Okay The 4 bits of LDOK field correspond to submodules 3-0, respectively. Each read/set bit loads CTRL[PRSC] and the INIT, FRACVALx, and VALx registers of the corresponding submodule into a set of buffers. The buffered prescaler divisor, submodule counter modulus value, and PWM pulse width take effect at the next PWM reload if CTRL[LDMOD] is clear or immediately if CTRL[LDMOD] is set. Set the corresponding MCTRL[LDOK] bit by reading it when it is logic zero and then writing a logic one to it. The VALx, FRACVALx, INIT, and CTRL[PRSC] registers of the corresponding submodule cannot be written while the corresponding MCTRL[LDOK] bit is set. In Master Reload Mode (CTRL2[RELOAD_SEL]=1), it is only necessary to set the LDOK bit corresponding to submodule0; however, it is recommended to also set the LDOK bit of the slave submodules, to prevent unwanted writes to the registers in the slave submodules.

Field	Function
	The MCTRL[LDOK] bit is automatically cleared after the new values are loaded, or it can be manually cleared before a reload by writing a logic 1 to the appropriate MCTRL[CLDOK] bit. LDOK bits cannot be written with a zero. MCTRL[LDOK] can be set in DMA mode when the DMA indicates that it has completed the update of all CTRL[PRSC], INIT, FRACVALx, and VALx registers in the corresponding submodule. Reset clears LDOK field. 0000b - Do not load new values. 0001b - Load prescaler, modulus, and PWM values of the corresponding submodule.

55.8.49 Master Control 2 Register (MCTRL2)

55.8.49.1 Offset

Register	Offset
MCTRL2	18Ah

55.8.49.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0												0		MONPLL	
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

55.8.49.3 Fields

Field	Function
15-4 —	RESERVED
3-2 —	RESERVED
1-0 MONPLL	Monitor PLL State These bits are used to control disabling of the fractional delay block when the chip PLL is unlocked and/or missing its input reference. The fractional delay block requires a continuous 200 MHz clock from the PLL. If this clock turns off when the fractional delay block is being used, then the output of the fractional delay block can be stuck high or low even if the PLL restarts. When this control bit is set, PLL problems cause

Field	Function
	the fractional delay block to be disabled until the PLL returns to a locked state. Once the PLL is receiving a proper reference and is locked, the fractional delay block requires a 25 µs startup time just as if the FRCTRL[FRAC*_EN] bits had been turned off and turned on again. If PLL monitoring is disabled, then software should manually clear and then set the FRCTRL[FRAC*_EN] bits when the PLL loses its reference or loses lock. This will cause the fractional delay block to be disabled and restarted. If the fractional delay block is not being used, then the value of these bits do not matter. 00b - Not locked. Do not monitor PLL operation. Resetting of the fractional delay block in case of PLL losing lock will be controlled by software. 01b - Not locked. Monitor PLL operation to automatically disable the fractional delay block when the PLL encounters problems. 10b - Locked. Do not monitor PLL operation. Resetting of the fractional delay block in case of PLL losing lock will be controlled by software. These bits are write protected until the next reset. 11b - Locked. Monitor PLL operation to automatically disable the fractional delay block when the PLL encounters problems. These bits are write protected until the next reset.

55.8.50 Fault Control Register (FCTRL0)

55.8.50.1 Offset

Register	Offset
FCTRL0	18Ch

55.8.50.2 Function

For every 4-bit field in this register, the bits act on the fault inputs in order. For example, FLVL bits 15-12 act on faults 3-0, respectively.

55.8.50.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	FLVL				FAUTO				FSAFE				FIE			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

55.8.50.4 Fields

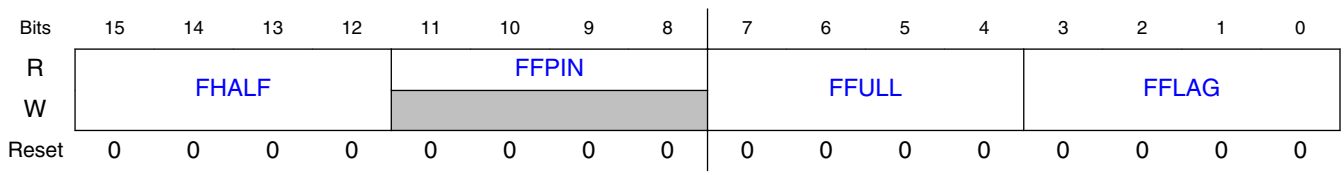
Field	Function
15-12 FLVL	Fault Level The four read/write bits of this field select the active logic level of the individual fault inputs 3-0, respectively. A reset clears this field. 0000b - A logic 0 on the fault input indicates a fault condition. 0001b - A logic 1 on the fault input indicates a fault condition.
11-8 FAUTO	Automatic Fault Clearing The four read/write bits of this field select automatic or manual clearing of faults 3-0, respectively. A reset clears this field. 0000b - Manual fault clearing. PWM outputs disabled by this fault are not enabled until FSTS[FFLAGx] is clear at the start of a half cycle or full cycle depending the states of FSTS[FHALF] and FSTS[FFULL]. If neither FFULL nor FHALF is set, then the fault condition cannot be cleared. This is further controlled by FCTRL[FSAFE]. 0001b - Automatic fault clearing. PWM outputs disabled by this fault are enabled when FSTS[FFPINx] is clear at the start of a half cycle or full cycle depending on the states of FSTS[FHALF] and FSTS[FFULL] without regard to the state of FSTS[FFLAGx]. If neither FFULL nor FHALF is set, then the fault condition cannot be cleared.
7-4 FSAFE	Fault Safety Mode These read/write bits select the safety mode during manual fault clearing. A reset clears this field. FSTS[FFPINx] may indicate a fault condition still exists even though the actual fault signal at the FAULTx pin is clear due to the fault filter latency. 0000b - Normal mode. PWM outputs disabled by this fault are not enabled until FSTS[FFLAGx] is clear at the start of a half cycle or full cycle depending on the states of FSTS[FHALF] and FSTS[FFULL] without regard to the state of FSTS[FFPINx]. If neither FHALF nor FFULL is set then the fault condition cannot be cleared. The PWM outputs disabled by this fault input will not be re-enabled until the actual FAULTx input signal de-asserts since the fault input will combinationally disable the PWM outputs (as programmed in DISMAPn). 0001b - Safe mode. PWM outputs disabled by this fault are not enabled until FSTS[FFLAGx] is clear and FSTS[FFPINx] is clear at the start of a half cycle or full cycle depending on the states of FSTS[FHALF] and FSTS[FFULL]. If neither FHLAF nor FFULL is set, then the fault condition cannot be cleared.
3-0 FIE	Fault Interrupt Enables This read/write field enables CPU interrupt requests generated by the FAULTx pins. A reset clears this field. **NOTE:** The fault protection circuit is independent of the FIEx bit and is always active. If a fault is detected, the PWM outputs are disabled according to the disable mapping register. 0000b - FAULTx CPU interrupt requests disabled. 0001b - FAULTx CPU interrupt requests enabled.

55.8.51 Fault Status Register (FSTS0)

55.8.51.1 Offset

Register	Offset
FSTS0	18Eh

55.8.51.2 Diagram



55.8.51.3 Fields

Field	Function
15-12 FHALF	Half Cycle Fault Recovery These read/write bits are used to control the timing for re-enabling the PWM outputs after a fault condition. These bits apply to both automatic and manual clearing of a fault condition. **NOTE:** Both FHALF and FFULL can be set so that the fault recovery occurs at the start of a full cycle and at the start of a half cycle (as defined by VAL0). If neither FHALF nor FFULL is set, then no fault recovery is possible. 0000b - PWM outputs are not re-enabled at the start of a half cycle. 0001b - PWM outputs are re-enabled at the start of a half cycle (as defined by VAL0).
11-8 FFPIN	Filtered Fault Pins These read-only bits reflect the current state of the filtered FAULTx pins converted to high polarity. A logic 1 indicates a fault condition exists on the filtered FAULTx pin. A reset has no effect on this field.
7-4 FFULL	Full Cycle These read/write bits are used to control the timing for re-enabling the PWM outputs after a fault condition. These bits apply to both automatic and manual clearing of a fault condition. **NOTE:** Both FHALF and FFULL can be set so that the fault recovery occurs at the start of a full cycle and at the start of a half cycle (as defined by VAL0). If neither FHALF nor FFULL is set, then no fault recovery is possible. 0000b - PWM outputs are not re-enabled at the start of a full cycle 0001b - PWM outputs are re-enabled at the start of a full cycle
3-0 FFLAG	Fault Flags These read-only flag is set within two CPU cycles after a transition to active on the FAULTx pin. Clear this bit by writing a logic one to it. A reset clears this field. While the reset value is 0, these bits may be set to 1 by the time they can be read depending on the state of the fault input signals. 0000b - No fault on the FAULTx pin. 0001b - Fault on the FAULTx pin.

55.8.52 Fault Filter Register (FFILT0)

55.8.52.1 Offset

Register	Offset
FFILT0	190h

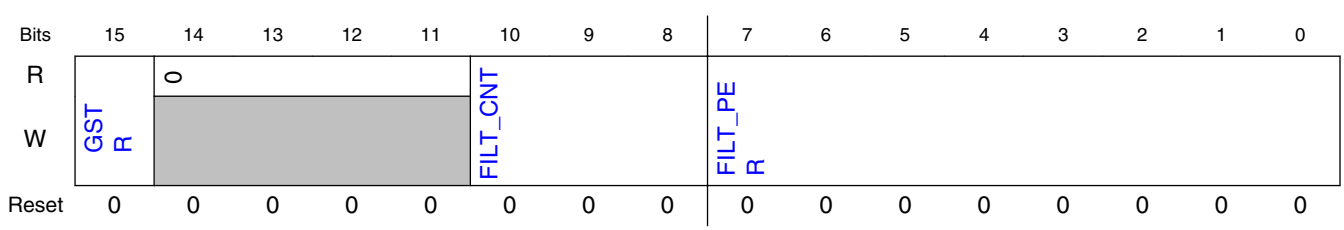
55.8.52.2 Function

The settings in this register are shared among each of the fault input filters within the fault channel.

Input filter considerations include:

- The FILT_PER value should be set such that the sampling period is larger than the period of the expected noise. This way a noise spike will only corrupt one sample. The FILT_CNT value should be chosen to reduce the probability of noisy samples causing an incorrect transition to be recognized. The probability of an incorrect transition is defined as the probability of an incorrect sample raised to the FILT_CNT+3 power.
- The values of FILT_PER and FILT_CNT must also be traded off against the desire for minimal latency in recognizing input transitions. Turning on the input filter (setting FILT_PER to a non-zero value) introduces a latency of ((FILT_CNT+4) x FILT_PER x IPBus clock period). Note that even when the filter is enabled, there is a combinational path to disable the PWM outputs. This is to ensure rapid response to fault conditions and also to ensure fault response if the PWM module loses its clock. The latency induced by the filter will be seen in the time to set FSTS[FFLAG] and FSTS[FFPIN].

55.8.52.3 Diagram



55.8.52.4 Fields

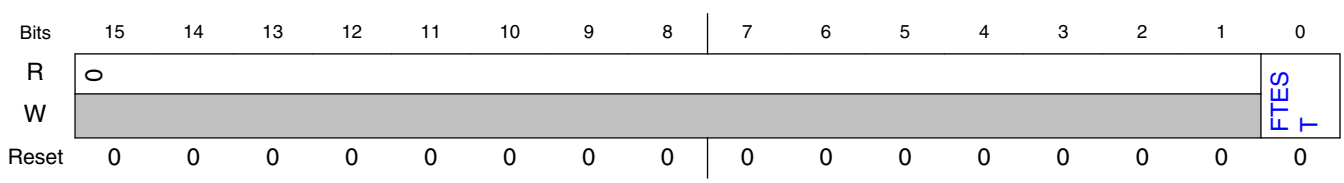
Field	Function
15 GSTR	Fault Glitch Stretch Enable This bit is used to enable the fault glitch stretching logic. This logic ensures that narrow fault glitches are stretched to be at least 2 IPBus clock cycles wide. In some cases a narrow fault input can cause problems due to the short PWM output shutdown/re-activation time. The stretching logic ensures that a glitch on the fault input, when the fault filter is disabled, will be registered in the fault flags. 0b - Fault input glitch stretching is disabled. 1b - Input fault signals will be stretched to at least 2 IPBus clock cycles.
14-11 —	RESERVED
10-8 FILT_CNT	Fault Filter Count These bits represent the number of consecutive samples that must agree prior to the input filter accepting an input transition. The number of samples is the decimal value of this field plus three: the bitfield value of 0-7 represents 3-10 samples, respectively. The value of FILT_CNT affects the input latency.
7-0 FILT_PER	Fault Filter Period This 8-bit field applies universally to all fault inputs. These bits represent the sampling period (in IPBus clock cycles) of the fault pin input filter. Each input is sampled multiple times at the rate specified by this field. If FILT_PER is 0x00 (default), then the input filter is bypassed. The value of FILT_PER affects the input latency. **NOTE:** When changing values for FILT_PER from one non-zero value to another non-zero value, first write a value of zero to clear the filter.

55.8.53 Fault Test Register (FTST0)

55.8.53.1 Offset

Register	Offset
FTST0	192h

55.8.53.2 Diagram



55.8.53.3 Fields

Field	Function
15-1 —	RESERVED
0 FTEST	Fault Test This read/write bit is used to simulate a fault condition. Setting this bit causes a simulated fault to be sent into all of the fault filters. The condition propagates to the fault flags and possibly the PWM outputs depending on the DISMAPn settings. Clearing this bit removes the simulated fault condition. 0b - No fault 1b - Cause a simulated fault

55.8.54 Fault Control 2 Register (FCTRL20)

55.8.54.1 Offset

Register	Offset
FCTRL20	194h

55.8.54.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0												NOCOMB			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

55.8.54.3 Fields

Field	Function
15-4 —	RESERVED
3-0	No Combinational Path From Fault Input To PWM Output

Field	Function
NOCOMB	This read/write field is used to control the combinational path from the fault inputs to the PWM outputs. When these bits are low (default), the corresponding fault inputs have a combinational path to the PWM outputs that are sensitive to these fault inputs (as defined by DISMAP0 and DISMAP1). This combinational path is a safety feature that ensures the output is disabled even if the SOC has a failure of its clocking system. The combinational path also means that a pulse on the fault input can cause a brief disable of the PWM output even if the fault pulse is not wide enough to get through the input filter and be latched in the fault logic. Setting these bits removes the combinational path and uses the filtered and latched fault signals as the fault source to disable the PWM outputs. This eliminates fault glitches from creating PWM output glitches but also increases the latency to respond to a real fault. 0000b - There is a combinational link from the fault inputs to the PWM outputs. The fault inputs are combined with the filtered and latched fault signals to disable the PWM outputs. 0001b - The direct combinational path from the fault inputs to the PWM outputs is disabled and the filtered and latched fault signals are used to disable the PWM outputs.

Chapter 56
Quadrature Decoder (QDC)

56.1 Chip-specific QDC information

Table 56-1. Reference links to related information

Topic	Related module(s)	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
Interrupts, DMA Events and XBAR Assignments	-	Interrupts, DMA Events and XBAR Assignments

56.2 Overview

The enhanced quadrature decoder module interfaces to position/speed sensors that are used in industrial motor control applications. Using 5 input signals (PHASEA, PHASEB, INDEX, TRIGGER, and HOME) from those position/speed sensors, the quadrature decoder module decodes shaft position, revolution count, and speed.

56.2.1 System Block Diagram

The next figure shows the block diagram of the quad decoder module integrated into an SoC.

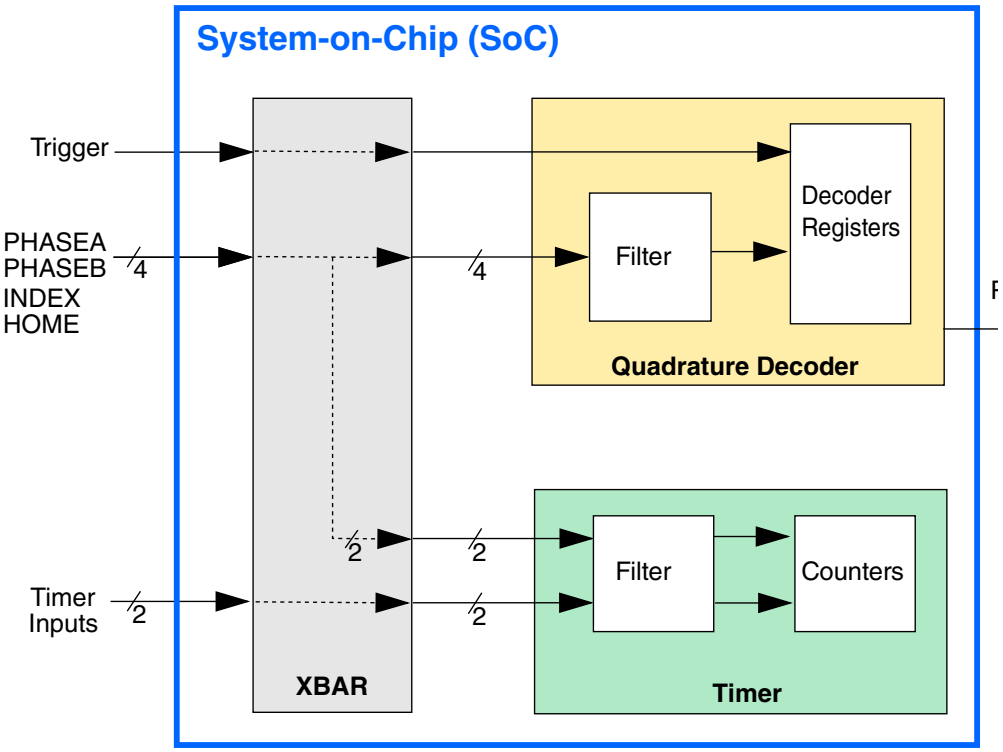


Figure 56-1. System Block Diagram

56.2.2 Features

- Includes logic to decode quadrature signals
- Inputs can be connected to a general purpose timer to make low speed velocity measurements
- Configurable digital filter for inputs
- Quadrature decoder filter can be bypassed
- 32-bit position counter capable of modulo counting
- Position counter can be initialized by software or external events
- 16-bit position difference register
- Compare function can indicate when shaft has reached a defined position
- A watchdog timer can detect a non-rotating shaft condition
- Preloadable 16-bit revolution counter
- Maximum count frequency equals the peripheral clock rate

56.2.3 Decoder Block Diagram

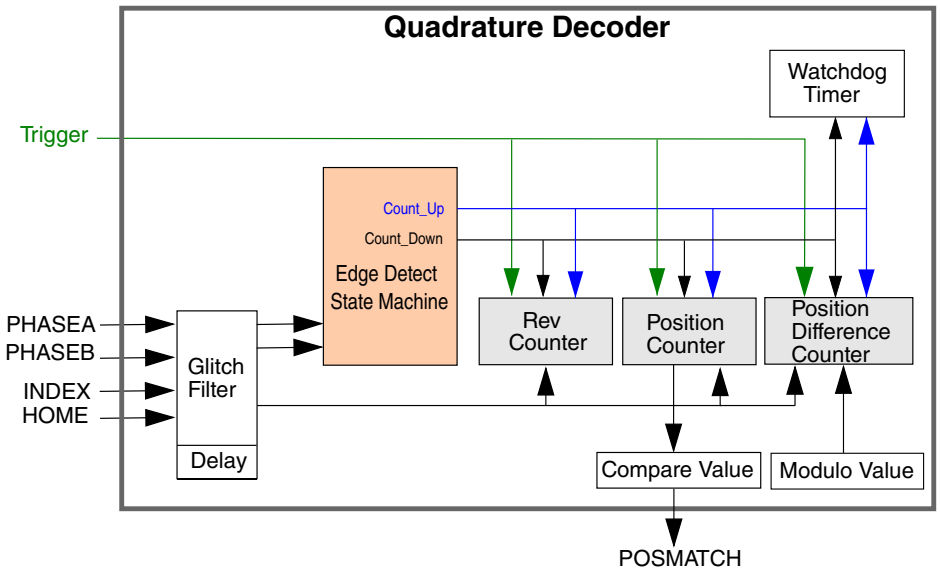


Figure 56-2. Quad Decoder Block Diagram

56.2.3.1 Glitch Filter

Because the quad decoder logic must sense signal transitions, the signal inputs are first run through a glitch filter. This glitch filter has a digital delay line, which samples multiple time points on the signal and verifies a stable new signal state, before outputting this new signal state to the internal quad decoder logic. To adapt to a variety of signal bandwidths, the sample rate of this delay line is programmable.

56.2.3.2 Edge Detect State Machine

The edge detect state machine looks for changes in the 4 possible states of the filtered PHASEA and PHASEB inputs, which are used to calculate the direction of motion. This information is formatted as Count_Up and Count_Down signals. Each counter has its own hold register. These signals are routed into up to 3 up/down counters:

- Position counter
- Revolution counter
- Position difference counter

56.2.3.3 Position Counter

The 32-bit position counter calculates up or down on every count pulse, generated by the difference of PHASEA and PHASEB. The position counter acts as an integration_info, whose count value is proportional to position. The direction of the count is determined by the Count_Up and Count_Down signals. Position counters may be initialized to a predetermined value by one of 3 different methods:

- Software-triggered event
- INDEX signal transition
- HOME signal transition

The INDEX and HOME signals can be programmed to interrupt the processor. Whenever the position counters (UPOS or LPOS) are read, a snapshot of the

- position counter
- position difference counter
- revolution counter

are each placed into their respective hold registers.

56.2.3.4 Position Difference Counter

The 16-bit position difference counter contains the position difference value occurring between each read of the position register. The position register counts up or down on every count pulse. The position difference counter acts as a differentiator, whose count value is proportional to the change in position since the last time the position counter was read.

When the position registers, the position difference counter, or the revolution counter is read, the position difference counter's contents are copied into the position difference hold register (POSDH) and the position difference counter is cleared.

56.2.3.5 Position Difference Counter Hold

The Position Difference Counter Hold register stores a copy of the position difference counter at the time that the position register was read. When the position register, the position difference counter, or the revolution counter is read, the position difference counter's contents are copied into the position difference hold register (POSDH), and the position difference counter is cleared.

56.2.3.6 Revolution Counter

The 16-bit up/down revolution counter is intended to count or integrate revolutions by counting index pulses. The direction of the count is determined by the Count_Up and Count_Down signals, determined by the Phase A and B inputs. A different count direction on the rising and falling edges of the index pulse indicates that the quad encoder changed direction on the index pulse.

56.2.3.7 Watchdog Timer

The watchdog timer ensures that the algorithm is indicating motion in the shaft; 2 successive counts indicate proper operation and will reset the timer.

- The timeout value is programmable.
- When a timeout occurs, an interrupt to the processor can be generated.

56.2.3.8 Pulse Accumulator Functionality

The logic can be programmed to integrate only selected transitions of the PHASEA signal. In this way, the position counter can be used as a pulse accumulator.

- The count direction is up.
- The pulse accumulator can also optionally be initialized by the INDEX input.

56.3 Functional Description

The next timing diagram shows the basic operation of a quadrature incremental position quad decoder.

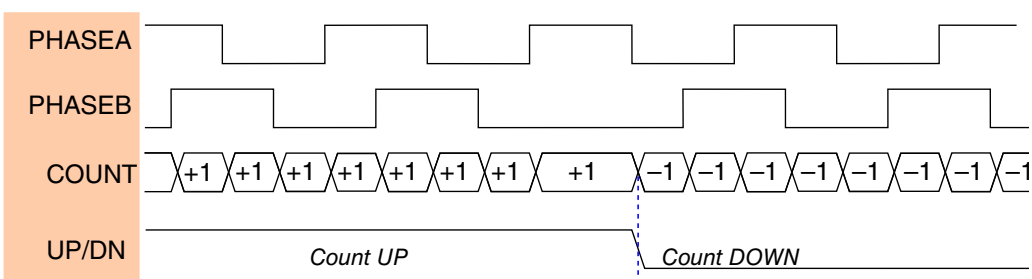


Figure 56-3. Quad Decoder Signals

56.3.1 Positive versus Negative Direction

A typical quad encoder has 3 outputs: PHASEA signal, PHASEB signal, INDEX pulse (not shown).

- If PHASEA leads PHASEB, then motion is in the positive direction.
- If PHASEA trails PHASEB, then motion is in the negative direction.

Transitions on these phases can be integrated to yield position or differentiated to yield velocity. The quad decoder is designed to perform these functions in hardware.

56.3.2 Prescaler for Slow or Fast Speed Measurement

- For applications with a fast moving shaft encoder, the speed can be computed by calculating the change in the position counter per unit time, or by reading the position difference counter register (POSD) and calculating speed.
- For applications with slow motor speeds and low line count quad encoders, the timer module enables high resolution velocity measurements by measuring the time period between quad phases.
  - The timer module uses a 16-bit free running counter operated from a prescaled version of the IPBus clock.
  - The prescaler divides the IPBus clock by values ranging from 1 to 128. A 60 MHz IPBus clock frequency would yield a resolution of from 17 ns to 2.1 µs and a maximum count period of from 1.09 ms to 140 ms. For example, with a 1000-tooth decoder, speeds could be calculated down to 0.11 rpm using a prescaler.

56.3.3 Holding Registers and Initializing Registers

Hold registers are associated with 3 types of counters:

- Position
- Position difference
- Revolution

When any of the counter registers are read, the contents of each counter register is written to the corresponding hold register. Taking a snapshot of the counters' values allows a consistent view of a system's position and the velocity to be attained. To capture a time stamp of when these registers are read, use the POSMATCH output with a timer channel.

The position counter is 32 bits wide. To ensure that the position counter can be reliably initialized with two 16-bit accesses, two registers (an upper and a lower initialization register) are provided. The upper initialization register (UINIT) and lower initialization register (LINIT) should be loaded with the desired value. Next, the position counter can

be loaded by writing 1 to the Software-Triggered Initialization of Position Counters UPOS and LPOS bit (CTRL[SWIP]). Alternatively, either the INDEX-Triggered Initialization of Position Counters UPOS and LPOS bit (CTRL[XIP]) or the Enable HOME to Initialize Position Counters UPOS and LPOS bit (CTRL[HIP]) can enable the position counter to be initialized in response to a HOME or INDEX signal transition.

56.3.4 Speed Measurement Method

This section explains speed measurement method. It includes speed measurement algorithm.

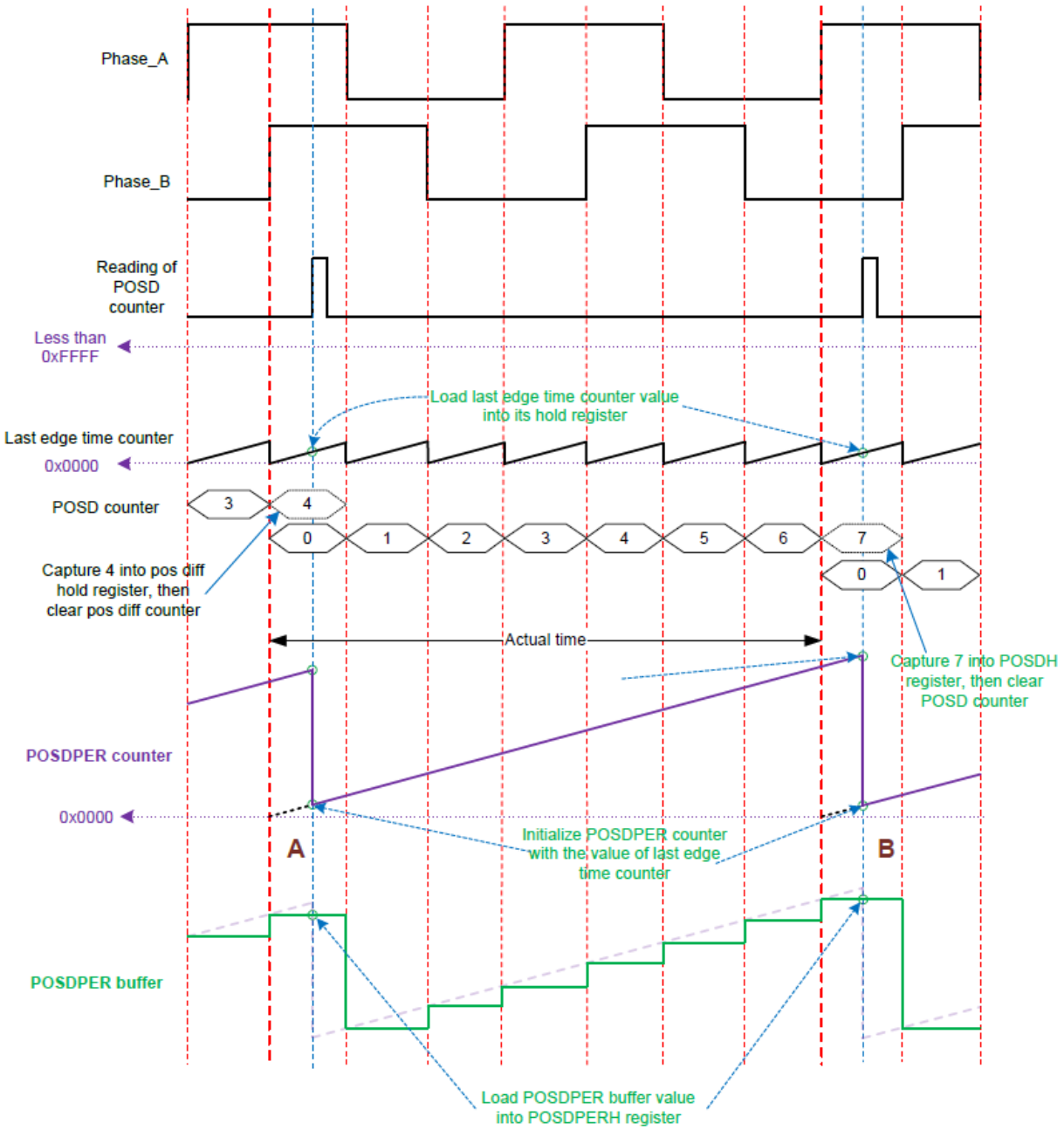


Figure 56-4. Counters behavior and updating mechanism

A reading of POSD occurs at time point A and it also occurs at time point B. “Actual time” represents the time duration between the first phase_a/b edges right before time point A and B. At time point B, after reading POSD register, POSDPERH contains the time length of “Actual time” and POSDH contains the value of phase_a/b pulses within that “Actual time”, speed can be calculated just based on POSDPERH and POSDH. A visualized explanation of speed measurement with this method is shown in figure below.

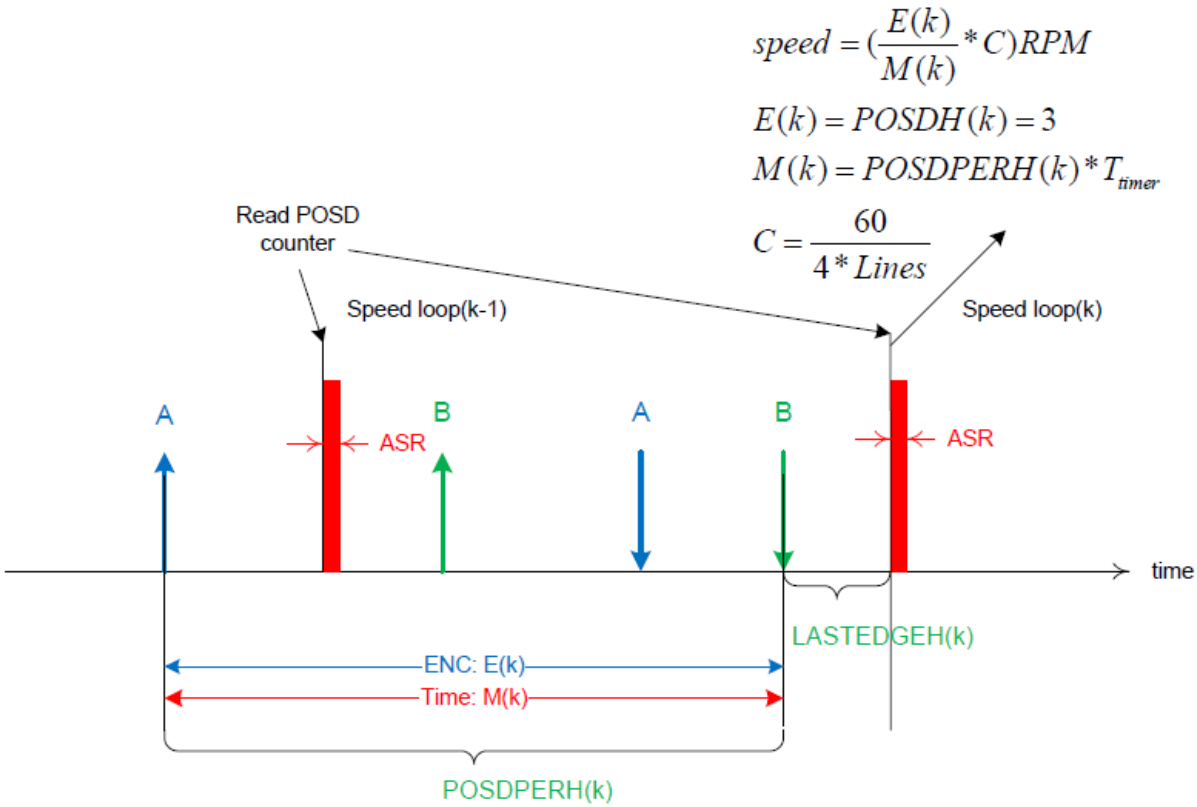


Figure 56-5. High speed measurement with method

56.3.4.1 Speed calculation algorithm

However, in order to cover all the cases in real applications, a simple speed measurement algorithm is necessary. Below is the algorithm.

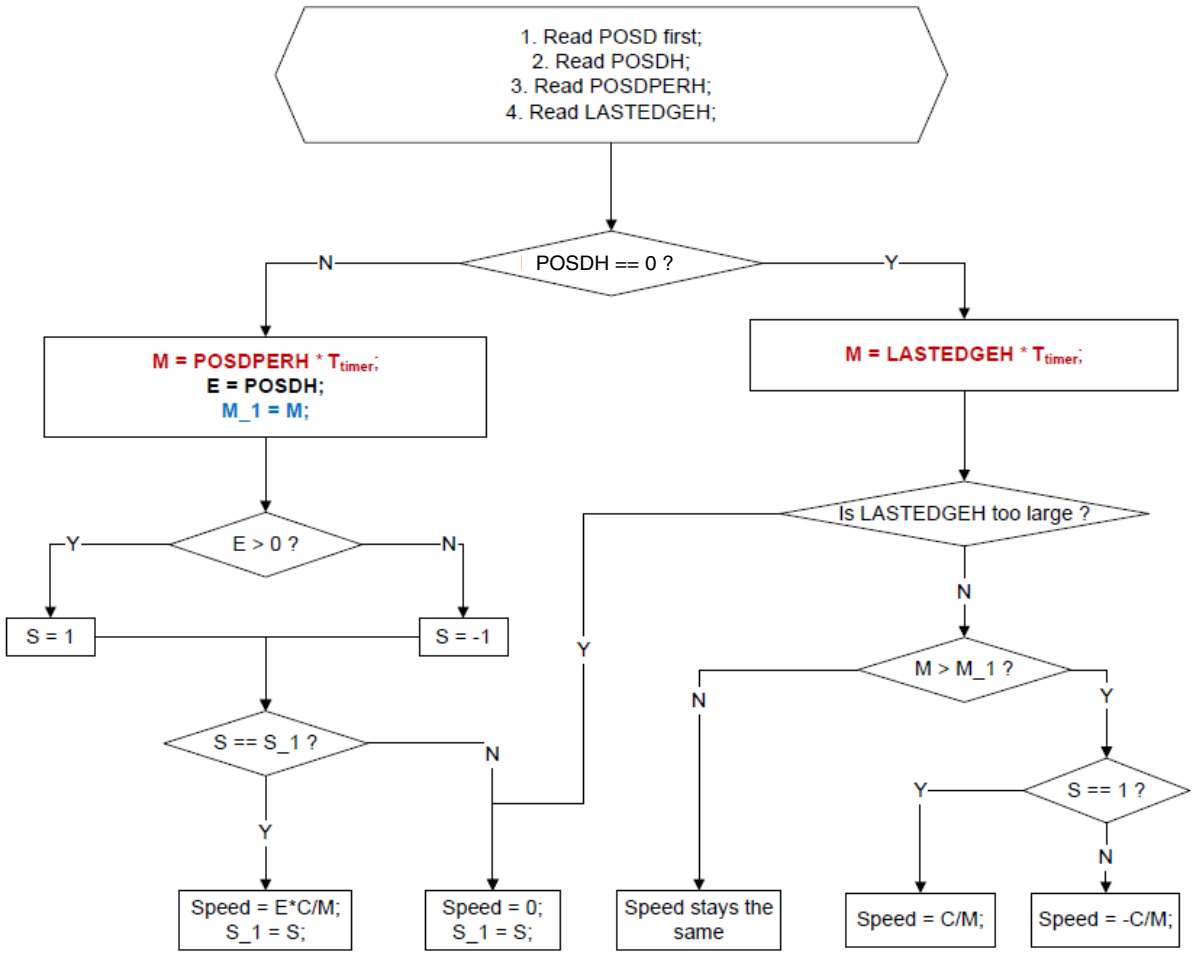


Figure 56-6. Flowchart of speed calculation with method

56.3.4.2 Speed calculation of motor's starting from standstill (Chip is out of reset)

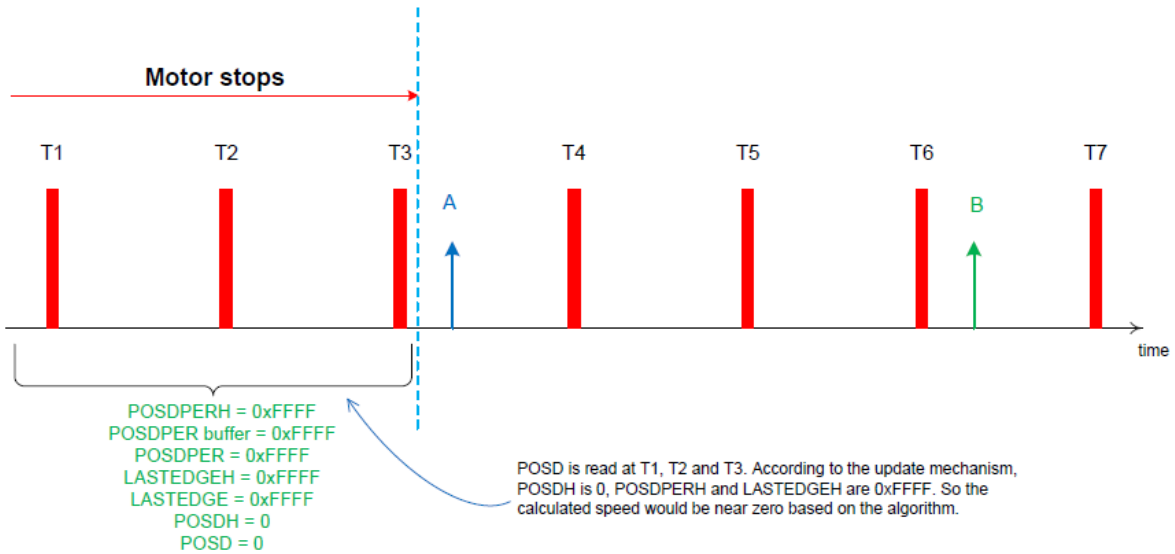


Figure 56-7. Speed measurement at time point T1~T3

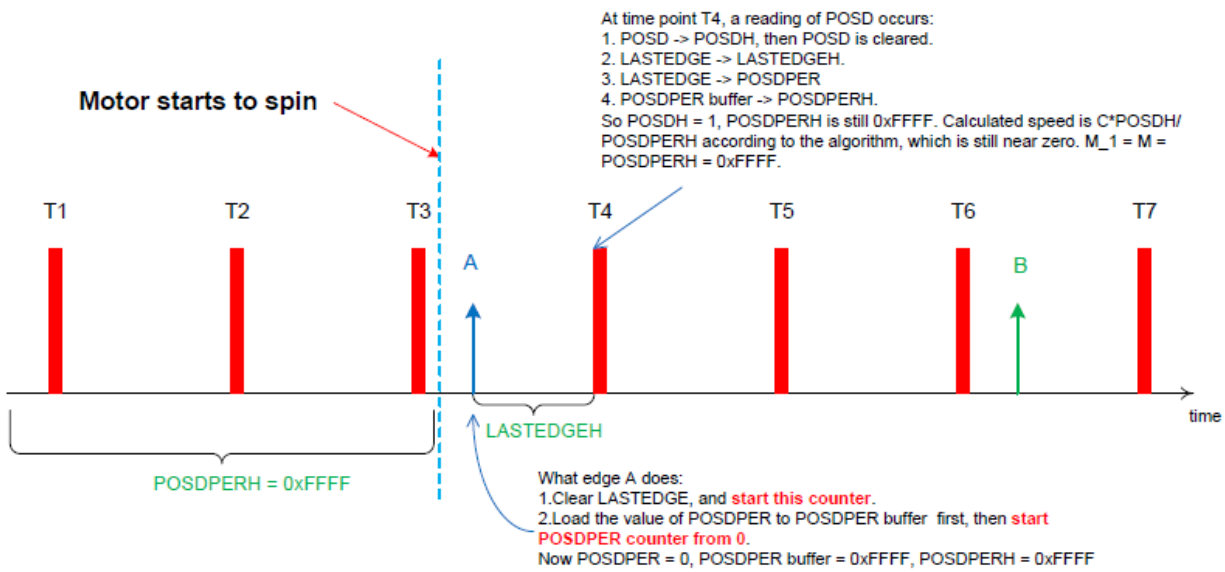


Figure 56-8. Speed measurement at time point T4

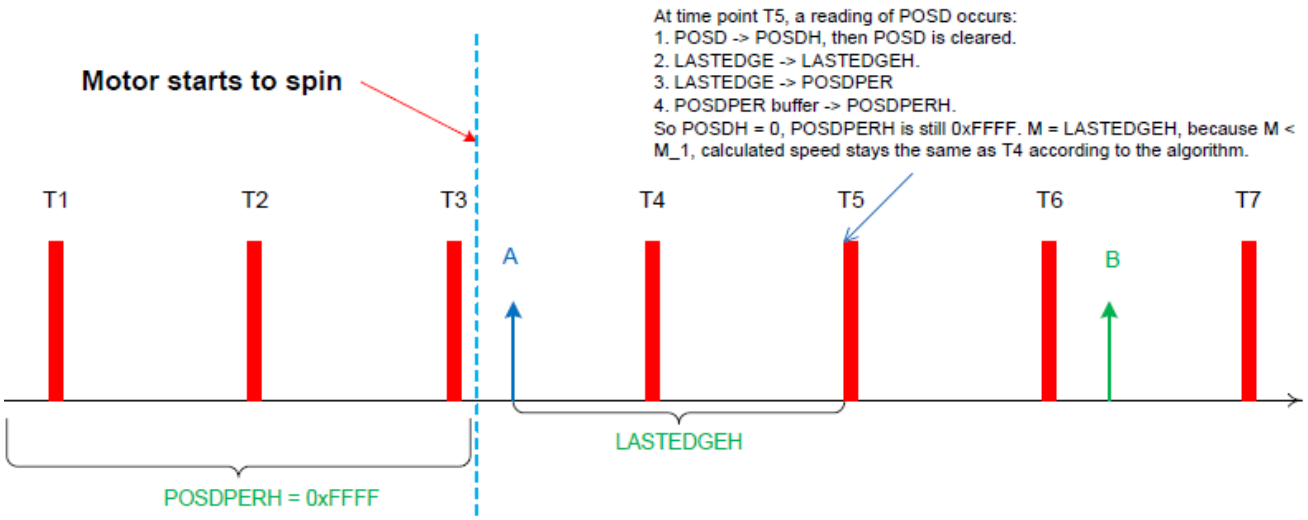


Figure 56-9. Speed measurement at time point T5

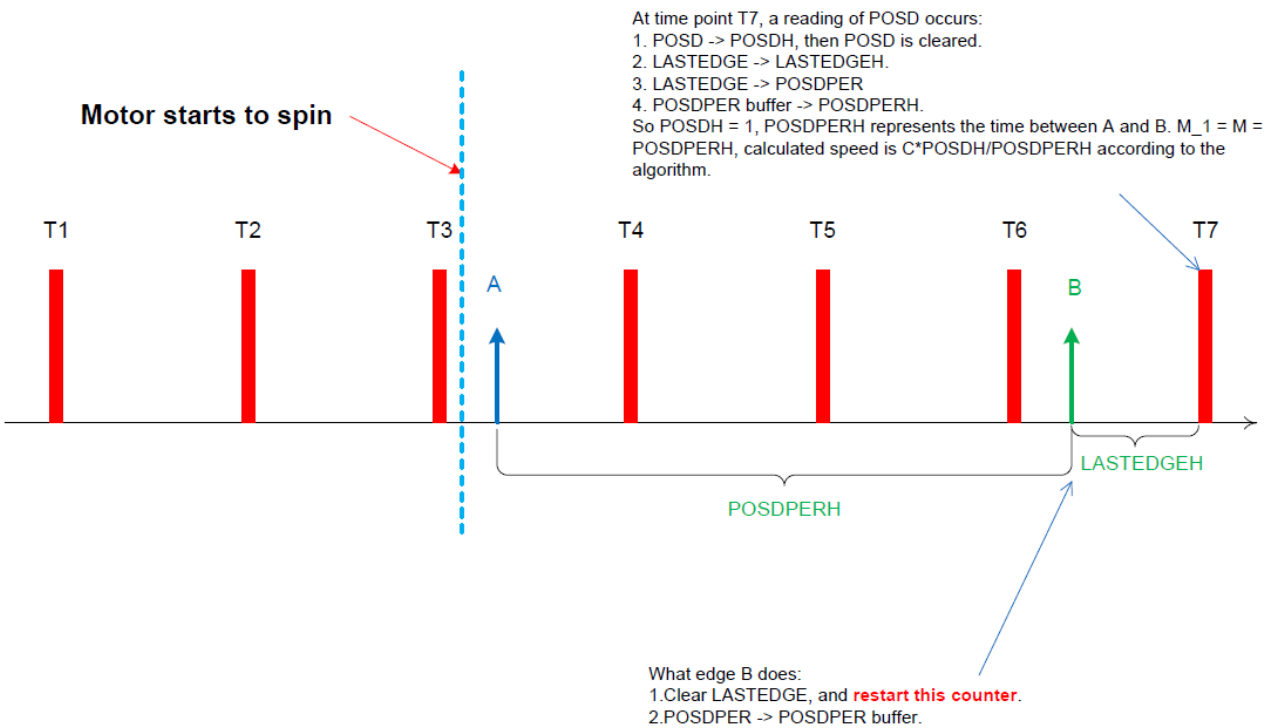


Figure 56-10. Speed measurement at time point T7

56.3.4.3 Speed calculation when motor stops

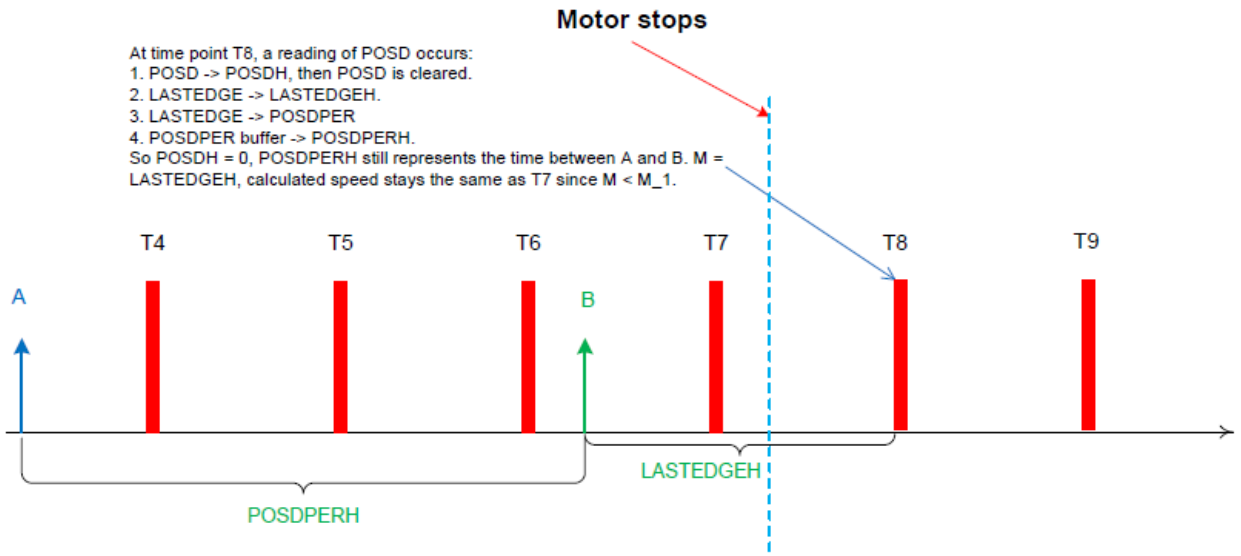


Figure 56-11. Speed measurement at time point T8

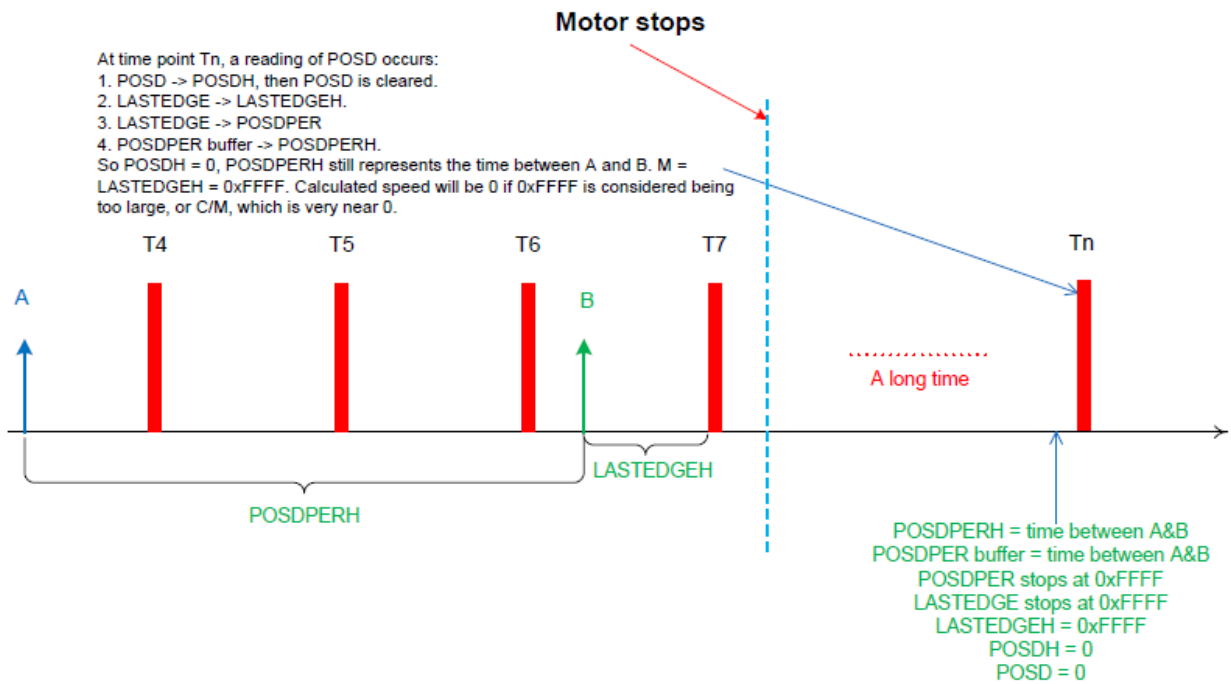


Figure 56-12. Speed measurement at time point Tn

56.3.4.4 Speed calculation when motor starts again

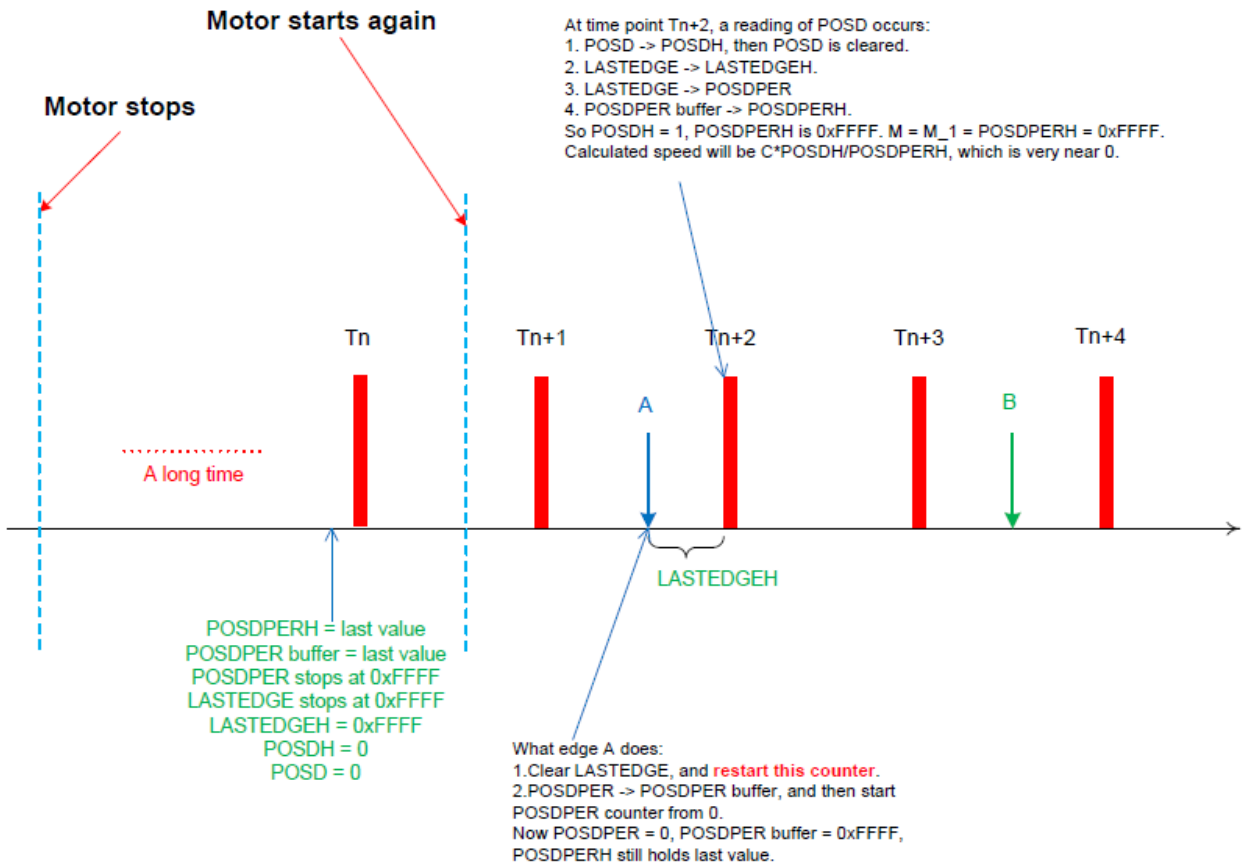


Figure 56-13. Speed measurement at time point Tn+2

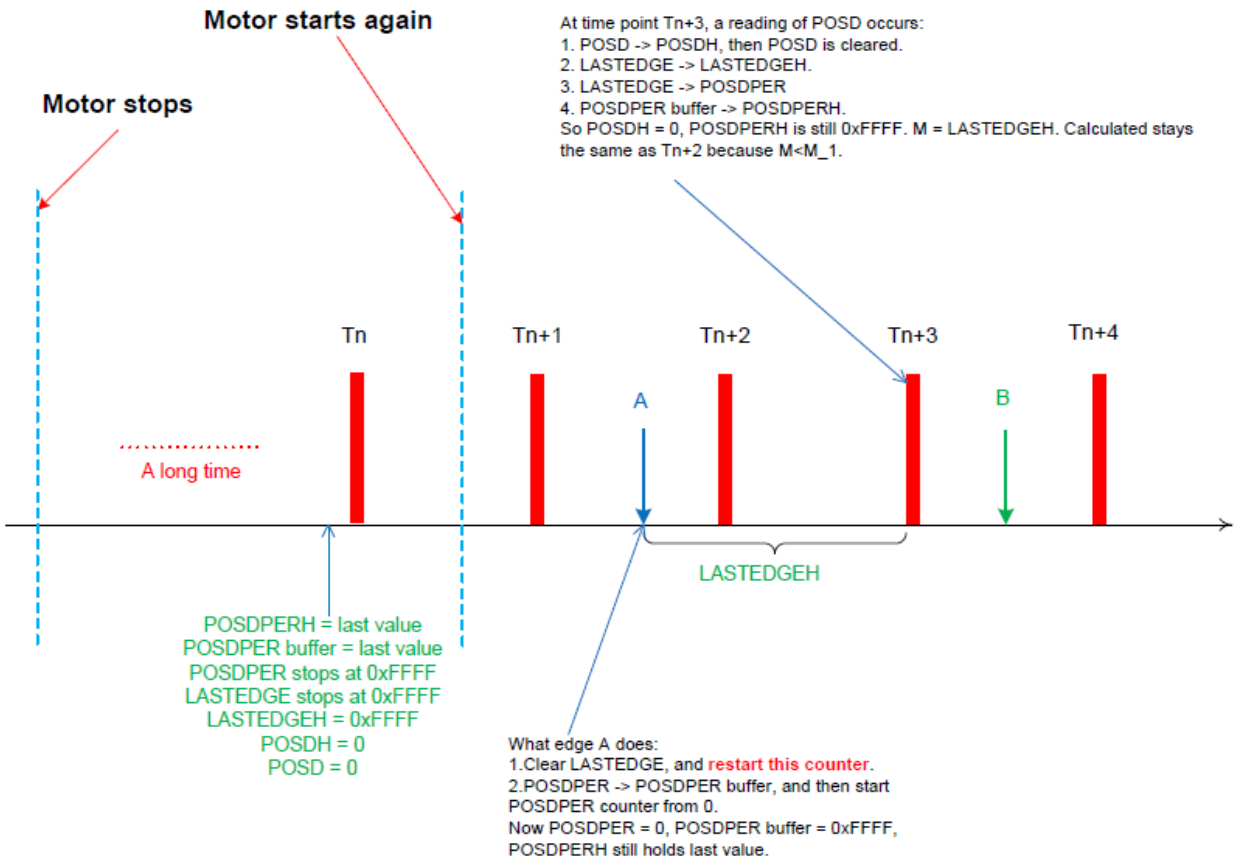


Figure 56-14. Speed measurement at time point Tn+3

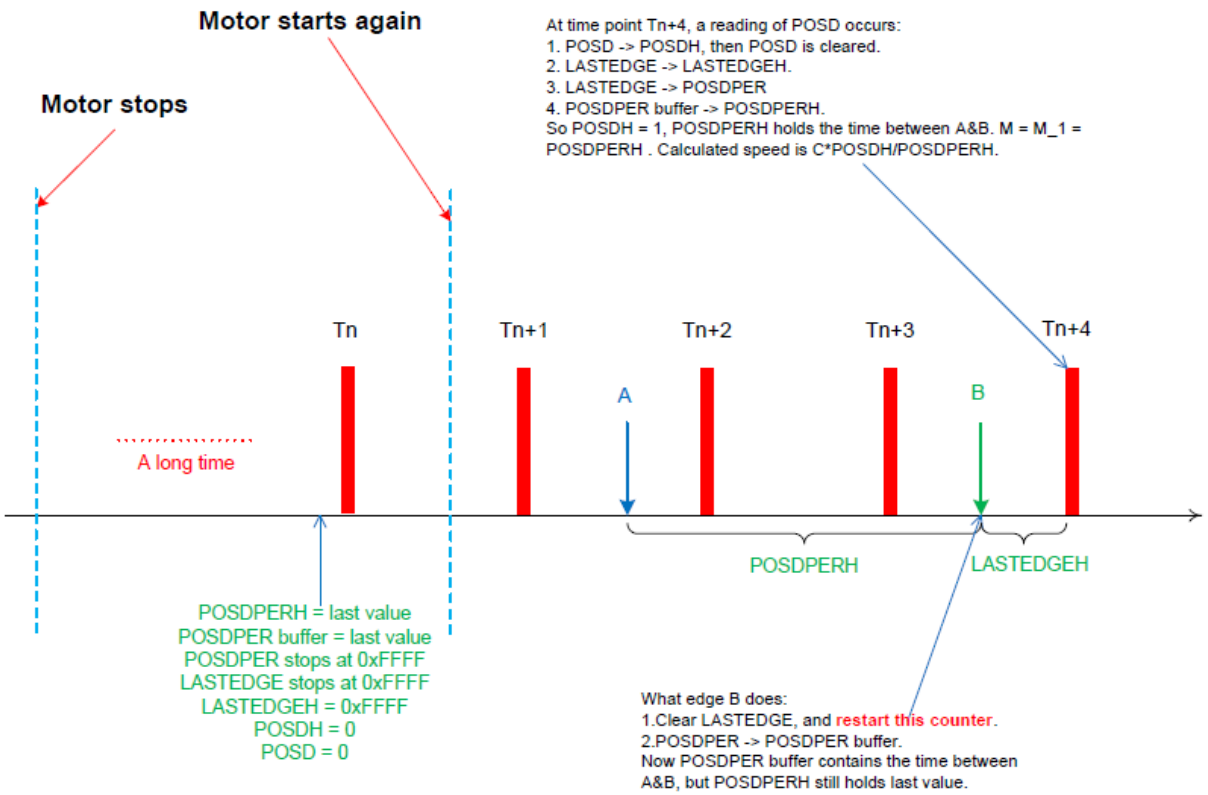


Figure 56-15. Speed measurement at time point Tn+4

56.3.5 Clocks

The IPBus clock is the only clock required by this module in normal operation.

56.3.6 Resets

There are no special requirements. This module is reset by any system reset.

56.3.7 Interrupts

The following table lists the module interrupts.

Table 56-2. Interrupt Summary

Core Interrupt	Interrupt Flag	Interrupt Enable	Description
ipi_int_home	CTRL[HIRQ]	CTRL[HIE]	HOME signal transition interrupt

Table continues on the next page...

Table 56-2. Interrupt Summary (continued)

Core Interrupt	Interrupt Flag	Interrupt Enable	Description
ipi_int_index	CTRL[XIRQ]	CTRL[XIE]	INDEX signal transition interrupt or roll-over/ under interrupt
	CTRL2[ROIRQ]	CTRL2[ROIE]	
	CTRL2[RUIRQ]	CTRL2[RUIE]	
ipi_int_wdog	CTRL[DIRQ]	CTRL[DIE]	Watchdog timeout interrupt
ipi_int_comp	CTRL[CMPIRQ]	CTRL[CMPIE]	Compare match interrupt

56.4 Signal Descriptions

Four external signals are multiplexed to the four pins (PHASEA, PHASEB, INDEX, HOME) of a quad timer module. Two internal signals (TRIGGER, POSMATCH) can be connected to other SoC resources.

Table 56-3. Signals

Signal		Description
PHASEA	Phase A Input	The PHASEA input can be connected to one of the phases from a two-phase shaft quad encoder output. PHASEA and PHASEB are used by the quad decoder module to indicate a decoder increment has passed, and to calculate its direction. • When the quad decoder module is used as a single phase pulse accumulator, PHASEA can also be used as the single input. • The PHASEA input can be an input capture channel for one of the timer modules (in the SoC), which can be connected to using a crossbar switch. Direction for two-phase shaft quad encoder output: • PHASEA is the leading phase for a shaft rotating in the positive direction. • PHASEA is the trailing phase for a shaft rotating in the negative direction.
PHASEB	Phase B Input	The PHASEB input can be connected to the other phase from a two-phase shaft quad encoder output. • The PHASEB input can be an input capture channel for one of the timer modules (in the SoC), which can be connected to using a crossbar switch. Direction for two-phase shaft quad encoder output: • PHASEB is the trailing phase for a shaft rotating in the positive direction. • PHASEB is the leading phase for a shaft rotating in the negative direction.
INDEX	Index Input	• Normally connected to the index pulse output of a quad encoder, the INDEX pulse can optionally reset the position counter and the pulse accumulator of the quad decoder module. The INDEX pulse also causes a change of state on the revolution counter. The direction of this state change (increment or decrement) is calculated from the PHASEA and PHASEB inputs. • The INDEX input can be an input capture channel for one of the timer modules (in the SoC), which can be connected to using a crossbar switch.
HOME	Home Switch Input	The HOME input can be used by the quad decoder and the timer module. • The HOME input can be used to trigger the initialization of the position counters (UPOS and LPOS). Often the HOME signal is connected to a

Table continues on the next page...

Table 56-3. Signals (continued)

Signal		Description
		sensor on the motor or machine, sending notification that it has reached a defined home position. • The HOME signal can be connected to the timer module (in the SoC) using a crossbar switch.
TRIGGER	Trigger Input	The TRIGGER input can be used • to clear the position counters (UPOS and LPOS) • to take a snapshot of the POS, REV, and POSD registers • to indicate an elapsed time period, by connecting the TRIGGER input signal to a periodic pulse generator or timer
POSMATCH	Position Match Output	• To record the time at which the position of the shaft matches a user-defined compare value (COMP), the POSMATCH output can be used to trigger a timer channel. • Alternatively, to record the time at which the position information was read, (when the position counters (UPOS and LPOS), revolution counter (REV), or position difference counter (POSD) registers are read) the POSMATCH output can be used to trigger a timer channel.

56.5 Memory Map and Registers

This section describes the module memory map and registers.

56.5.1 QDC register descriptions

The address of a register is the sum of a base address and an address offset.

- Base address is defined at the MCU level
- Address offset is defined at the module level

For the base address, see the specific chip documentation. All memory locations base and offsets are given in hex.

56.5.1.1 QDC memory map

QDC1 base address: 403C_8000h

QDC2 base address: 403C_C000h

QDC3 base address: 403D_0000h

QDC4 base address: 403D_4000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Control Register (CTRL)	16	RW	0000h
2h	Input Filter Register (FILT)	16	RW	0000h
4h	Watchdog Timeout Register (WTR)	16	RW	0000h
6h	Position Difference Counter Register (POSD)	16	RW	0000h
8h	Position Difference Hold Register (POSDH)	16	RO	0000h
Ah	Revolution Counter Register (REV)	16	RW	0000h
Ch	Revolution Hold Register (REVH)	16	RO	0000h
Eh	Upper Position Counter Register (UPOS)	16	RW	0000h
10h	Lower Position Counter Register (LPOS)	16	RW	0000h
12h	Upper Position Hold Register (UPOSH)	16	RO	0000h
14h	Lower Position Hold Register (LPOSH)	16	RO	0000h
16h	Upper Initialization Register (UINIT)	16	RW	0000h
18h	Lower Initialization Register (LINIT)	16	RW	0000h
1Ah	Input Monitor Register (IMR)	16	RO	0000h
1Ch	Test Register (TST)	16	RW	0000h
1Eh	Control 2 Register (CTRL2)	16	RW	0000h
20h	Upper Modulus Register (UMOD)	16	RW	0000h
22h	Lower Modulus Register (LMOD)	16	RW	0000h
24h	Upper Position Compare Register (UCOMP)	16	RW	FFFFh
26h	Lower Position Compare Register (LCOMP)	16	RW	FFFFh

56.5.1.2 Control Register (CTRL)

56.5.1.2.1 Offset

Register	Offset
CTRL	0h

56.5.1.2.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	HIRQ	HIE	HIP	HNE	0	REV	PH1	XIRQ	XIE	XIP	XNE	DIRQ	DIE	WDE	CMPIRQ	CMPIE
W	W1C				SWIP			W1C				W1C			W1C	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

56.5.1.2.3 Fields

Field	Function
15 HIRQ	HOME Signal Transition Interrupt Request HOME Signal Transition Interrupt Request is set when a transition on the HOME signal occurs, according to the Use Negative Edge of HOME Input bit (CTRL[HNE]). If HOME Signal Transition Interrupt Request bit is set and HOME Interrupt Enable bit (CTRL[HIE]) is set, then a HOME interrupt occurs. • HOME Signal Transition Interrupt Request bit will remain set until it is cleared by software • Write a one to this bit (HIRQ) to clear it 0b - No transition on the HOME signal has occurred 1b - A transition on the HOME signal has occurred
14 HIE	HOME Interrupt Enable Enables/disables HOME signal interrupts. 0b - Disabled 1b - Enabled
13 HIP	Enable HOME to Initialize Position Counters UPOS and LPOS Enables the position counter to be initialized by the HOME signal. 0b - No action 1b - HOME signal initializes the position counter
12 HNE	Use Negative Edge of HOME Input Selects using the positive/negative edge of the HOME input. 0b - Use positive-going edge-to-trigger initialization of position counters UPOS and LPOS 1b - Use negative-going edge-to-trigger initialization of position counters UPOS and LPOS
11 SWIP	Software-Triggered Initialization of Position Counters UPOS and LPOS Writing 1 to Software-Triggered Initialization of Position Counters bit will transfer the UINIT and LINIT contents to UPOS and LPOS (thereby initializing those counters). • Software-Triggered Initialization of Position Counters bit is always read as 0. 0b - No action 1b - Initialize position counter (using upper and lower initialization registers, UINIT and LINIT)
10 REV	Enable Reverse Direction Counting Selects the direction of the count (how the quadrature signal is interpreted). 0b - Count normally 1b - Count in the reverse direction
9 PH1	Enable Signal Phase Count Mode Bypass/use the quadrature decoder logic. 0b - Use the standard quadrature decoder, where PHASEA and PHASEB represent a two-phase quadrature signal. 1b - Bypass the quadrature decoder. A positive transition of the PHASEA input generates a count signal. The PHASEB input and the REV bit control the counter direction: If CTRL[REV] = 0, PHASEB = 0, then count up If CTRL[REV] = 1, PHASEB = 1, then count up If CTRL[REV] = 0, PHASEB = 1, then count down If CTRL[REV] = 1, PHASEB = 0, then count down
8 XIRQ	INDEX Pulse Interrupt Request INDEX Pulse Interrupt Request is set when a transition on the INDEX signal occurs, according to the Use Negative Edge of INDEX Pulse bit (CTRL[XNE]). If INDEX Pulse Interrupt Request bit is set and INDEX Pulse Interrupt Enable bit (CTRL[XIE]) is set, then an INDEX interrupt occurs.

Table continues on the next page...

Field	Function
	• INDEX Pulse Interrupt Request will remain set until cleared by software • Write a one to this bit (XIRQ) to clear it 0b - INDEX pulse has not occurred 1b - INDEX pulse has occurred
7 XIE	INDEX Pulse Interrupt Enable Enables/disables INDEX pulse interrupts. 0b - Disabled 1b - Enabled
6 XIP	INDEX Triggered Initialization of Position Counters UPOS and LPOS Enables/disables the position counter to be initialized by the INDEX pulse. 0b - INDEX pulse does not initialize the position counter 1b - INDEX pulse initializes the position counter
5 XNE	Use Negative Edge of INDEX Pulse Selects the positive/negative edge of the INDEX pulse used to initialize the position counter. 0b - Use positive edge of INDEX pulse 1b - Use negative edge of INDEX pulse
4 DIRQ	Watchdog Timeout Interrupt Request Watchdog Timeout Interrupt Request is set when a watchdog timeout interrupt occurs. • Watchdog Timeout Interrupt Request will remain set until cleared by software • Write a one to this bit (DIRQ) to clear it • Watchdog Timeout Interrupt Request bit is also cleared when Watchdog Enable (CTRL[WDE]) is disabled (=0) 0b - No Watchdog timeout interrupt has occurred 1b - Watchdog timeout interrupt has occurred
3 DIE	Watchdog Timeout Interrupt Enable Enables/disables watchdog timeout interrupts. 0b - Disabled 1b - Enabled
2 WDE	Watchdog Enable Enables/disables the watchdog timer that is monitoring the PHASEA and PHASEB inputs for motor movement. 0b - Disabled 1b - Enabled
1 CMPIRQ	Compare Interrupt Request Compare Interrupt Request is set when the counter matches the COMP value. • Compare Interrupt Request remains set until cleared by software • Write 1 to this bit (CMPIRQ) to clear it 0b - No match has occurred (the counter does not match the COMP value) 1b - COMP match has occurred (the counter matches the COMP value)
0 CMPIE	Compare Interrupt Enable Enables/disables compare interrupt. 0b - Disabled 1b - Enabled

56.5.1.3 Input Filter Register (FILT)

56.5.1.3.1 Offset

Register	Offset
FILT	2h

56.5.1.3.2 Function

The Input Filter Register sets the values of the input filter sample period (FILT_PER),the filter prescaler (FILT_PRSC) and the input filter sample count (FILT_CNT).

- The Input Filter prescaler value (FILT_PRSC) should be set to prescale the IPBus clock. (frequency of FILT clock) = (frequency of IPBus clock) / 2^FILT_PRSC
- The Input Filter Sample Period (FILT_PER) should be set so that the sampling period is larger than the period of the expected noise. Doing this means that a noise spike will only corrupt one sample.
- The Input Filter Sample Count (FILT_CNT) should be chosen to reduce the probability of noisy samples causing an incorrect transition to be recognized. The probability of an incorrect transition is defined as the probability of an incorrect sample raised to the power of FILT_CNT+3.

The values of the Input Filter Sample Period (FILT_PER) and the Input Filter Sample Count (FILT_CNT) must also be traded off against the desire for minimal latency in recognizing valid input transitions. Turning on the input filter (setting the Input Filter Sample Period (FILT_PER) to a non-zero value) introduces a latency of ((FILT_CNT +3)*FILT_PER) FILT clock cycles +2 IPBus clock periods.

The filter latency can be measured:

- Drive the quadrature decoder inputs (PHASEA, PHASEB, INDEX, HOME)
- Monitor the filtered output in the Input Monitor Register (IMR)
- Determine how many IPBus clock cycles that it takes before the output shows up, by using the following equations, where f is FILT_PER and s is FILT_CNT.
  - DELAY = f * (s+3)(FILT clock cycles) +1 (IPBus clock cycles)(to read the filtered output)
  - DELAY = f * (s+3)(FILT clock cycles) +2 (IPBus clock cycles) (to monitor the output in the IMR)

One more additional IPBus clock cycle is needed to read the filtered output, and 2 more IPBus clock cycles are needed to monitor the filtered output in the IMR. The sample rate is set when it reaches the number f. The following examples use the preceding equations:

- Example: when f = 0, the filter is bypassed: DELAY = 1 or 2 clock cycles.
- Example: when f = 5 and s = 2: DELAY = 5 * (2+3)FILT clock cycles + (1 or 2)IPBus clock cycles

56.5.1.3.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	FILT_PRSC			0		FILT_CNT			FILT_PER							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

56.5.1.3.4 Fields

Field	Function
15-13 FILT_PRSC	prescaler divide IPbus clock to FILT clk The Prescaler field is used to prescale the IPbus clock to FILT clock. The IPbus clock is prescaled by a value of 2^FILT_PRSC which means that the prescaler logic can divide the clock by a minimum of 1 and a maximum of 128. So FILTER can filt lower freq noise.
12-11 —	Reserved
10-8 FILT_CNT	Input Filter Sample Count The Input Filter Sample Count represents the number of consecutive samples that must agree, before the input filter accepts an input transition. • A value of 0x0 represents 3 samples • A value of 0x7 represents 10 samples The value of the Input Filter Sample Count (FILT_CNT) affects the input latency.
7-0 FILT_PER	Input Filter Sample Period The Input Filter Sample Period represents the sampling period (in IPBus clock cycles) of the decoder input signals. Each input is sampled multiple times at the rate specified by the Input Filter Sample Period. • If FILT_PER is 0x00 (default), then the input filter is bypassed. Bypassing the digital filter enables the position/position difference counters to operate with count rates up to the IPBus frequency. • The value of the Input Filter Sample Period (FILT_PER) affects the input latency. • When changing the Input Filter Sample Period (FILT_PER) from a non-zero value to another non-zero value, write a value of 0 first, to clear the filter.

56.5.1.4 Watchdog Timeout Register (WTR)

56.5.1.4.1 Offset

Register	Offset
WTR	4h

56.5.1.4.2 Function

The Watchdog Timeout Register stores the timeout count for the quadrature decoder module watchdog timer. This quadrature decoder module watchdog timer is separate from any other watchdog timer(s) that may also be in the device.

56.5.1.4.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	WDOG															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

56.5.1.4.4 Fields

Field	Function
15-0 WDOG	WDOG WDOG[15:0] is a binary representation of the number of clock cycles, plus one clock cycle that the watchdog timer counts before timing out (and optionally generating an interrupt).

56.5.1.5 Position Difference Counter Register (POSD)

56.5.1.5.1 Offset

Register	Offset
POSD	6h

56.5.1.5.2 Function

The Position Difference Counter Register contains the position change in value occurring between each read of the Position Register. The value of the Position Difference Counter Register can be used to calculate velocity.

- The 16-bit Position Difference Counter computes up or down on every count pulse.
- This Position Difference Counter acts as a differentiator, whose count value is proportional to the change in position since the last time that the Position Counter was read.
- When the Position Registers (UPOS or LPOS), the Position Difference Counter (POSD), or the Revolution Counter (REV) is read, the Position Difference Counter's contents are copied into the Position Difference Hold Register (POSDH) and the Position Difference Counter is cleared.

56.5.1.5.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	POSD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

56.5.1.5.4 Fields

Field	Function
15-0 POSD	POSD The position change in value between each read of the Position Register.

56.5.1.6 Position Difference Hold Register (POSDH)

56.5.1.6.1 Offset

Register	Offset
POSDH	8h

56.5.1.6.2 Function

The Position Difference Hold Register register contains a snapshot of the value of the Position Difference Counter Register (POSD). The value of the Position Difference Hold Register (POSDH) can be used to calculate velocity.

56.5.1.6.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	POSDH															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

56.5.1.6.4 Fields

Field	Function
15-0 POSDH	POSDH The value of the POSD register

56.5.1.7 Revolution Counter Register (REV)

56.5.1.7.1 Offset

Register	Offset
REV	Ah

56.5.1.7.2 Function

Contains the current value of the Revolution Counter.

56.5.1.7.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	REV															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

56.5.1.7.4 Fields

Field	Function
15-0 REV	REV The current value of the Revolution Counter

56.5.1.8 Revolution Hold Register (REVH)

56.5.1.8.1 Offset

Register	Offset
REVH	Ch

56.5.1.8.2 Function

Contains a snapshot of the value of the Revolution Counter Register (REV).

56.5.1.8.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	REVH															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

56.5.1.8.4 Fields

Field	Function
15-0 REVH	REVH The value of the Revolution Counter Register (REV)

56.5.1.9 Upper Position Counter Register (UPOS)

56.5.1.9.1 Offset

Register	Offset
UPOS	Eh

56.5.1.9.2 Function

Contains the upper (most significant) half of the Position Counter. This is the binary count from the Position Counter.

56.5.1.9.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	POS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

56.5.1.9.4 Fields

Field	Function
15-0 POS	POS The upper (most significant) half of the Position Counter

56.5.1.10 Lower Position Counter Register (LPOS)

56.5.1.10.1 Offset

Register	Offset
LPOS	10h

56.5.1.10.2 Function

Contains the lower (least significant) half of the Position Counter. This is the binary count from the Position Counter.

56.5.1.10.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	POS															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

56.5.1.10.4 Fields

Field	Function
15-0 POS	POS The lower (least significant) half of the Position Counter

56.5.1.11 Upper Position Hold Register (UPOSH)

56.5.1.11.1 Offset

Register	Offset
UPOSH	12h

56.5.1.11.2 Function

Contains a snapshot of the Upper Position Counter Register (UPOS register).

56.5.1.11.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	POSH															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

56.5.1.11.4 Fields

Field	Function
15-0 POSH	POSH The value of the Upper Position Counter Register (UPOS)

56.5.1.12 Lower Position Hold Register (LPOSH)

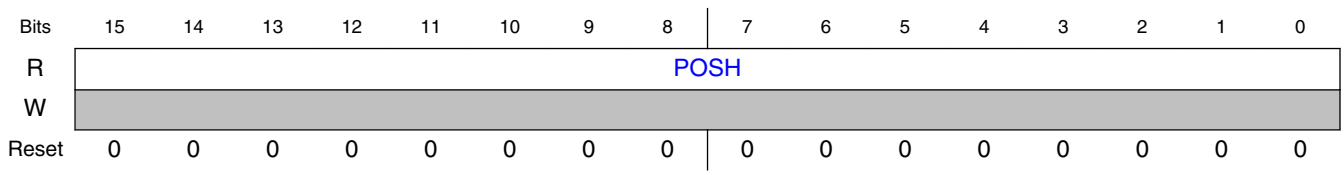
56.5.1.12.1 Offset

Register	Offset
LPOSH	14h

56.5.1.12.2 Function

Contains a snapshot of the Lower Position Counter Register (LPOS).

56.5.1.12.3 Diagram



56.5.1.12.4 Fields

Field	Function
15-0 POSH	POSH The value of the Lower Position Counter Register (LPOS)

56.5.1.13 Upper Initialization Register (UINIT)

56.5.1.13.1 Offset

Register	Offset
UINIT	16h

56.5.1.13.2 Function

Contains the value to be used to initialize the upper half of the position counter (UPOS).

56.5.1.13.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	INIT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

56.5.1.13.4 Fields

Field	Function
15-0 INIT	INIT The value to be used to initialize the upper half of the position counter (UPOS)

56.5.1.14 Lower Initialization Register (LINIT)

56.5.1.14.1 Offset

Register	Offset
LINIT	18h

56.5.1.14.2 Function

Contains the value to be used to initialize the lower half of the position counter (LPOS).

56.5.1.14.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	INIT															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

56.5.1.14.4 Fields

Field	Function
15-0 INIT	INIT The value to be used to initialize the lower half of the position counter (LPOS)

56.5.1.15 Input Monitor Register (IMR)

56.5.1.15.1 Offset

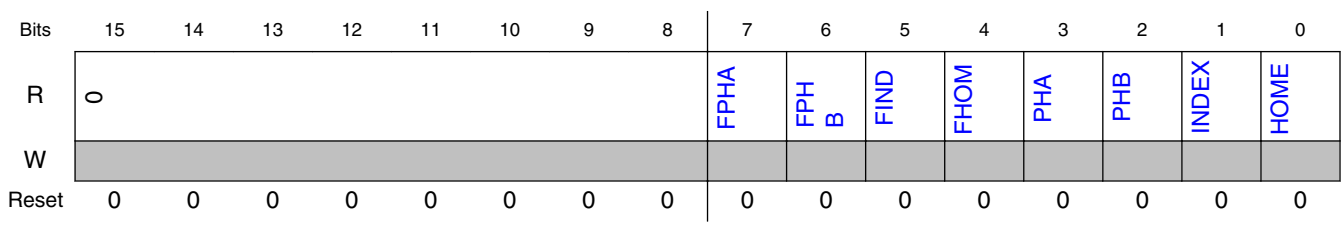
Register	Offset
IMR	1Ah

56.5.1.15.2 Function

The Input Monitor Register contains the values of the raw and filtered PHASEA, PHASEB, INDEX, and HOME input signals. The reset value depends on the values of the raw and filtered values of PHASEA, PHASEB, INDEX, and HOME input signals:

- If these input pins are connected to a pull-up, then bits 0–7 of the IMR are all ones.
- If these input pins are connected to a pull-down device, then bits 0–7 of the Input Monitor Register (IMR) are all zeros.

56.5.1.15.3 Diagram



56.5.1.15.4 Fields

Field	Function
15-8 —	Reserved
7 FPHA	FPHA Filtered version of PHASEA input
6 FPHB	FPHB Filtered version of PHASEB input
5 FIND	FIND Filtered version of INDEX input
4	FHOM

Table continues on the next page...

Field	Function
FHOM	Filtered version of HOME input
3 PHA	PHA Raw PHASEA input
2 PHB	PHB Raw PHASEB input
1 INDEX	INDEX Raw INDEX input
0 HOME	HOME The raw HOME input

56.5.1.16 Test Register (TST)

56.5.1.16.1 Offset

Register	Offset
TST	1Ch

56.5.1.16.2 Function

The Test Register controls and sets the frequency of a quadrature signal generator; it provides a quadrature test signal to the inputs of the quadrature decoder module.

- The TEST_COUNT value is counted down to zero when the test module is enabled (TEN = 1) and the count is enabled (TCE = 1).
- Each count value of one represents a single quadrature cycle interpreted as a count of one by the position counter (UPOS and LPOS) if it is enabled (TEN = 1, TCE = 1).
- Repeated writing of new values to TEST_COUNT can cause an extra phase transition and therefore an extra count by the Position Counter.
- The period field determines the length (in IPBus clock cycles) of each quadrature cycle phase.

The Test Register is a factory test feature; however, it can also be useful in customer software development and testing.

56.5.1.16.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	TEN	TCE	QDN	TEST_PERIOD					TEST_COUNT							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

56.5.1.16.4 Fields

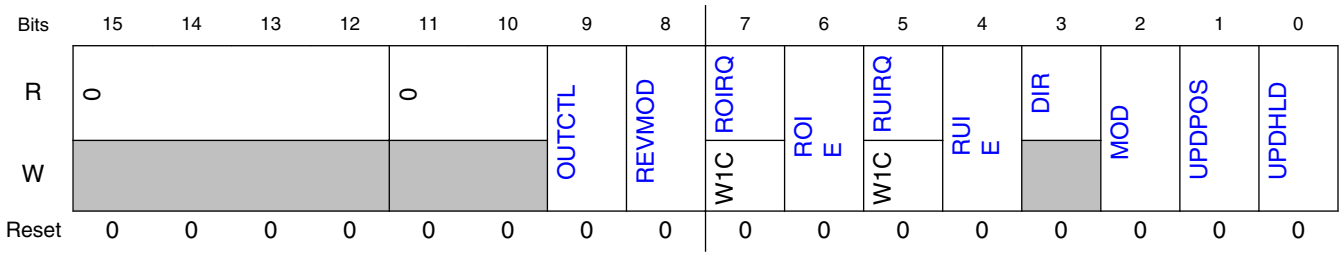
Field	Function
15 TEN	Test Mode Enable Connects the test module to inputs of the quadrature decoder module. 0b - Disabled 1b - Enabled
14 TCE	Test Counter Enable Connects the test counter to inputs of the quadrature decoder module. 0b - Disabled 1b - Enabled
13 QDN	Quadrature Decoder Negative Signal Selects whether a negative or positive Quadrature Decoder signal is generated. 0b - Generates a positive quadrature decoder signal 1b - Generates a negative quadrature decoder signal
12-8 TEST_PERIOD	TEST_PERIOD Period of quadrature phase in IPBus clock cycles
7-0 TEST_COUNT	TEST_COUNT The number of quadrature advances to generate

56.5.1.17 Control 2 Register (CTRL2)

56.5.1.17.1 Offset

Register	Offset
CTRL2	1Eh

56.5.1.17.2 Diagram



56.5.1.17.3 Fields

Field	Function
15-12 —	Reserved
11-10 —	Reserved
9 OUTCTL	Output Control Output Control controls the pulsing of the POSMATCH output signal. This can control when a timer channel captures a timestamp. 0b - POSMATCH pulses when a match occurs between the position counters (POS) and the corresponding compare value (COMP) 1b - POSMATCH pulses when the UPOS, LPOS, REV, or POSD registers are read
8 REVMOD	Revolution Counter Modulus Enable Revolution Counter Modulus Enable selects how the revolution counter (REV) is incremented/ decremented. By default, the revolution counter is controlled based on the count direction and the INDEX pulse. As an option, the revolution counter can be controlled using the roll-over/under detection during modulo counting. 0b - Use INDEX pulse to increment/decrement revolution counter (REV) 1b - Use modulus counting roll-over/under to increment/decrement revolution counter (REV)
7 ROIRQ	Roll-over Interrupt Request Roll-over Interrupt Request bit is set (=1) when the position counter (POS) rolls over from the MOD value to the INIT value, or from 0xFFFF_FFFF to 0x0000_0000. • Roll-over Interrupt Request will remain set until cleared by software • Write a one to this bit (ROIRQ) to clear it 0b - No roll-over has occurred 1b - Roll-over has occurred
6 ROIE	Roll-over Interrupt Enable Roll-over Interrupt Enable read/write bit enables roll-over interrupts, based on Roll-under Interrupt Request (CTRL2[ROIRQ]) being set. This roll-over interrupt is combined with the index interrupt signal. 0b - Disabled 1b - Enabled
5 RUIRQ	Roll-under Interrupt Request Roll-under Interrupt Request bit is set (=1) when the position counter (POS) rolls under from the INIT value to the MOD value, or from 0x0000_0000 to 0xFFFF_FFFF.

Table continues on the next page...

Field	Function
	• Roll-under Interrupt Request will remain set until cleared by software • Write a one to this bit (RUIRQ) to clear it 0b - No roll-under has occurred 1b - Roll-under has occurred
4 RUIE	Roll-under Interrupt Enable Roll-under Interrupt Enable read/write bit enables roll-under interrupts, based on the Roll-over Interrupt Request bit (CTRL2[RUIRQ]) being set. This roll-under interrupt is combined with the index interrupt signal. 0b - Disabled 1b - Enabled
3 DIR	Count Direction Flag The Count Direction Flag indicates the direction of the last count. 0b - Last count was in the down direction 1b - Last count was in the up direction

2
MOD
	Enable Modulo Counting
	• When Enable Modulo Counting is set (=1), it allows the position counters (UPOS and LPOS) to count in a modulo fashion, using MOD and INIT as the upper and lower bounds of the counting range. During modulo counting:
	  • When a "count up" is indicated and the position counter is equal to MOD, then the postion counter will be reloaded with the value of INIT.
	  • When a "count down" is indicated and the position counter is equal to INIT, then the position counter will be reloaded with the value of MOD.
	• When Enable Modulo Counting is clear (=0), then the values of MOD and INIT are ignored, and the position counter wraps to zero when counting up from 0xFFFF_FFFF, or wraps to 0xFFFF_FFFF when counting down from 0.
	
	0b - Disable modulo counting
	1b - Enable modulo counting

1 UPDPOS	Update Position Registers • When Update Position Registers is set (=1), it allows the TRIGGER input to clear the POSD, REV, UPOS and LPOS registers. • When Update Position Registers is clear (=0), the POSD, REV, UPOS and LPOS registers ignore the TRIGGER input. 0b - No action for POSD, REV, UPOS and LPOS registers on rising edge of TRIGGER 1b - Clear POSD, REV, UPOS and LPOS registers on rising edge of TRIGGER
0 UPDHLD	Update Hold Registers • When Update Hold Registers is set (=1), it allows the TRIGGER input to cause an update of the POSDH, REVH, UPOSH, and LPOSH registers • When Update Hold Registers is clear (=0), the hold registers (POSDH, REVH, UPOSH, LPOSH) are not updated by the TRIGGER input Updating the Position Difference Hold Register (POSDH) will also cause the Position Difference Counter Register (POSD) to be cleared. 0b - Disable updates of hold registers on the rising edge of TRIGGER input signal 1b - Enable updates of hold registers on the rising edge of TRIGGER input signal

56.5.1.18 Upper Modulus Register (UMOD)

56.5.1.18.1 Offset

Register	Offset
UMOD	20h

56.5.1.18.2 Function

The Upper Modulus Register contains the upper (most significant) half of the Modulus register. MOD acts as the upper bound during modulo counting, and as the upper reload value when rolling over from the lower bound.

56.5.1.18.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MOD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

56.5.1.18.4 Fields

Field	Function
15-0 MOD	MOD Upper (most significant) half of the Modulus register

56.5.1.19 Lower Modulus Register (LMOD)

56.5.1.19.1 Offset

Register	Offset
LMOD	22h

56.5.1.19.2 Function

The Lower Modulus Register contains the lower (least significant) half of the Modulus register. MOD acts as the upper bound during modulo counting, and as the upper reload value when rolling over from the lower bound.

56.5.1.19.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	MOD															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

56.5.1.19.4 Fields

Field	Function
15-0 MOD	MOD Lower (least significant) half of the Modulus register

56.5.1.20 Upper Position Compare Register (UCOMP)

56.5.1.20.1 Offset

Register	Offset
UCOMP	24h

56.5.1.20.2 Function

The Upper Position Compare Register contains the upper (most significant) half of the Position Compare register. When the value of the Position counter (POS) matches the value of the Position Compare register (COMP), the Compare Interrupt Request flag (CTRL[CMPIRQ]) is set and the POSMATCH output is asserted.

56.5.1.20.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	COMP															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

56.5.1.20.4 Fields

Field	Function
15-0 COMP	COMP Upper (most significant) half of the Position Compare register

56.5.1.21 Lower Position Compare Register (LCOMP)

56.5.1.21.1 Offset

Register	Offset
LCOMP	26h

56.5.1.21.2 Function

The Lower Position Compare Register contains the lower (least significant) half of the Position Compare register. When the value of the Position counter (POS) matches the value of the Position Compare register (COMP), the Compare Interrupt Request flag (CTRL[CMPIRQ]) is set and the POSMATCH output is asserted.

56.5.1.21.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	COMP															
Reset	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

56.5.1.21.4 Fields

Field	Function
15-0 COMP	COMP Lower (least significant) half of the Position Compare register

Chapter 57
Watchdog Timer (WDOG1-2)

57.1 Chip-specific WDOG information

Table 57-1. Reference links to related information

Topic	Related module(s)	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
Interrupts, DMA Events and XBAR Assignments	-	Interrupts, DMA Events and XBAR Assignments

57.2 Overview

The Watchdog Timer (WDOG) protects against system failures by providing a method by which to escape from unexpected events or programming errors.

Once the WDOG is activated, it must be serviced by the software on a periodic basis. If servicing does not take place, the timer times out. Upon timeout, the WDOG1 asserts the internal system reset signal, WDOG_RESET_B_DEB to the System Reset Controller (SRC); and WDOG2 asserts interrupt to SNVS, to report the security violation condition.

There is also a provision for WDOG(ipp_wdog) signal assertion by timeout counter expiration. There is an option of programmable interrupt generation before the counter actually times out. The time at which the interrupt needs to be generated prior to counter

timeout is programmable. There is a power down counter which is enabled out of any reset (POR, Warm/Cold). This counter has a fixed timeout period of 16 seconds, upon which it asserts the WDOG(ipp_wdog) signal.

Flow diagrams for the timeout counter, power down counter and interrupt operations are are shown in Flow Diagrams.

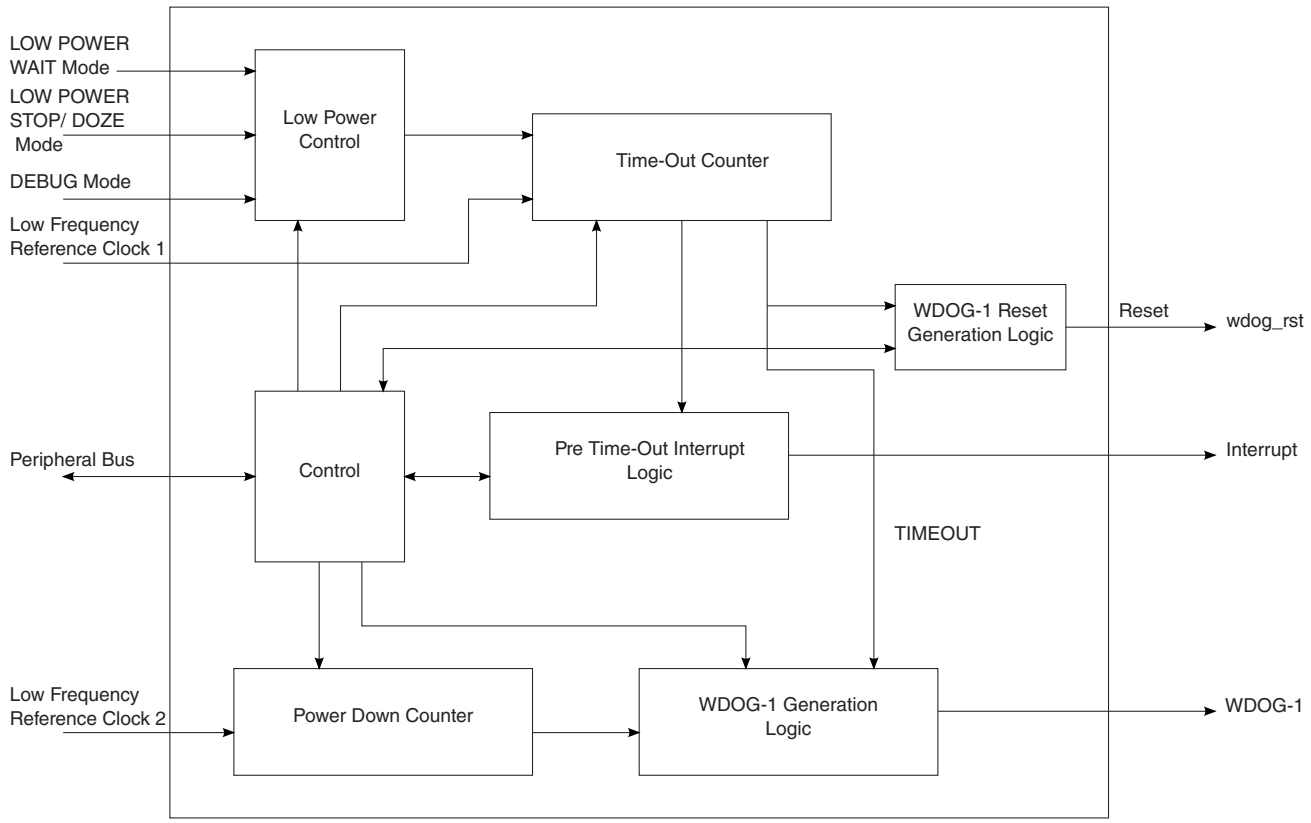


Figure 57-1. WDOG Diagram

NOTE

WDOG-1 means WDOG_B when it presents on a PAD.

57.2.1 Features

The WDOG features are listed below:

- Configurable timeout counter with timeout periods from 0.5 to 128 seconds which, after timeout expiration, result in the assertion of WDOG_RESET_B_DEB reset signal.
- Time resolution of 0.5 seconds
- Configurable timeout counter that can be programmed to run or stop during low-power modes

- Configurable timeout counter that can be programmed to run or stop during DEBUG mode
- Programmable interrupt generation prior to timeout
- The duration between interrupt and timeout events can be programmed from 0 to 127.5 seconds in steps of 0.5 seconds.
- Power down counter with fixed timeout period of 16 seconds, which if not disabled after reset will assert WDOG_B (ipp_wdog) signal low
    - Power down counter will be enabled out of any reset (POR, Warm / Cold reset) by default.

57.3 External signals

Table 57-2. WDOG External Signals

Signal	Description	Direction
WDOG1_ANY	Global WDOG signal. It is AND of WDOG1_B signal and WDOG2_B signal.	O
WDOG1_B	This signal can be routed to external pin of the chip. It is asserted by a software request (set the WDA bit), by power down counter and timeout counter expiration.	O
WDOG1_RST_B_DEB	This signal is a reset source for the chip.	O
WDOG2_B	This signal can be routed to external pin of the chip. It is asserted by a software request (set the WDA bit), by power down counter and timeout counter expiration.	O
WDOG2_RST_B_DEB	This signal asserts interrupt to SNVS to report the security violation condition.	O

See more detailed information in the Pin MUX chapter.

57.4 Clocks

This section describes clocks and special clocking requirements of the block.

The WDOG uses the low frequency reference clock for its counter and control operations. The peripheral bus clock is used for register read/write operations.

The following table describes the clock sources for WDOG. Please see for clock setting, configuration and gating information.

Table 57-3. WDOG Clocks

Clock name	Clock Root	Description
ipg_clk	ipg_clk_root	IP Global functional clock. All functionality inside the WDOG module is synchronized to this clock.
ipg_clk_s	ipg_clk_root	IP slave bus clock. This clock is synchronized to ipg_clk and is only used for register read/write operations.
ipg_clk_32k	ckil_sync_clk_root	Low frequency (32.768 kHz) clock that continues to run in low-power mode. It is assumed that the Clock Controller will provide this clock signal synchronized to ipg_clk in the normal mode, and switch to a non-synchronized signal in low-power mode when the ipg_clk is off.
ipg_async_ckil_clk	anatop_xtal32k_clk	This clock is used by clocking the power down counter.

57.5 Watchdog mechanism and system integration

The WDOG1 module and the WDOG2 (TZ) module are disabled by default (after reset). WDOG1 will be configured during boot while WDOG2 is dedicated for secure world purposes and will be activated by TZ software if required. The TZ watchdog (WDOG2) module protects against TZ starvation by providing a method of escaping normal mode and forcing a switch to the TZ mode. TZ starvation is a situation where the normal OS prevents switching to the TZ mode. Such a situation is undesirable as it can compromise the system's security.

Once the TZ WDOG module is activated, it must be serviced by TZ on a periodic basis. If servicing does not take place, the timer times out. Upon a timeout, the TZ WDOG asserts a TZ-mapped interrupt that forces switching to the TZ mode. If it is still not serviced, the TZ WDOG asserts a security violation signal to the CSU. The TZ WDOG module cannot be programmed or de-activated by normal mode software.

The WDOG modules operate as follows:

- If servicing does not take place, the timer times out and the wdog_rst_b signal is activated (low)
- Interrupt can be generated before the counter actually times out
- The wdog_rst_b signal can be activated by software
- There is a power-down counter which gets enabled out of any reset. This counter has a fixed timeout period of 16 seconds upon which it will assert the ipp_wdog_b signal.

57.6 Functional description

This section provides a complete functional description of the block.

57.6.1 Timeout event

The WDOG provides timeout periods from 0.5 to 128 seconds with a time resolution of 0.5 seconds.

The user can determine the timeout period by writing to the WDOG timeout field (WT[7:0]) in the Watchdog Control (WCR). The WDOG must be enabled by setting the WDE bit of Watchdog Control (WCR) for the timeout counter to start running. After the WDOG is enabled, the counter is activated, loads the timeout value and begins to count down from this programmed value. The timer will time out when the counter reaches zero and the WDOG outputs a system reset signal, WDOG_RESET_B_DEB and asserts WDOG_B (ipp_wdog) (WDT bit should be set in Watchdog Control (WCR)).

However, the timeout condition can be prevented by reloading the counter with the new timeout value (WT[7:0] of WDOG_WCR) if a service routine (see Servicing WDOG to reload the counter) is performed before the counter reaches zero. If any system errors occur which prevent the software from servicing the Watchdog Service (WSR), the timeout condition occurs. By performing the service routine, the WDOG reloads its counter to the timeout value indicated by bits WT[7:0] of the Watchdog Control (WCR) and it restarts the countdown.

A system reset will reset the counter and place it in the idle state at any time during the countdown. The counter flow diagram is shown in Flow Diagrams.

NOTE

The timeout value is reloaded to the counter either at the time WDOG is enabled or after the service routine has been performed.

57.6.1.1 Servicing WDOG to reload the counter

To reload a timeout value to the counter the proper service sequence begins by writing 0x_5555 followed by 0x_AAAA to the Watchdog Service (WSR). Any number of instructions can be executed between the two writes. If the WDOG_WSR is not loaded with 0x_5555 prior to writing 0x_AAAA to the WDOG_WSR, the counter is not reloaded. If any value other than 0x_AAAA is written to the WDOG_WSR after 0x_5555, the counter is not reloaded. This service sequence will reload the counter with the timeout value WT[7:0] of Watchdog Control (WCR). The timeout value can be changed at any point; it is reloaded when WDOG is serviced by the core.

57.6.2 Interrupt event

Prior to timeout, the WDOG can generate an interrupt which can be considered a warning that timeout will occur shortly.

The duration between interrupt event and timeout event can be controlled by writing to the WICT field of Watchdog Interrupt Control (WICR). It can vary between 0 and 127.5 seconds. If the WDOG is serviced (Servicing WDOG to reload the counter) before the interrupt generation, the counter will be reloaded with the timeout value WT[7:0] of Watchdog Control (WCR) and the interrupt will not be triggered.

57.6.3 Power-down counter event

The power-down counter inside WDOG will be enabled out of reset. This counter has a fixed timeout value of 16 seconds, after which it will drive the WDOG_B ($\overline{\text{ipp_wdog}}$) signal low.

To prevent this, the software must disable this counter by clearing the PDE bit of Watchdog Miscellaneous Control (WMCR) within 16 seconds of reset deassertion. Once disabled, this counter can't be enabled again until the next system reset occurs. This feature is intended to prevent the hanging up of cores after reset, as WDOG is not enabled out of reset.

57.6.4 Low power modes

57.6.4.1 STOP and DOZE mode

If the WDOG timer disable bit for low power STOP and DOZE mode (WDZST) bit in the Watchdog Control (WCR), is cleared, the WDOG timer continues to operate using the low frequency reference clock. If the low power enable (WDZST) bit is set, the WDOG timer operation will be suspended in low power STOP or DOZE mode. Upon exiting low power STOP or DOZE mode, the WDOG operation returns to what it was prior to entering the STOP or DOZE mode.

57.6.4.2 WAIT mode

If the WDOG timer disable bit for low power WAIT mode (WDW) bit in the Watchdog Control (WCR), is cleared, the WDOG timer continues to operate using the low frequency reference clock i.e *ipg_clk_32k* clock (32.768 kHz frequency clock). If the low power WAIT enable (WDW) bit is set, the WDOG timer operation will be suspended. Upon exiting low power WAIT mode, the WDOG operation returns to what it was prior to entering the WAIT mode.

NOTE

The WDOG timer won't be able to detect events that happen for periods shorter than one low frequency reference clock cycle. For example, in repeated WAIT mode entry or exit, if the RUN mode time is less than one low frequency reference clock cycle and if the WDW bit is set, the WDOG timer may never time out, even though the system is in RUN mode for a finite duration; WDOG may not see a low frequency reference clock edge during its wake time.

57.6.5 Debug mode

The WDOG timer can be configured for continual operation, or for suspension during debug mode. If the WDOG debug enable (WDBG) bit is set in the Watchdog Control (WCR), the WDOG timer operation is suspended in debug mode. If the WDBG bit is set and the debug mode (*ipg_debug* signal assertion) is entered, WDOG timer operation is suspended after two low frequency reference (*ipg_clk_32k*) clocks. Similarly, WDOG timer operation continues after two low frequency reference clocks of debug mode exit. Register read and write accesses in debug mode continue to function normally. Also, while in debug mode, the WDE bit of Watchdog Control (WCR) can be enabled/disabled directly. If the WDOG debug enable (WDBG) bit is cleared then WDOG timer operation is not suspended. The power-down counter is not affected by debug mode entry/exit.

NOTE

If the WDE bit of Watchdog Control (WCR) is set/cleared while in debug mode, it remains set/cleared even after exiting debug mode.

57.6.6 Operations

57.6.6.1 Watchdog reset generation

The WDOG generated reset signal WDOG_RESET_B_DEB is asserted by the following operations:

- A software write to the Software Reset Signal (SRS) bit of the Watchdog Control (WCR).
- WDOG timeout. See Timeout event.

The $\overline{\text{wdog_rst}}$ will be asserted for one clock cycle of low frequency reference clock for both a timeout condition and a software write occurrence. It remains asserted for 1 clock cycle of low frequency reference clock even if a system reset is asserted in between. Figure 57-3 shows the timing diagram of this signal due to a timeout condition.

57.6.6.2 WDOG_B generation

The WDOG asserts WDOG_B in the following scenarios:

- Software write to WDA bit of Watchdog Control (WCR). WDOG_B signal remains asserted as long as the WDA bit is "0".
- WDOG timeout condition, WDT bit of Watchdog Control (WCR) must be set for this scenario. A description of the timeout condition can be found in the Timeout event. WDOG_B (ipp_wdog) signal remains asserted until a power-on reset (POR) occurs. It gets cleared after the POR occurs (not due to any other system reset). Figure 57-4 shows the timing diagram of WDOG_B (ipp_wdog) due to timeout condition.
- WDOG power-down counter timeout, PDE bit of Watchdog Miscellaneous Control (WMCR) should not be cleared for this scenario. A description of this counter can be found in the Power-down counter event. WDOG_B (ipp_wdog) signal remains asserted for one clock cycle of low frequency reference clock (*ipg_clk_32k*).

Figure 57-2 shows the scenarios under which WDOG_B($\overline{\text{ipp_wdog}}$) gets asserted.

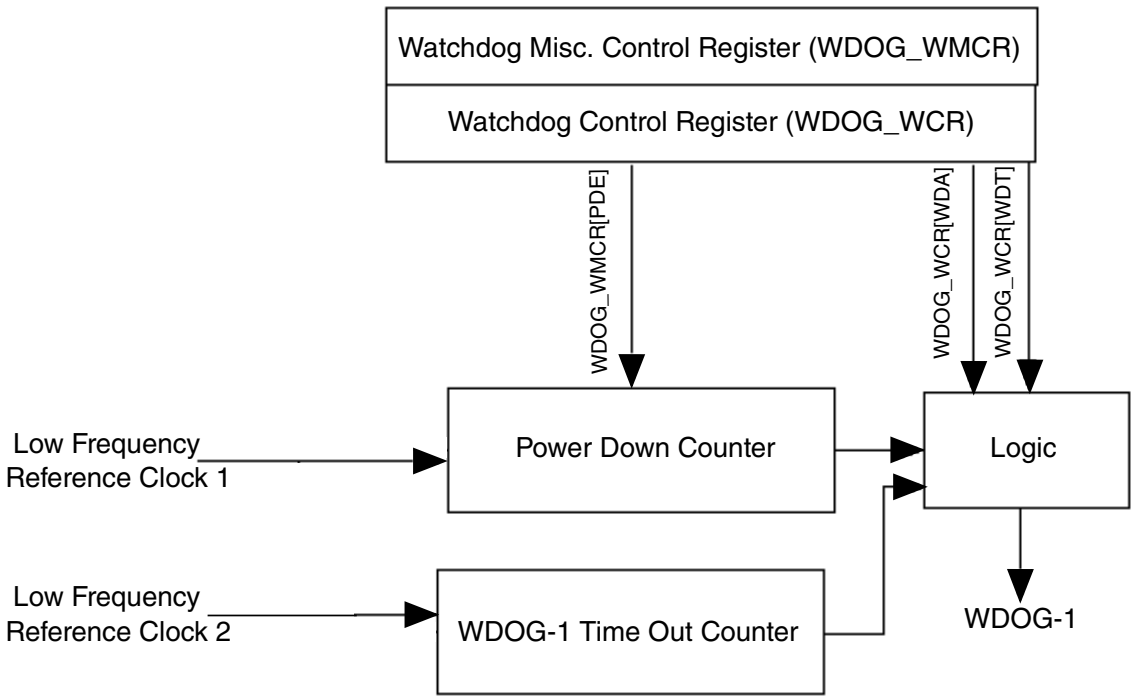


Figure 57-2. WDOG_B (ipp_wdog) generation

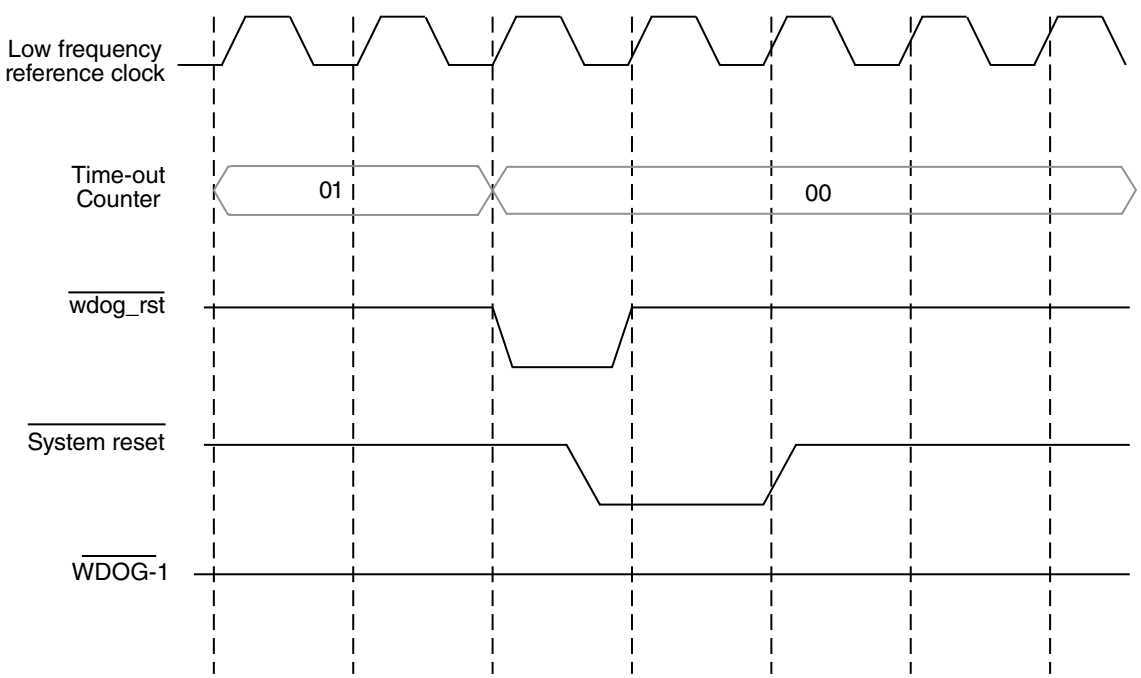


Figure 57-3. WDOG timeout condition/WDT bit is not set

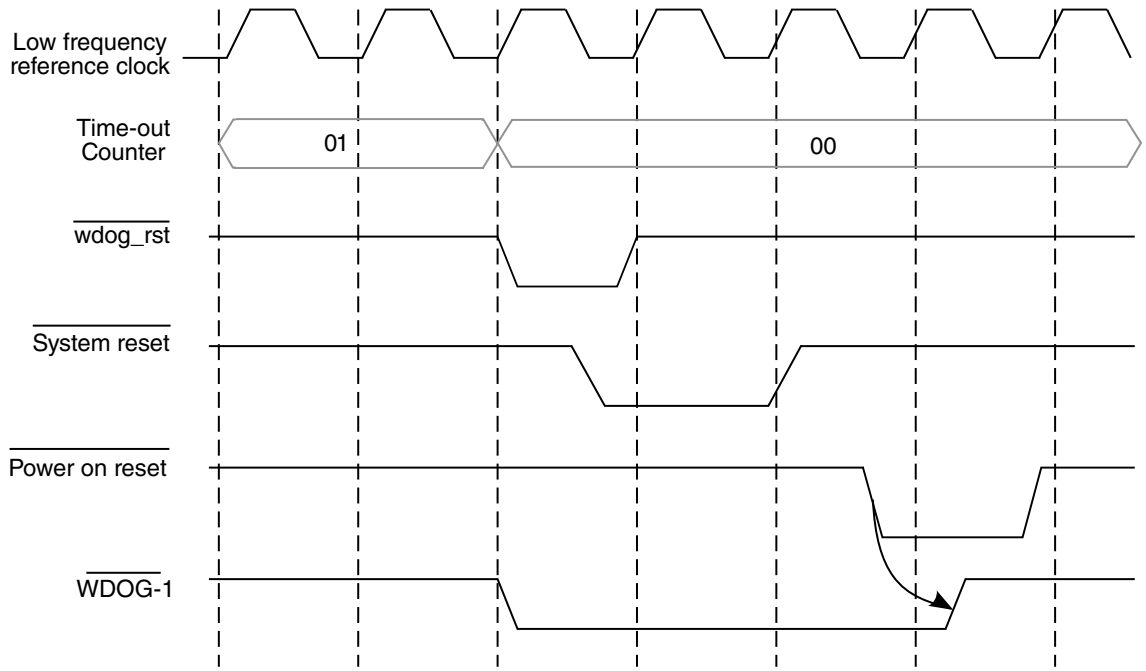


Figure 57-4. WDOG timeout condition/WDT bit is set

57.6.7 Reset

The block is reset by a system reset and the WDOG counter will be disabled. The power-down counter is enabled and starts counting.

57.6.8 Interrupt

The WDOG has the feature of Interrupt generation before timeout.

The interrupt will be generated only if the WIE bit in Watchdog Interrupt Control (WICR) is set. The exact time at which the interrupt should occur (prior to timeout) depends on the value of WICT field of Watchdog Interrupt Control (WICR). For example, if the WICT field has a value 0x04, then the interrupt will be generated two seconds prior to timeout. Once the interrupt is triggered the WTIS bit in Watchdog Interrupt Control (WICR) will be set. The software needs to clear this bit to deassert the interrupt. If the WDOG is serviced before the interrupt generation then the counter will be reloaded with the timeout value WT[7:0] of Watchdog Control (WCR) and interrupt would not be triggered.

57.6.9 Flow Diagrams

A flow diagram of WDOG operation is shown below.

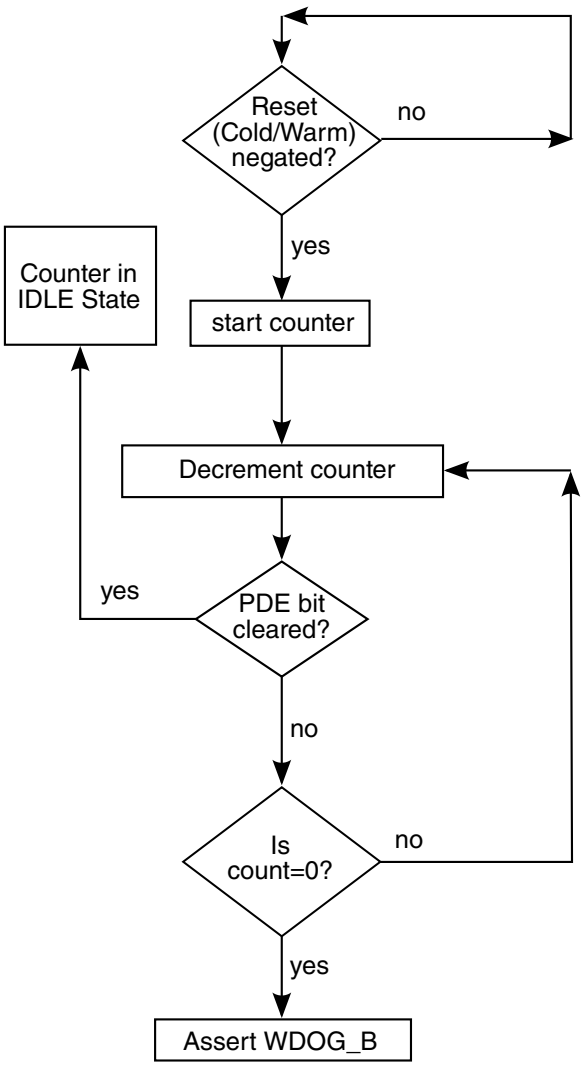


Figure 57-5. Power-Down Counter Flow Diagram

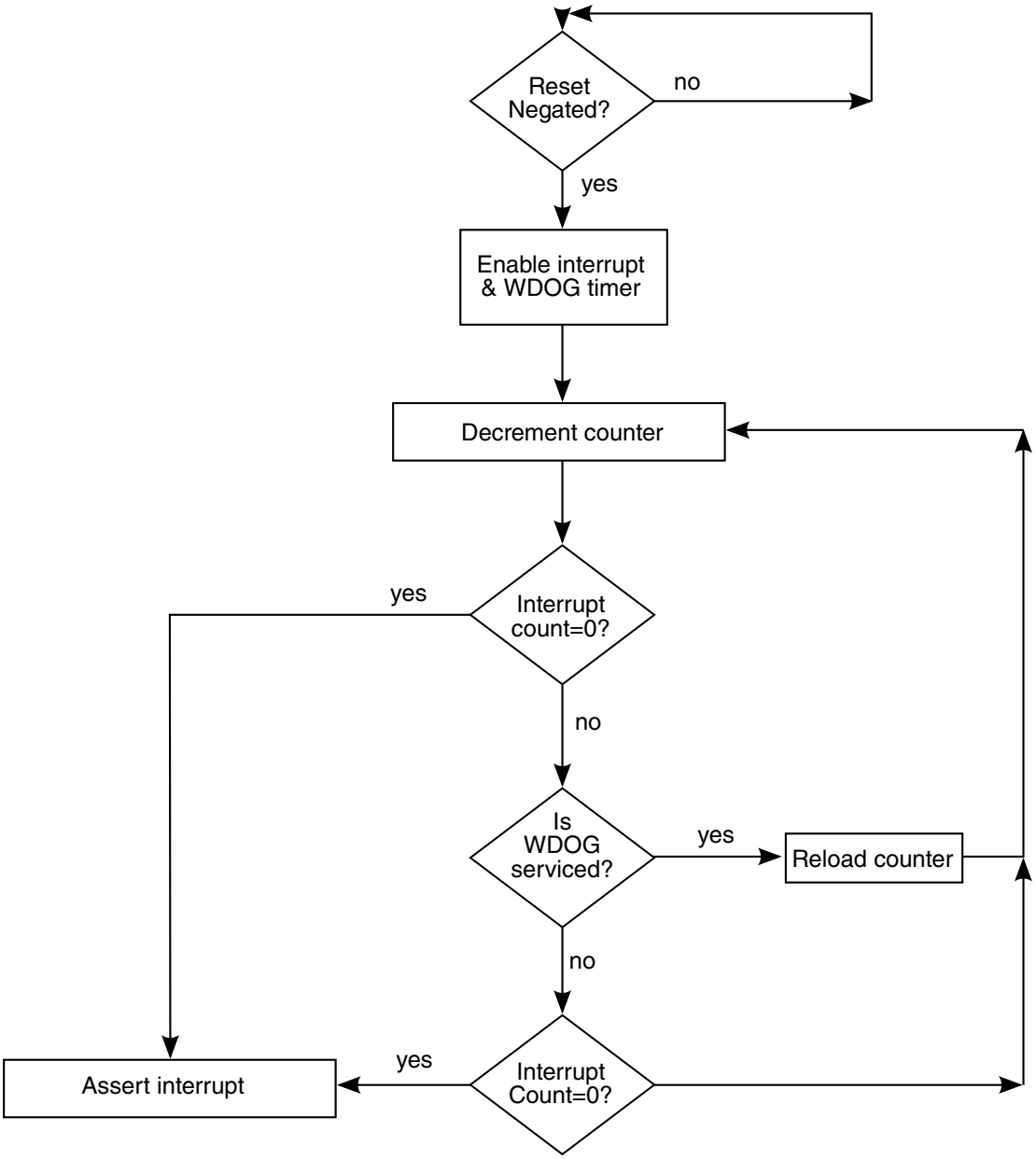


Figure 57-6. Interrupt Generation Flow Diagram

57.7 Initialization

The following sequence should be performed for WDOG initialization.

- PDE bit of Watchdog Miscellaneous Control (WMCR) should be cleared to disable the power down counter.

- WT field of Watchdog Control (WCR) should be programmed for sufficient timeout value.
- WDOG should be enabled by setting WDE bit of Watchdog Control (WCR) so that the timeout counter loads the WT field value of Watchdog Control (WCR) and starts counting.

57.8 WDOG Memory Map/Register Definition

57.8.1 WDOG Register Descriptions

The WDOG has user-accessible, 16-bit registers used to configure, operate, and monitor the state of the Watchdog Timer. Byte operations can be performed on these registers. If a 32-bit access is performed,the WDOG will not generate a peripheral bus error but will behave normally, like a 16-Bit access, making read/write possible. A 32-Bit access should be avoided, as the system may go to an unknown state.

57.8.1.1 WDOG Memory Map

WDOG1 base address: 400B_8000h.

WDOG2 base address: 400D_0000h.

Offset	Register	Width (In bits)	Access	Reset value
0h	Watchdog Control (WCR)	16	RW	0030h
2h	Watchdog Service (WSR)	16	RW	0000h
4h	Watchdog Reset Status (WRSR)	16	RO	0010h
6h	Watchdog Interrupt Control (WICR)	16	RW	0004h
8h	Watchdog Miscellaneous Control (WMCR)	16	RW	0001h

57.8.1.2 Watchdog Control (WCR)

57.8.1.2.1 Address

Register	Offset
WCR	0h

57.8.1.2.2 Function

The Watchdog Control Register (WDOG_WCR) controls the WDOG operation.

- WDZST, WDBG and WDW are write-once only bits. Once the software does a write access to these bits, they will be locked and cannot be reprogrammed until the next system reset assertion.
- WDE is a write one once only bit. Once software performs a write "1" operation to this bit it cannot be reset/cleared until the next system reset.
- WDT is also a write one once only bit. Once software performs a write "1" operation to this bit it cannot be reset/cleared until the next POR. This bit does not get reset/cleared due to any system reset.

57.8.1.2.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	WT								WDW	SRE	WDA	SRS	WDT	WDE	WDBG	WDZST
Reset	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0

57.8.1.2.4 Fields

Field	Function
15-8 WT	WT Watchdog Time-out Field. This 8-bit field contains the time-out value that is loaded into the Watchdog counter after the service routine has been performed or after the Watchdog is enabled. After reset, WT[7:0] must have a value written to it before enabling the Watchdog otherwise count value of zero which is 0.5 seconds is loaded into the counter. **NOTE:** The time-out value can be written at any point of time but it is loaded to the counter at the time when WDOG is enabled or after the service routine has been performed. For more information see Timeout event . 00000000b - - 0.5 Seconds (Default). 00000001b - - 1.0 Seconds. 00000010b - - 1.5 Seconds. 00000011b - - 2.0 Seconds. 11111111b - - 128 Seconds.

Table continues on the next page...

Field	Function
7 WDW	WDW Watchdog Disable for Wait. This bit determines the operation of WDOG during Low Power WAIT mode. This is a write once only bit. 0b - Continue WDOG timer operation (Default). 1b - Suspend WDOG timer operation.
6 SRE	software reset extension, an option way to generate software reset adopt a new way to generate a more robust software reset. This bit can be set/clear with IP bus and will be reset with power-on reset . 0b - using original way to generate software reset (default) 1b - using new way to generate software reset.
5 WDA	WDA WDOG_B assertion. Controls the software assertion of the WDOG_B signal. 0b - Assert WDOG_B output. 1b - No effect on system (Default).
4 SRS	SRS Software Reset Signal. Controls the software assertion of the WDOG-generated reset signal WDOG_RESET_B_DEB . This bit automatically resets to "1" after it has been asserted to "0". **NOTE:** This bit does not generate the software reset to the block. 0b - Assert system reset signal. 1b - No effect on the system (Default).
3 WDT	WDT WDOG_B Time-out assertion. Determines if the WDOG_B gets asserted upon a Watchdog Time-out Event. This is a write-one once only bit. **NOTE:** There is no effect on WDOG_RESET_B_DEB (WDOG Reset) upon writing on this bit. WDOG_B gets asserted along with WDOG_RESET_B_DEB if this bit is set. 0b - No effect on WDOG_B (Default). 1b - Assert WDOG_B upon a Watchdog Time-out event.
2 WDE	WDE Watchdog Enable. Enables or disables the WDOG block. This is a write one once only bit. It is not possible to clear this bit by a software write, once the bit is set. **NOTE:** This bit can be set/reset in debug mode (exception). 0b - Disable the Watchdog (Default). 1b - Enable the Watchdog.
1 WDBG	WDBG Watchdog DEBUG Enable. Determines the operation of the WDOG during DEBUG mode. This bit is write once only. 0b - Continue WDOG timer operation (Default). 1b - Suspend the watchdog timer.
0 WDZST	WDZST Watchdog Low Power. Determines the operation of the WDOG during low-power modes. This bit is write once-only. **NOTE:** The WDOG can continue/suspend the timer operation in the low-power modes (STOP and DOZE mode). 0b - Continue timer operation (Default). 1b - Suspend the watchdog timer.

57.8.1.3 Watchdog Service (WSR)

57.8.1.3.1 Address

Register	Offset
WSR	2h

57.8.1.3.2 Function

When enabled, the WDOG requires that a service sequence be written to the Watchdog Service Register (WSR) to prevent the timeout condition.

NOTE

Executing the service sequence will reload the WDOG timeout counter.

57.8.1.3.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	WSR															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

57.8.1.3.4 Fields

Field	Function
15-0 WSR	WSR Watchdog Service Register. This 16-bit field contains the Watchdog service sequence. Both writes must occur in the order listed prior to the time-out, but any number of instructions can be executed between the two writes. The service sequence must be performed as follows: 0101010101010101b - Write to the Watchdog Service Register (WDOG_WSR). 1010101010101010b - Write to the Watchdog Service Register (WDOG_WSR).

57.8.1.4 Watchdog Reset Status (WRSR)

57.8.1.4.1 Address

Register	Offset
WRSR	4h

57.8.1.4.2 Function

The WRSR is a read-only register that records the source of the output reset assertion. It is not cleared by a hard reset. Therefore, only one bit in the WRSR will always be asserted high. The register will always indicate the source of the last reset generated due to WDOG. Read access to this register is with one wait state. Any write performed on this register will generate a Peripheral Bus Error .

A reset can be generated by the following sources, as listed in priority from highest to lowest:

- Watchdog Time-out
- Software Reset

57.8.1.4.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0											POR	0		TOUT	SFTW
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0

57.8.1.4.4 Fields

Field	Function
15-5 —	Reserved.
4 POR	POR Power On Reset. Indicates whether the reset is the result of a power on reset. 0b - Reset is not the result of a power on reset. 1b - Reset is the result of a power on reset.
3-2 —	Reserved.
1 TOUT	TOUT Timeout. Indicates whether the reset is the result of a WDOG timeout.

Table continues on the next page...

Field	Function
	0b - Reset is not the result of a WDOG timeout. 1b - Reset is the result of a WDOG timeout.
0 SFTW	SFTW Software Reset. Indicates whether the reset is the result of a WDOG software reset by asserting SRS bit 0b - Reset is not the result of a software reset. 1b - Reset is the result of a software reset.

57.8.1.5 Watchdog Interrupt Control (WICR)

57.8.1.5.1 Address

Register	Offset
WICR	6h

57.8.1.5.2 Function

The WDOG_WICR controls the WDOG interrupt generation.

57.8.1.5.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	WIE	WTIS	0						WICT							
W		w1c														
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

57.8.1.5.4 Fields

Field	Function
15 WIE	WIE Watchdog Timer Interrupt enable bit. Reset value is 0. **NOTE:** This bit is a write once only bit. Once the software does a write access to this bit, it will get locked and cannot be reprogrammed until the next system reset assertion 0b - Disable Interrupt (Default). 1b - Enable Interrupt.
14 WTIS	WTIS

Table continues on the next page...

Field	Function
	Watchdog TImer Interrupt Status bit will reflect the timer interrupt status, whether interrupt has occurred or not. Once the interrupt has been triggered software must clear this bit by writing 1 to it. 0b - No interrupt has occurred (Default). 1b - Interrupt has occurred
13-8 —	Reserved.
7-0 WICT	WICT Watchdog Interrupt Count Time-out (WICT) field determines, how long before the counter time-out must the interrupt occur. The reset value is 0x04 implies interrupt will occur 2 seconds before time-out. The maximum value that can be programmed to WICT field is 127.5 seconds with a resolution of 0.5 seconds. **NOTE:** This field is write once only. Once the software does a write access to this field, it will get locked and cannot be reprogrammed until the next system reset assertion. 00000000b - WICT[7:0] = Time duration between interrupt and time-out is 0 seconds. 00000001b - WICT[7:0] = Time duration between interrupt and time-out is 0.5 seconds. 00000100b - WICT[7:0] = Time duration between interrupt and time-out is 2 seconds (Default). 11111111b - WICT[7:0] = Time duration between interrupt and time-out is 127.5 seconds.

57.8.1.6 Watchdog Miscellaneous Control (WMCR)

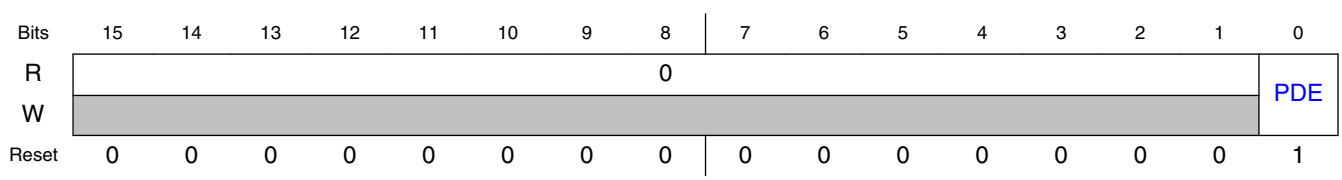
57.8.1.6.1 Address

Register	Offset
WMCR	8h

57.8.1.6.2 Function

WDOG_WMCR Controls the Power Down counter operation.

57.8.1.6.3 Diagram



57.8.1.6.4 Fields

Field	Function
15-1 —	Reserved.
0 PDE	PDE Power Down Enable bit. Reset value of this bit is 1, which means the power down counter inside the WDOG is enabled after reset. The software must write 0 to this bit to disable the counter within 16 seconds of reset de-assertion. Once disabled this counter cannot be enabled again. See Power-down counter event for operation of this counter. **NOTE:** This bit is write-one once only bit. Once software sets this bit it cannot be reset until the next system reset. 0b - Power Down Counter of WDOG is disabled. 1b - Power Down Counter of WDOG is enabled (Default).

Chapter 58
RTWDOG (WDOG3)

58.1 Chip-specific RTWDOG information

Table 58-1. Reference links to related information

Topic	Related module(s)	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
Interrupts, DMA Events and XBAR Assignments	-	Interrupts, DMA Events and XBAR Assignments

The clock source options for this module includes the following items, selected via CS[CLK].

- 00b: bus clock (BUS_CLK)
- 01b: Low-Power Oscillator clock (LPO_CLK)
- 10b: internal clock (INTCLK)
- 11b: external clock (ERCLK)

NOTE

- On this device, both LPO clock and INTCLK are 32 KHz clock generated by 32 KHz XTAL, which could be automatically switched to 32 KHz RCOSC upon XTAL clock loss. ERCLK is 1 MHz RCOSC clock generated by ANATOP.

58.2 Overview

The Watchdog Timer (WDOG) module is an independent timer that is available for system use. It provides a safety feature to ensure that software is executing as planned and that the CPU is not stuck in an infinite loop or executing unintended code. If the WDOG module is not serviced (refreshed) within a certain period, it resets the MCU.

58.2.1 Block diagram

The following figure shows a block diagram of the WDOG module.

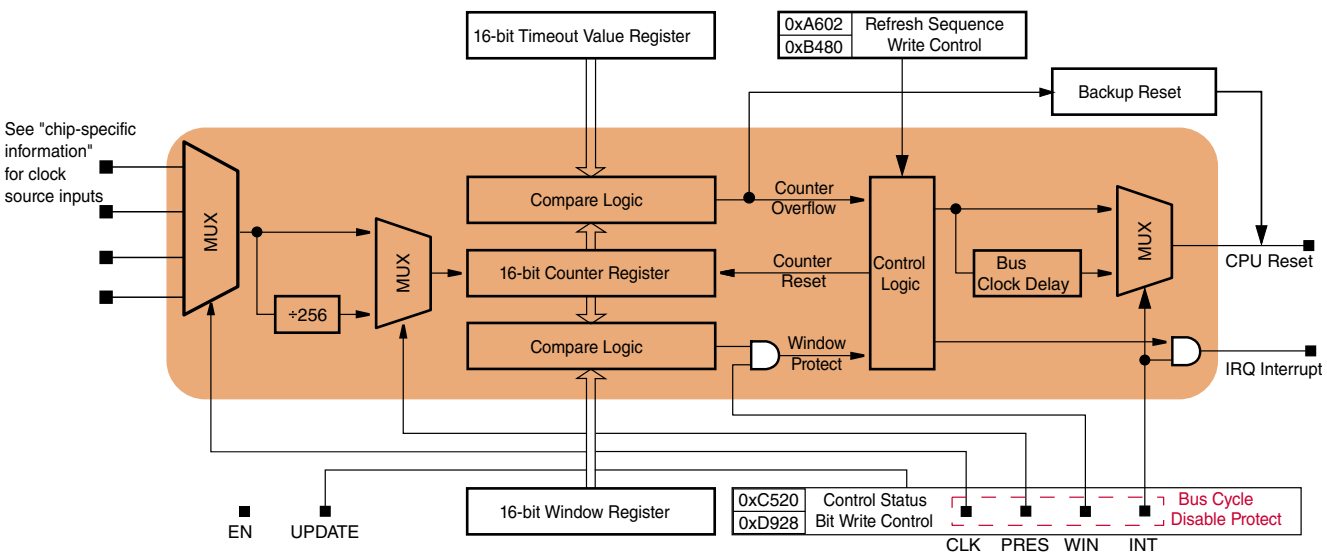


Figure 58-1. WDOG block diagram

58.2.2 Features

Features of the WDOG module include:

- Configurable clock source inputs independent from the bus clock
- Programmable timeout period
  - Programmable 16-bit timeout value
  - Optional fixed 256 clock prescaler when longer timeout periods are needed
- Robust write sequence for counter refresh

    - Refresh sequence of writing 0xA602 and then 0xB480
- Window mode option for the refresh mechanism
    - Programmable 16-bit window value
    - Provides robust check that program flow is faster than expected
    - Early refresh attempts trigger a reset.
- Optional timeout interrupt to allow post-processing diagnostics
    - Interrupt request to CPU with interrupt vector for an interrupt service routine (ISR)
    - Forced reset occurs 128 bus clocks after the interrupt vector fetch.
- Configuration bits are write-once-after-reset to ensure watchdog configuration cannot be mistakenly altered.
- Robust write sequence for unlocking write-once configuration bits
    - Unlock sequence of writing 0xC520 and then 0xD928 for allowing updates to write-once configuration bits
    - Software must make updates within 128 bus clocks after unlocking and before WDOG closing unlock window.

58.3 Functional description

The WDOG module provides a fail safe mechanism to ensure the system can be reset to a known state of operation in case of system failure, such as the CPU clock stopping or there being a run away condition in the software code. The watchdog counter runs continuously off a selectable clock source and expects to be serviced (refreshed) periodically. If it is not, it generates a reset triggering event.

58.3.1 Watchdog refresh mechanism

The watchdog resets the MCU if the watchdog counter is not refreshed. A robust refresh mechanism makes it very unlikely that the watchdog can be refreshed by runaway code.

To refresh the watchdog counter, software must execute a refresh write sequence before the timeout period expires. In addition, if window mode is used, software must not start the refresh sequence until after the time value set in the WIN register. See the following figure.

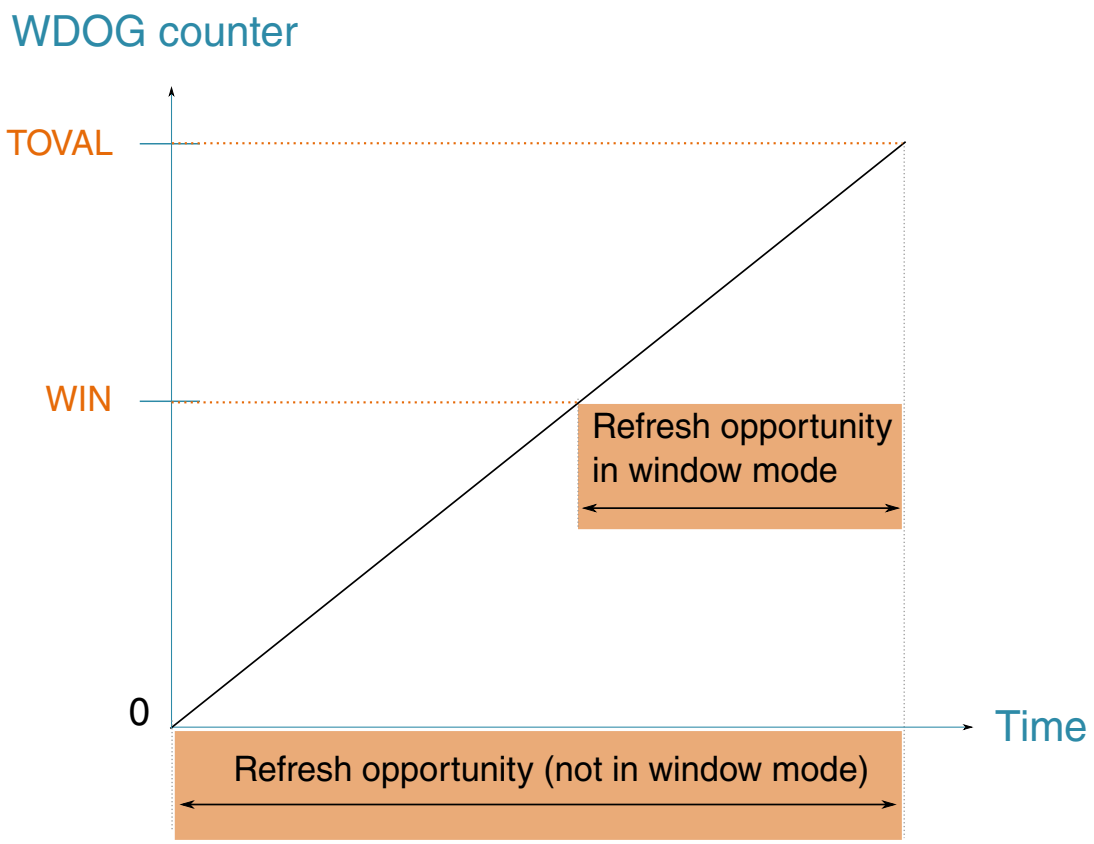


Figure 58-2. Refresh opportunity for the Watchdog counter

58.3.1.1 Window mode

Software finishing its main control loop faster than expected could be an indication of a problem. Depending on the requirements of the application, the WDOG can be programmed to force a reset when refresh attempts are early.

When Window mode is enabled, the watchdog must be refreshed after the counter has reached a minimum expected time value; otherwise, the watchdog resets the MCU. The minimum expected time value is specified in the WIN register. Setting CS[WIN] enables Window mode.

58.3.1.2 Refreshing the Watchdog

The refresh write sequence can be

- either two 16-bit writes (0xA602, 0xB480) or four 8-bit writes (0xA6, 0x02, 0xB4, 0x80) if WDOG_CS[CMD32EN] is 0;
- one 32-bit write (0xB480_A602) if WDOG_CS[CMD32EN] is 1.

to the CNT register. Both methods must occur before the WDG timeout; otherwise, the watchdog resets the MCU.

Note

Before starting the refresh sequence, disable the global interrupts. Otherwise, an interrupt could effectively invalidate the refresh sequence, if the interrupt occurs before the refresh writes finish. After the sequence finishes, restore the global interrupt control state.

The example codes can be found at the end of this chapter.

58.3.2 Configuring the Watchdog

58.3.2.1 Configuring the Watchdog Once

All watchdog control bits, timeout value, and window value are write-once after reset . This means that after a write has occurred they cannot be changed unless a reset occurs. This is guaranteed by the user configuring the window and timeout value first, followed by the other control bits, and ensuring that CS[UPDATE] is also set to 0.

This provides a robust mechanism to configure the watchdog and ensure that a runaway condition cannot mistakenly disable or modify the watchdog configuration after configured.

The new configuration takes effect only after all registers except CNT are written after reset. Otherwise, the WDOG uses the reset values by default. If window mode is not used (CS[WIN] is 0), writing to WIN is not required to make the new configuration take effect.

58.3.2.2 Reconfiguring the Watchdog

In some cases (like when supporting a bootloader function), you may want to reconfigure or disable the watchdog, *without forcing a reset first*.

- By setting CS[UPDATE] to 1 on the initial configuration of the watchdog after a reset, you can reconfigure the watchdog at any time by executing an unlock sequence.
- Conversely, if CS[UPDATE] remains 0, the only way to reconfigure the watchdog is by initiating a reset.

The unlock sequence is similar to the refresh sequence but uses different values.

58.3.2.2.1 Unlocking the Watchdog

The unlock sequence is a write to the CNT register of 0xC520 followed by 0xD928 within 16 bus clocks at any time after the watchdog has been configured. On completing the unlock sequence, the user must reconfigure the watchdog within 128 bus clocks; otherwise, the watchdog closes the unlock window.

NOTE

Due to the 128 bus clocks requirement for reconfiguring the watchdog, some delays must be inserted before executing STOP or WAIT instructions after reconfiguring the watchdog. This ensures that the watchdog's new configuration takes effect before the MCU enters low power mode. Otherwise, the MCU may not be waken up from low power mode.

The example codes can be found at end of this chapter.

58.3.3 Functionality in debug and low-power modes

By default, the watchdog is not functional in Debug mode, Wait mode, or Stop mode. However, the watchdog can remain functional in these modes as follows:

- **For Debug mode,** set CS[DBG]. (This way the watchdog is functional in Debug mode even when the CPU is held by the Debug module.)
- **For Wait mode,** set CS[WAIT].
- **For Stop mode,** set CS[STOP], CS[WAIT], and ensure the clock source is active in Stop mode.

NOTE

The watchdog can generate interrupt in Stop mode.

For Debug mode and Stop mode, in addition to the above configurations, a clock source other than the bus clock must be used as the reference clock for the counter; otherwise, the watchdog cannot function.

58.3.4 Fast testing of the watchdog

Before executing application code in safety critical applications, users are required to test that the watchdog works as expected and resets the MCU. Testing every bit of a 16-bit counter by letting it run to the overflow value takes a relatively long time (64 k clocks).

To help minimize the startup delay for application code after reset, the watchdog has a feature to test the watchdog more quickly by splitting the counter into its constituent byte-wide stages. The low and high bytes are run independently and tested for timeout against the corresponding byte of the timeout value register. (For complete coverage when testing the high byte of the counter, the test feature feeds the input clock via the 8th bit of the low byte, thus ensuring that the overflow connection from the low byte to the high byte is tested.)

Using this test feature reduces the test time to 512 clocks (not including overhead, such as user configuration and reset vector fetches). To further speed testing, use a faster clock (such as the bus clock) for the counter reference.

On a power-on reset, the POR bit in the system reset register is set, indicating the user should perform the WDOG fast test.

58.3.4.1 Testing each byte of the counter

The test procedure follows these steps:

1. Program the preferred watchdog timeout value in the TOVAL register during the watchdog configuration time period.
2. Select a byte of the counter to test using the CS[TST] = 10b for the low byte; CS[TST] = 11b for the high byte.
3. Wait for the watchdog to timeout. Optionally, in the idle loop, increment RAM locations as a parallel software counter for later comparison. Because the RAM is not affected by a watchdog reset, the timeout period of the watchdog counter can be compared with the software counter to verify the timeout period has occurred as expected.
4. The watchdog counter times out and forces a reset.
5. Confirm the WDOG flag in the system reset register is set, indicating that the watchdog caused the reset. (The POR flag remains clear.)
6. Confirm that CS[TST] shows a test (10b or 11b) was performed.

If confirmed, the count and compare functions work for the selected byte. Repeat the procedure, selecting the other byte in step 2.

NOTE

CS[TST] is cleared by a POR only and not affected by other resets.

58.3.4.2 Entering user mode

After successfully testing the low and high bytes of the watchdog counter, the user can configure CS[TST] to 01b to indicate the watchdog is ready for use in application user mode. Thus if a reset occurs again, software can recognize the reset trigger as a real watchdog reset caused by runaway or faulty application code.

As an ongoing test when using the default clock source, software can periodically read the CNT register to ensure the counter is being incremented.

58.3.5 Clock source

The watchdog counter has the clock source options selected by programming CS[CLK]. See the "chip-specific information" section for available clock inputs and the default option for this device.

The options allow software to select a clock source independent of the bus clock for applications that need to meet more robust safety requirements. Using a clock source other than the bus clock ensures that the watchdog counter continues to run if the bus clock is somehow halted; see Backup reset.

NOTE

The *default* clock source of WDOG should be enabled after its functional reset is deasserted, for the WDOG module to function properly.

An optional fixed prescaler for all clock sources allows for longer timeout periods. When CS[PRES] is set, the clock source is prescaled by 256 before clocking the watchdog counter.

The following table summarizes the different watchdog timeout periods which could be available, as an example.

Table 58-2. Watchdog timeout availability

Reference clock	Prescaler	Watchdog time-out availability
REF_CLK	Pass through	1×RCP to 65535×RCP [1]
	Enable	256×RCP to 16776960×RCP

1. RCP means Reference Clock Period.

NOTE

When the programmer switches clock sources during reconfiguration, the watchdog hardware holds the counter at zero for 2.5 periods of the previous clock source and 2.5 periods of the new clock source after the configuration time period (128 bus clocks) ends. This delay ensures a smooth transition before restarting the counter with the new configuration.

58.3.6 Backup reset

NOTE

A clock source other than the bus clock must be used as the reference clock for the counter; otherwise, the backup reset function is not available.

The backup reset function is a safeguard feature that independently generates a reset in case the main WDOG logic loses its clock (the bus clock) and can no longer monitor the counter. If the watchdog counter overflows twice in succession (without an intervening reset), the backup reset function takes effect and generates a reset.

Backup reset becomes valid when interrupt is enabled and the watchdog clock is from non bus clock. If interrupt is enabled, once the bus clock is cut off before exiting interrupt routine, the normal watchdog reset will be blocked. Under this case, the second overflow will cause backup reset directly.

58.3.7 Using interrupts to delay resets

- **When interrupts are enabled (CS[INT] = 1):** After a reset-triggering event (like a counter timeout or invalid refresh attempt), the watchdog first generates an interrupt request. Next, the watchdog delays 128 bus clocks (from the interrupt vector fetch, not the reset-triggering event) before forcing a reset, to allow the interrupt service routine (ISR) to perform tasks (like analyzing the stack to debug code).
- **When interrupts are disabled (CS[INT] = 0):** the watchdog does not delay before forcing a reset.

58.4 Application Information

The watchdog is enabled by default after reset. To disable or reconfigure the watchdog, it is better to be done before the first watchdog timeout. It is suggested to disable or reconfigure the watchdog at the very beginning of the software code, e.g. beginning of the startup or main function.

NOTE

When the watchdog is configured by user, it needs at least 2.5 periods of watchdog clock to take effect. This means interval between two configures by user must be larger than 2.5 clocks.

NOTE

When Chip startup from BOOT ROM then jump to flash, the watchdog would be disabled in the beginning of bootloader, and enabled when bootloader exits. If there is any code in the flash program want to reconfigure the watchdog, it must be run 2.5 watchdog clocks later after the bootloader exits.

To disable or reconfigure the watchdog without forcing a reset, WDOG_CS[UPDATE] bit must be set during the initial configuration of the WDOG module. Then, the unlock sequence can be used at any time within the timeout limit to reconfigure the watchdog.

58.4.1 Disable Watchdog

To disable the watchdog, first do unlock sequence, then unset the WDOG_CS[EN] bit. The code snippet below shows an example for 32-bit write.

```
DisableInterrupts; // disable global interrupt
WDOG_CNT = 0xD928C520; //unlock watchdog
WDOG_CS &= ~WDOG_CS_EN_MASK; //disable watchdog
EnableInterrupts; //enable global interrupt
```

58.4.2 Disable Watchdog after Reset

All of watchdog registers are unlocked by reset. Therefore, unlock sequence is unnecessary, but it needs to write all of watchdog registers to make the new configuration take effect. The code snippet below shows an example of disabling watchdog after reset.

```
DisableInterrupts; // disable global interrupt
WDOG_CS &= ~WDOG_CS_EN_MASK; // disable watchdog
WDOG_TOVAL= 0xFFFF;
while(WDOG_CS[ULK]);  // waiting for lock
while(~WDOG_CS[RCS]); // waiting for new configuration to take effect
EnableInterrupts; // enable global interrupt
```

58.4.3 Configure Watchdog

The watchdog can be configured once by setting the WDOG_CS[UPDATE]=0. After that, the watchdog cannot be reconfigured until a reset. If set WDOG_CS[UPDATE]=1 when configuring the watchdog, the watchdog can be reconfigured without forcing a reset. The following example code shows how to configure the watchdog without window mode, clock source as LPO, interrupt enabled and timeout value to 256 clocks. The code snippet below shows an example for 32-bit write.

Configure once

```
DisableInterrupts; // disable global interrupt
WDOG_CNT = 0xD928C520; //unlock watchdog
while(WDOG_CS[ULK]==0);  //wait until registers are unlocked
WDOG_TOVAL = 256; //set timeout value
WDOG_CS = WDOG_CS_EN(1) | WDOG_CS_CLK(1) | WDOG_CS_INT(1) |
          WDOG_CS_WIN(0) | WDOG_CS_UPDATE(0);
while(WDOG_CS[RCS]==0); //wait until new configuration takes effect
EnableInterrupts; //enable global interrupt
```

Configure for reconfigurable

```
DisableInterrupts; //disable global interrupt
WDOG_CNT = 0xD928C520; //unlock watchdog
while(WDOG_CS[ULK]==0);  //wait until registers are unlocked
WDOG_TOVAL = 256; //set timeout value
WDOG_CS = WDOG_CS_EN(1) | WDOG_CS_CLK(1) | WDOG_CS_INT(1) |
          WDOG_CS_WIN(0) | WDOG_CS_UPDATE(1);
while(WDOG_CS[RCS]==0); //wait until new configuration takes effect
EnableInterrupts; //enable global interrupt
```

58.4.4 Refreshing the Watchdog

To refresh the watchdog and reset the watchdog counter to zero, a refresh sequence is required. The code snippet below shows an example for 32-bit write.

```
DisableInterrupts; // disable global interrupt
WDOG_CNT = 0xB480A602; // refresh watchdog
EnableInterrupts; // enable global interrupt
```

58.5 Memory map and register definition

58.5.1 WDOG register descriptions

58.5.1.1 WDOG memory map

RTWDOG base address: 400B_C000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Watchdog Control and Status Register (CS)	32	RW	0000_2980h
4h	Watchdog Counter Register (CNT)	32	RW	0000_0000h
8h	Watchdog Timeout Value Register (TOVAL)	32	RW	0000_0400h
Ch	Watchdog Window Register (WIN)	32	RW	0000_0000h

58.5.1.2 Watchdog Control and Status Register (CS)

58.5.1.2.1 Offset

Register	Offset
CS	0h

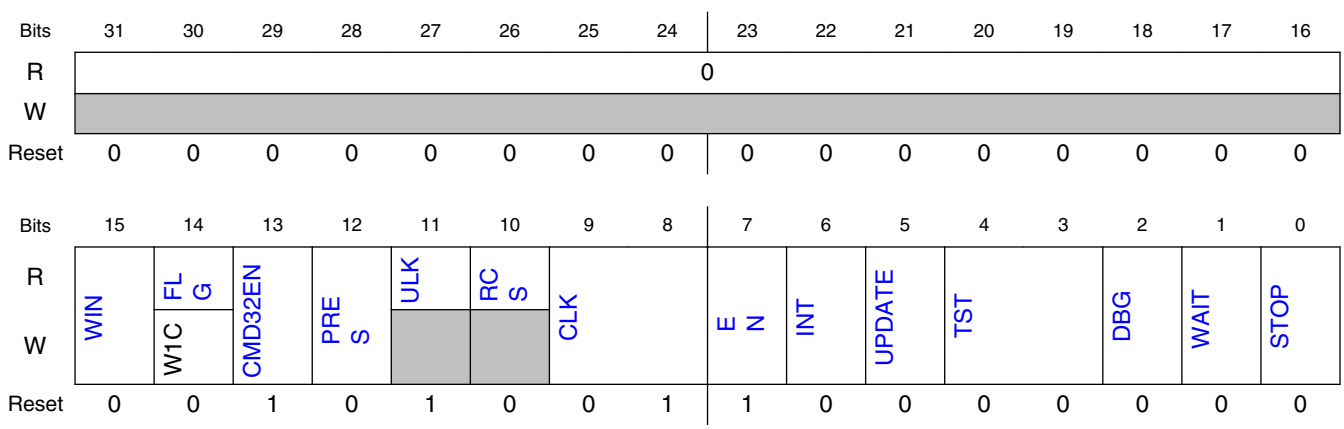
58.5.1.2.2 Function

This section describes the function of Watchdog Control and Status Register.

NOTE

TST is cleared (0:0) on POR only. Any other reset does not affect the value of this field.

58.5.1.2.3 Diagram



58.5.1.2.4 Fields

Field	Function
31-16 —	Reserved
15 WIN	Watchdog Window This write-once bit enables window mode. See the Window mode section. 0b - Window mode disabled. 1b - Window mode enabled.
14 FLG	Watchdog Interrupt Flag This bit is an interrupt indicator when INT is set in control and status register 1. Write 1 to clear it. 0b - No interrupt occurred. 1b - An interrupt occurred.
13 CMD32EN	Enables or disables WDOG support for 32-bit (otherwise 16-bit or 8-bit) refresh/unlock command write words This is write-once field, and the user needs to unlock WDOG after writing this field for reconfiguration. 0b - Disables support for 32-bit refresh/unlock command write words. Only 16-bit or 8-bit is supported. 1b - Enables support for 32-bit refresh/unlock command write words. 16-bit or 8-bit is NOT supported.
12 PRES	Watchdog prescaler This write-once bit enables a fixed 256 pre-scaling of watchdog counter reference clock. (The block diagram shows this clock divider option.) 0b - 256 prescaler disabled. 1b - 256 prescaler enabled.
11 ULK	Unlock status This read-only bit indicates whether WDOG is unlocked or not. 0b - WDOG is locked. 1b - WDOG is unlocked.
10 RCS	Reconfiguration Success This read-only bit indicates whether the reconfiguration is successful or not. Default reset value is 0.This bit is set when new configuration takes effect, and is cleared by successful unlock command. 0b - Reconfiguring WDOG. 1b - Reconfiguration is successful.
9-8 CLK	Watchdog Clock This write-once field indicates the clock source that feeds the watchdog counter. See the "chip-specific information" section.
7 EN	Watchdog Enable This write-once bit enables the watchdog counter to start counting. 0b - Watchdog disabled. 1b - Watchdog enabled.
6 INT	Watchdog Interrupt This write-once bit configures the watchdog to immediately generate an interrupt request upon a reset-triggering event (timeout or illegal write to the watchdog), before forcing a reset. After the interrupt vector fetch (which comes after the reset-triggering event), the reset occurs after a delay of 128 bus clocks. 0b - Watchdog interrupts are disabled. Watchdog resets are not delayed.

Table continues on the next page...

Field	Function
	1b - Watchdog interrupts are enabled. Watchdog resets are delayed by 128 bus clocks from the interrupt vector fetch.
5 UPDATE	Allow updates This write-once bit allows software to reconfigure the watchdog without a reset. 0b - Updates not allowed. After the initial configuration, the watchdog cannot be later modified without forcing a reset. 1b - Updates allowed. Software can modify the watchdog configuration registers within 128 bus clocks after performing the unlock write sequence.
4-3 TST	Watchdog Test Enables the fast test mode. The test mode allows software to exercise all bits of the counter to demonstrate that the watchdog is functioning properly. See the Fast testing of the watchdog section. This write-once field is cleared (0:0) on POR only. Any other reset does not affect the value of this field. 00b - Watchdog test mode disabled. 01b - Watchdog user mode enabled. (Watchdog test mode disabled.) After testing the watchdog, software should use this setting to indicate that the watchdog is functioning normally in user mode. 10b - Watchdog test mode enabled, only the low byte is used. CNT[CNTLOW] is compared with TOVAL[TOVALLOW]. 11b - Watchdog test mode enabled, only the high byte is used. CNT[CNTHIGH] is compared with TOVAL[TOVALHIGH].
2 DBG	Debug Enable This write-once bit enables the watchdog to operate when the chip is in debug mode. 0b - Watchdog disabled in chip debug mode. 1b - Watchdog enabled in chip debug mode.
1 WAIT	Wait Enable This write-once bit enables the watchdog to operate when the chip is in wait mode. 0b - Watchdog disabled in chip wait mode. 1b - Watchdog enabled in chip wait mode.
0 STOP	Stop Enable This write-once bit enables the watchdog to operate when the chip is in stop mode. 0b - Watchdog disabled in chip stop mode. 1b - Watchdog enabled in chip stop mode.

58.5.1.3 Watchdog Counter Register (CNT)

58.5.1.3.1 Offset

Register	Offset
CNT	4h

58.5.1.3.2 Function

This section describes the watchdog counter register.

The watchdog counter register provides access to the value of the free-running watchdog counter. Software can read the counter register at any time.

Software cannot write directly to the watchdog counter; however, two write sequences to these registers have special functions:

1. The *refresh sequence* resets the watchdog counter to 0x0000. See the "Refreshing the Watchdog" section.
2. The *unlock sequence* allows the watchdog to be reconfigured without forcing a reset (when CS[UPDATE] = 1). See the "Configure for reconfigurable" section.

NOTE

All other writes to this register are illegal and force a reset.

58.5.1.3.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	CNTHIGH								CNTLOW							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

58.5.1.3.4 Fields

Field	Function
31-16 —	Reserved
15-8 CNTHIGH	High byte of the Watchdog Counter
7-0 CNTLOW	Low byte of the Watchdog Counter

58.5.1.4 Watchdog Timeout Value Register (TOVAL)

58.5.1.4.1 Offset

Register	Offset
TOVAL	8h

58.5.1.4.2 Function

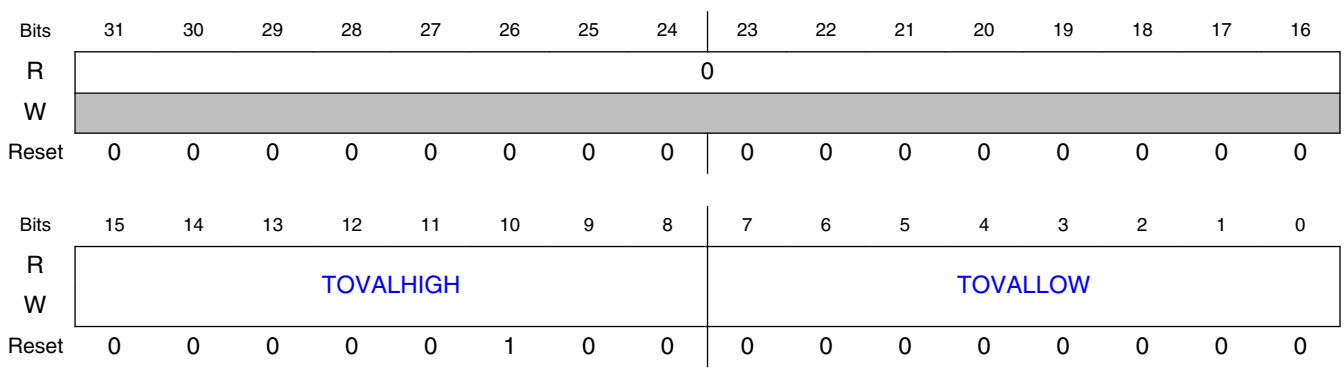
This section describes the watchdog timeout value register. TOVAL contains the 16-bit value used to set the timeout period of the watchdog.

The watchdog counter (CNT) is continuously compared with the timeout value (TOVAL). If the counter reaches the timeout value, the watchdog forces a reset triggering event.

NOTE

Do not write 0 to the TOVAL register (if CS[TST]=11b, then TOVALHIGH cannot be written as 0; if CS[TST]=10b, then TOVALLOW cannot be 0); otherwise, the watchdog always generates a reset.

58.5.1.4.3 Diagram



58.5.1.4.4 Fields

Field	Function
31-16 —	Reserved
15-8 TOVALHIGH	High byte of the timeout value
7-0 TOVALLOW	Low byte of the timeout value

58.5.1.5 Watchdog Window Register (WIN)

58.5.1.5.1 Offset

Register	Offset
WIN	Ch

58.5.1.5.2 Function

This section describes the watchdog window register. When window mode is enabled (CS[WIN] is set), The WIN register determines the earliest time that a refresh sequence is considered valid. See the Watchdog refresh mechanism section.

The WIN register value must be less than the TOVAL register value.

58.5.1.5.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	WINHIGH								WINLOW							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

58.5.1.5.4 Fields

Field	Function
31-16 —	Reserved
15-8 WINHIGH	High byte of Watchdog Window
7-0 WINLOW	Low byte of Watchdog Window

Chapter 59
External Watchdog Monitor (EWM)

59.1 Chip-specific EWM information

Table 59-1. Reference links to related information

Topic	Related module(s)	Reference
System memory map	-	System Memory Map
Clocking	CCM Low Power Clock Source	Clock Management Clock Control Module (CCM) Low Power Clock Source (lpo_clk[3:0])
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
Interrupts, DMA Events and XBAR Assignments	-	Interrupts, DMA Events and XBAR Assignments

NOTE

- lpo_clk[0] is 32 kHz clock generated by 32 kHz XTAL, which could be automatically switched to 32 kHz RCOSC upon XTAL clock loss;
- lpo_clk[1] is 1 MHz RCOSC clock generated by ANATOP;
- lpo_clk[3:2] are not used on this device.

59.2 Overview

For safety, a redundant watchdog system, External Watchdog Monitor (EWM), is designed to monitor external circuits, as well as the MCU software flow. This provides a back-up mechanism to the internal watchdog that resets the MCU's CPU and peripherals.

The watchdog is generally used to monitor the flow and execution of embedded software within an MCU. The watchdog consists of a counter that if allowed to overflow, forces an internal reset (asynchronous) to all on-chip peripherals and optionally asserts the RESET_B pin to reset external devices/circuits. The overflow of the watchdog counter must not occur if the software code works well and services the watchdog to re-start the actual counter.

The EWM differs from the internal watchdog in that it does not reset the MCU's CPU and peripherals. The EWM provides an independent EWM_OUT_b signal that when asserted resets or places an external circuit into a safe mode. The EWM_OUT_b signal is asserted upon the EWM counter time-out. An optional external input EWM_in is provided to allow additional control of the assertion of EWM_OUT_b signal.

59.2.1 Block Diagram

This figure shows the EWM block diagram.

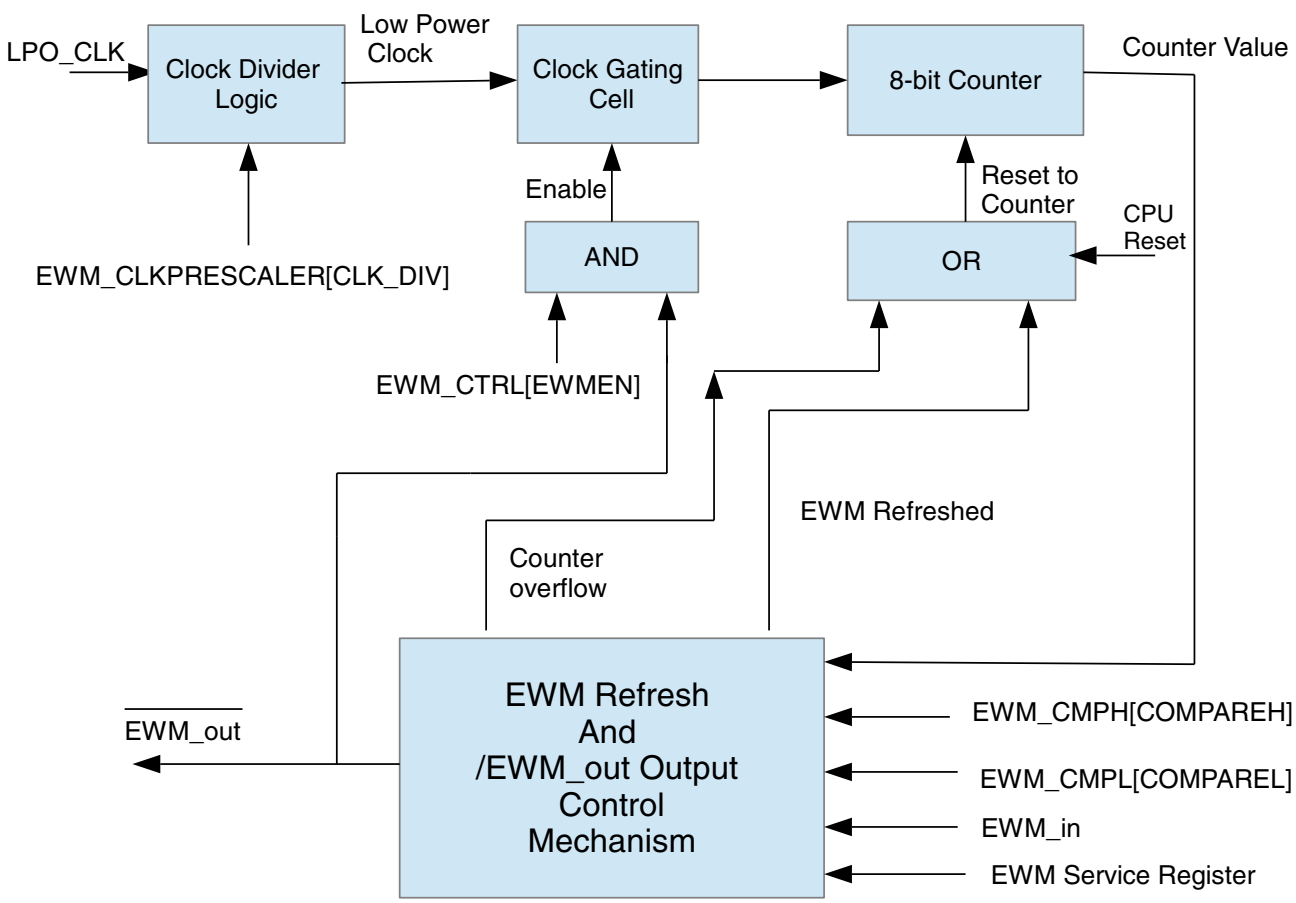


Figure 59-1. EWM Block Diagram

59.2.2 Features

Features of EWM module include:

- Independent LPO_CLK clock source
- Programmable time-out period specified in terms of number of EWM LPO_CLK clock cycles.
- Windowed refresh option
  - Provides robust check that program flow is faster than expected.
  - Programmable window.
  - Refresh outside window leads to assertion of EWM_OUT_b.
- Robust refresh mechanism
  - Write values of 0xB4 and 0x2C to EWM Refresh Register within 63 peripheral bus clock cycles.
- One output port, EWM_OUT_b, when asserted is used to reset or place the external circuit into safe mode.
- One Input port, EWM_in, allows an external circuit to control the assertion of the EWM_OUT_b signal.

59.3 Functional Description

The following sections describe functional details of the EWM module.

NOTE

When the BUS_CLK is lost, then EWM module doesn't generate the EWM_OUT_b signal and no refresh operation is possible

59.3.1 Modes of Operation

This section describes the module's operating modes.

59.3.1.1 Stop Mode

When the EWM is in stop mode, the CPU refreshes to the EWM cannot occur. On entry to stop mode, the EWM's counter freezes.

There are two possible ways to exit from Stop mode:

- On exit from stop mode through a reset, the EWM remains disabled.
- On exit from stop mode by an interrupt, the EWM is re-enabled, and the counter continues to be clocked from the same value prior to entry to stop mode.

Note the following if the EWM enters the stop mode during CPU refresh mechanism: At the exit from stop mode by an interrupt, refresh mechanism state machine starts from the previous state which means, if first refresh command is written correctly and EWM enters the stop mode immediately, the next command has to be written within the next *63* peripheral bus clocks after exiting from stop mode. User must mask all interrupts prior to executing EWM refresh instructions.

59.3.1.2 Wait Mode

The EWM module treats the stop and wait modes as the same. EWM functionality remains the same in both of these modes.

59.3.1.3 Debug Mode

Entry to debug mode has no effect on the EWM.

- If the EWM is enabled prior to entry of debug mode, it remains enabled.
- If the EWM is disabled prior to entry of debug mode, it remains disabled.

59.3.2 The EWM_OUT_b Signal

The EWM_OUT_b is a digital output signal used to gate an external circuit (application specific) that controls critical safety functions. For example, the EWM_OUT_b could be connected to the high voltage transistors circuits that control an AC motor in a large appliance.

The EWM_OUT_b signal remains deasserted when the EWM is being regularly refreshed by the CPU within the programmable refresh window, indicating that the application code is executed as expected.

The EWM_OUT_b signal is asserted in any of the following conditions:

- The EWM refresh occurs when the counter value is less than CMPL value.
- The EWM counter value reaches the CMPH value, and no EWM refresh has occurred.
- If functionality of EWM_in pin is enabled and EWM_in pin is asserted while refreshing the EWM.
- After any reset (by the virtue of the external pull-down mechanism on the EWM_OUT_b pin)

The EWM_OUT_b is asserted after any reset by the virtue of the external pull-down mechanism on the EWM_OUT_b signal. Then, to deassert the EWM_OUT_b signal, set EWMEN bit in the CTRL register to enable the EWM.

If the EWM_OUT_b signal shares its pad with a digital I/O pin, on reset this actual pad defers to being an input signal. The pad state is controlled by the EWM_OUT_b signal only after the EWM is enabled by the EWMEN bit in the CTRL register.

Note

EWM_OUT_b pad must be in pull down state when EWM functionality is used and when EWM is under Reset.

59.3.3 EWM_OUT_b pin state in low power modes

During Wait, Stop and Power Down modes the EWM_OUT_b pin enters a high-impedance state. A user has the option to control the logic state of the pin using an external pull device or by configuring the internal pull device. When the CPU enters a Run mode from Wait or Stop recovery, the pin resumes its previous state before entering Wait or Stop mode. When the CPU enters Run mode from Power Down, the pin returns to its reset state.

59.3.4 The EWM_in Signal

The EWM_in is a digital input signal for safety status of external safety circuits, that allows an external circuit to control the assertion of the EWM_OUT_b signal. For example, in the application, an external circuit monitors a critical safety function, and if there is fault with safety function, the external circuit can then actively initiate the EWM_OUT_b signal that controls the gating circuit.

The EWM_in signal is ignored if the EWM is disabled, or if INEN bit of CTRL register is cleared, as after any reset.

On enabling the EWM (setting the CTRL[EWMEN] bit) and enabling EWM_in functionality (setting the CTRL[INEN] bit), the EWM_in signal must be in the deasserted state prior to the CPU start refreshing the EWM. This ensures that the EWM_OUT_b stays in the deasserted state; otherwise, the EWM_OUT_b output signal is asserted.

Note

The user must update the CMPH and CMPL registers prior to enabling the EWM. After enabling the EWM, the counter resets to zero, therefore the user shall provide a reasonable time after a power-on reset for the external monitoring circuit to stabilize. The user shall also ensure that the EWM_in pin is deasserted.

59.3.5 EWM Counter

It is an 8-bit ripple counter fed from a clock source that is independent of the peripheral bus clock source. As the preferred time-out is between 1 ms and 100 ms the actual clock source should be in the kHz range.

The counter is reset to zero after the CPU reset, or when EWM refresh action completes, or at counter overflow. The counter value is not accessible to the CPU.

59.3.6 EWM Compare Registers

The compare registers CMPL and CMPH are write-once after a CPU reset and cannot be modified until another CPU reset occurs.

The EWM compare registers are used to create a refresh window to refresh the EWM module.

It is illegal to program CMPL and CMPH with same value. In this case, as soon as counter reaches (CMPL + 1), EWM_OUT_b is asserted.

59.3.7 EWM Refresh Mechanism

Other than the initial configuration of the EWM, the CPU can only access the EWM by the EWM Service Register. The CPU must access the EWM service register with correct write of unique data within the windowed time frame as determined by the CMPL and CMPH registers for correct EWM refresh operation. Therefore, three possible conditions can occur:

Table 59-2. EWM Refresh Mechanisms

Condition	Mechanism
An EWM refresh action completes when: CMPL < Counter < CMPH.	The software behaves as expected and the EWM counter is reset to zero. The EWM_OUT_b output signal remains in the deasserted state if, during the EWM refresh action, the EWM_in input has been in deasserted state.
An EWM refresh action completes when Counter < CMPL	The software refreshes the EWM before the windowed time frame, the counter is reset to zero and the EWM_OUT_b output signal is asserted irrespective of the input EWM_in signal.
Counter value reaches CMPH prior to completion of EWM refresh action.	Software has not refreshed the EWM. The EWM counter is reset to zero and the EWM_OUT_b output signal is asserted irrespective of the input EWM_in signal.

59.3.8 EWM Interrupt

When EWM_OUT_b is asserted, an interrupt request is generated to indicate the assertion of the EWM reset out signal. This interrupt is enabled when CTRL[INTEN] is set. Clearing this bit clears the interrupt request but does not affect EWM_OUT_b. The EWM_OUT_b signal can be deasserted only by forcing a system reset.

59.3.9 Selecting the EWM counter clock

There are four possible low power clock sources for the EWM counter. Select one of the available clock sources by programming CLKCTRL[CLKSEL].

59.3.10 Counter clock prescaler

The EWM counter clock source can be prescaled by a clock divider, by programming CLKPRESCALER[CLK_DIV]. This divided clock is used to run the EWM counter.

NOTE

The divided clock used to run the EWM counter must be no more than half the frequency of the bus clock.

59.4 EWM Signal Descriptions

The EWM has two external signals and internal options for the counter clock sources, as shown in the following table.

NOTE

All active-low signals are now represented with the suffix "_b" throughout the chapter.

Table 59-3. EWM Signal Descriptions

Signal	Description	I/O
EWM_in	EWM input for safety status of external safety circuits. The polarity of EWM_in is programmable using the EWM_CTRL[ASSIN] bit. The default polarity is active low.	I
EWM_OUT_b	EWM reset out signal	O
lpo_clk[3:0]	Low power clock sources for running counter	I

59.5 Memory Map/Register Definition

This section contains the module memory map and registers.

NOTE

EWM only supports 8-bit register access. 16-bit and 32-bit access are not possible.

59.5.1 EWM register descriptions

59.5.1.1 EWM memory map

EWM base address: 400B_4000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Control Register (CTRL)	8	RW	00h
1h	Service Register (SERV)	8	WORZ	00h
2h	Compare Low Register (CMPL)	8	RWONCE	00h
3h	Compare High Register (CMPH)	8	RWONCE	FFh
4h	Clock Control Register (CLKCTRL)	8	RWONCE	00h
5h	Clock Prescaler Register (CLKPRESCALER)	8	RWONCE	00h

59.5.1.2 Control Register (CTRL)

59.5.1.2.1 Offset

Register	Offset
CTRL	0h

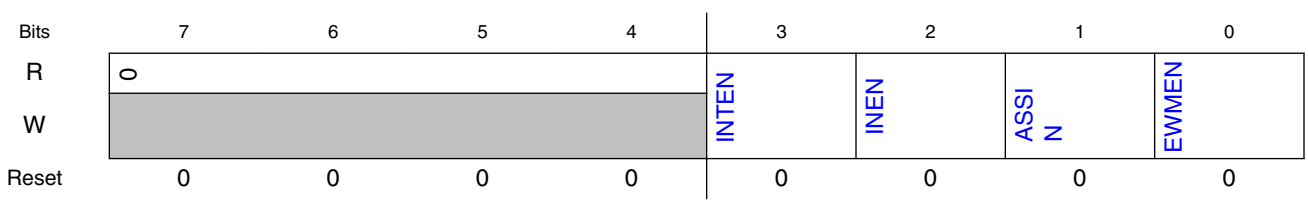
59.5.1.2.2 Function

The CTRL register is cleared by any reset.

NOTE

INEN, ASSIN and EWMEN bits can be written once after a CPU reset. Modifying these bits more than once, generates a bus transfer error.

59.5.1.2.3 Diagram



59.5.1.2.4 Fields

Field	Function
7-4 —	Reserved
3 INTEN	Interrupt Enable. This bit when set and EWM_OUT_b is asserted, an interrupt request is generated. To de-assert interrupt request, user should clear this bit by writing 0. 0b - Deasserts the interrupt request. 1b - Generates an interrupt request, when EWM_OUT_b is asserted.
2 INEN	Input Enable. This bit when set, enables the EWM_in port. 0b - EWM_in port is disabled. 1b - EWM_in port is enabled.
1 ASSIN	EWM_in's Assertion State Select. Default assert state of the EWM_in signal is logic zero. Setting the ASSIN bit inverts the assert state of EWM_in signal to a logic one. 0b - Default assert state of the EWM_in signal. 1b - Inverts the assert state of EWM_in signal.
0 EWMEN	EWM enable. This bit when set, enables the EWM module. This resets the EWM counter to zero and deasserts the EWM_OUT_b signal. This bit when unset, keeps the EWM module disabled. It cannot be re-enabled until a next reset, due to the write-once nature of this bit. 0b - EWM module is disabled. 1b - EWM module is enabled.

59.5.1.3 Service Register (SERV)

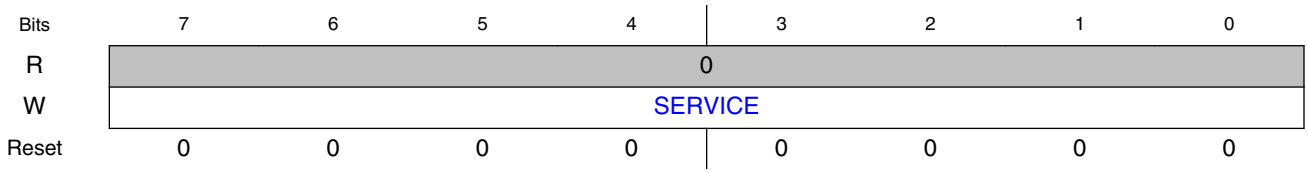
59.5.1.3.1 Offset

Register	Offset
SERV	1h

59.5.1.3.2 Function

The SERV register provides the interface from the CPU to the EWM module. It is write-only and reads of this register return zero.

59.5.1.3.3 Diagram



59.5.1.3.4 Fields

Field	Function
7-0 SERVICE	SERVICE The EWM refresh mechanism requires the CPU to write two values to the SERV register: a first data byte of 0xB4, followed by a second data byte of 0x2C. The EWM refresh is invalid if either of the following conditions is true. • The first or second data byte is not written correctly. • The second data byte is not written within a fixed number of peripheral bus cycles of the first data byte. This fixed number of cycles is called *EWM_refresh_time* . 63 peripheral bus clock cycles are required for *EWM_refresh_time*

59.5.1.4 Compare Low Register (CMPL)

59.5.1.4.1 Offset

Register	Offset
CMPL	2h

59.5.1.4.2 Function

The CMPL register is reset to zero after a CPU reset. This provides no minimum time for the CPU to refresh the EWM counter.

NOTE

This register can be written only once after a CPU reset. Writing this register more than once generates a bus transfer error.

59.5.1.4.3 Diagram

Bits	7	6	5	4	3	2	1	0
R W	COMPAREL							
Reset	0	0	0	0	0	0	0	0

59.5.1.4.4 Fields

Field	Function
7-0 COMPAREL	COMPAREL To prevent runaway code from changing this field, software should write to this field after a CPU reset even if the (default) minimum refresh time is required.

59.5.1.5 Compare High Register (CMPH)

59.5.1.5.1 Offset

Register	Offset
CMPH	3h

59.5.1.5.2 Function

The CMPH register is reset to 0xFF after a CPU reset. This provides a maximum of 256 clocks time, for the CPU to refresh the EWM counter.

NOTE

This register can be written only once after a CPU reset. Writing this register more than once generates a bus transfer error.

NOTE

The valid values for CMPH are up to 0xFE because the EWM counter never expires when CMPH = 0xFF. The expiration happens only if EWM counter is greater than CMPH.

59.5.1.5.3 Diagram

Bits	7	6	5	4	3	2	1	0
R W	COMPAREH							
Reset	1	1	1	1	1	1	1	1

59.5.1.5.4 Fields

Field	Function
7-0 COMPAREH	COMPAREH To prevent runaway code from changing this field, software should write to this field after a CPU reset even if the (default) maximum refresh time is required.

59.5.1.6 Clock Control Register (CLKCTRL)

59.5.1.6.1 Offset

Register	Offset
CLKCTRL	4h

59.5.1.6.2 Function

This CLKCTRL register is reset to 0x00 after a CPU reset.

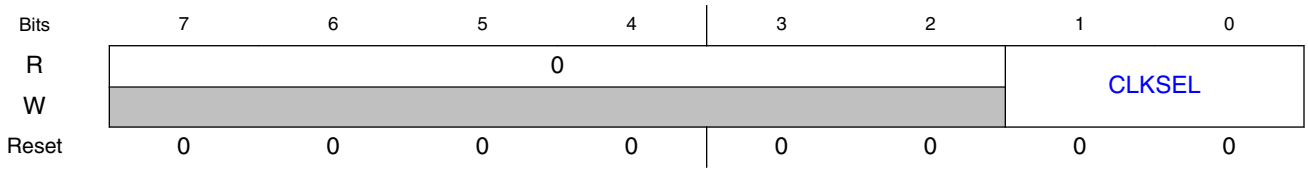
NOTE

This register can be written only once after a CPU reset. Writing this register more than once generates a bus transfer error.

NOTE

User should select the required low power clock before enabling the EWM.

59.5.1.6.3 Diagram



59.5.1.6.4 Fields

Field	Function
7-2 —	reserved
1-0 CLKSEL	CLKSEL EWM has 4 possible low power clock sources for running EWM counter. One of the clock source can be selected by writing into this field. • 00 - lpo_clk[0] will be selected for running EWM counter. • 01 - lpo_clk[1] will be selected for running EWM counter. • 10 - lpo_clk[2] will be selected for running EWM counter. • 11 - lpo_clk[3] will be selected for running EWM counter.

59.5.1.7 Clock Prescaler Register (CLKPRESCALER)

59.5.1.7.1 Offset

Register	Offset
CLKPRESCALER	5h

59.5.1.7.2 Function

This CLKPRESCALER register is reset to 0x00 after a CPU reset.

NOTE

This register can be written only once after a CPU reset. Writing this register more than once generates a bus transfer error.

NOTE

Write the required prescaler value before enabling the EWM.

NOTE

The implementation of this register is chip-specific. See the Chip Configuration details.

59.5.1.7.3 Diagram

Bits	7	6	5	4	3	2	1	0
R W	CLK_DIV							
Reset	0	0	0	0	0	0	0	0

59.5.1.7.4 Fields

Field	Function
7-0 CLK_DIV	CLK_DIV Selected low power clock source for running the EWM counter can be prescaled as below. • Prescaled clock frequency = low power clock source frequency / (1 + CLK_DIV)

Chapter 60
On Chip Cross Triggers Overview

60.1 Overview

This chip integrates an on-chip cross trigger network. The following diagram shows the cross trigger network of this device.

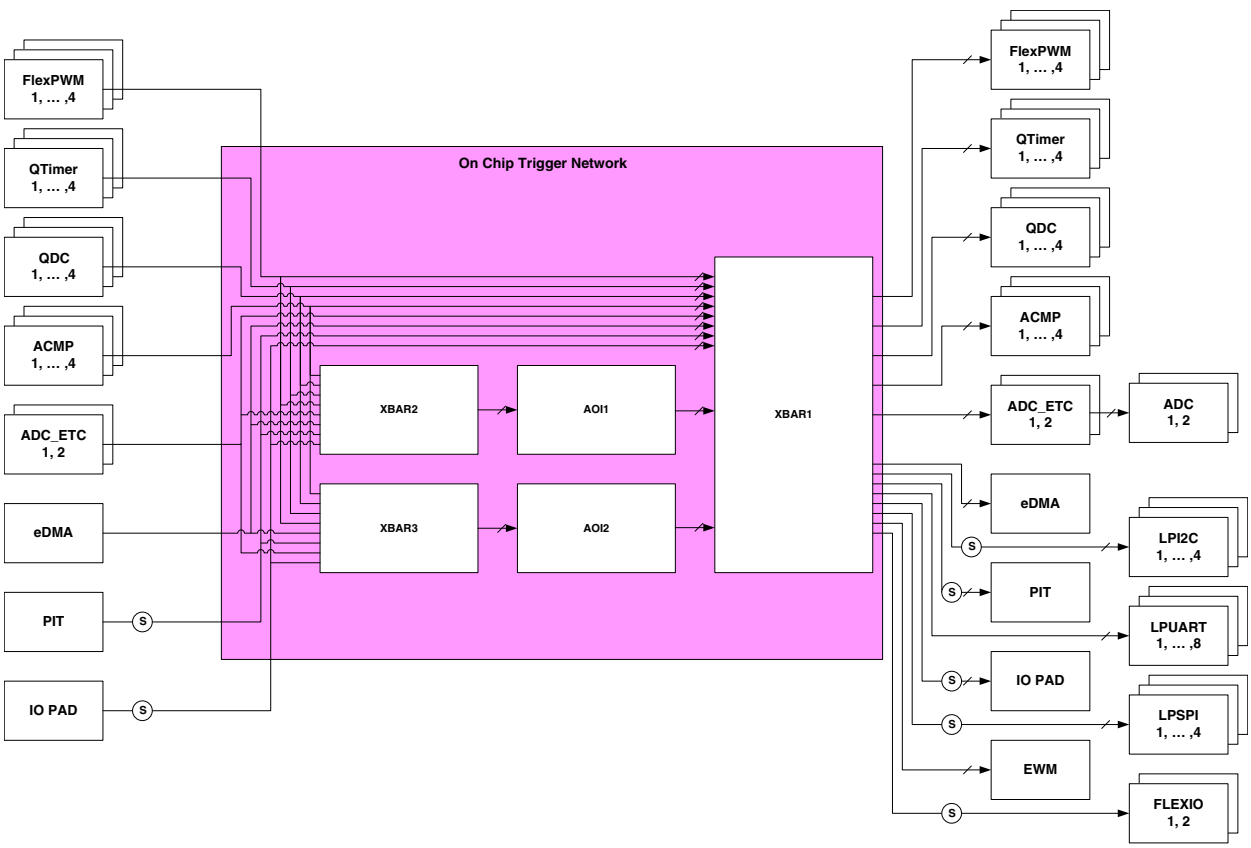


Figure 60-1. On-Chip Cross Trigger Network

60.1.1 Cross BAR (XBAR)

Each crossbar switch is an array of MUXes with shared inputs. Each mux output provides one output of the crossbar. The number of inputs and the number of MUXes/outputs is user configurable and registers are provided to select which of the shared inputs is routed to each output. The crossbar switches are used to reconfigure data paths between peripherals (peripheral output to peripheral input) as well as between peripherals and GPIO.

60.1.2 And-Or-Inverter (AOI)

The AOI module provides an universal Boolean function generator using a four term sum of products expression, with each product term containing true or complement values of the four selected inputs (A, B, C, D).

Chapter 61
Inter-Peripheral Crossbar Switch A (XBARA)

61.1 Chip-specific XBAR information

Table 61-1. Reference links to related information

Topic	Related module(s)	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
Interrupts, DMA Events and XBAR Assignments	-	Interrupts, DMA Events and XBAR Assignments

On this device, for XBAR1 (see the XBARA chapter), the number of inputs is 88, and the number of outputs is 132. For XBAR2 and XBAR3 (see the XBARB chapter), the number of inputs is 64, and the number of outputs is 16.

NOTE

The XBAR_INn and XBAR_OUTn signals (n=4 to 19) share the same IOs. The user needs to configure the corresponding IOMUXC_GPR_GPR6[IOMUXC_XBAR_DIR_SEL_n] bit to use either XBAR_IN or XBAR_OUT.

61.2 Overview

This module implements an array of M N-input combinational muxes. All muxes share the same N inputs in the same order, but each mux has its own independent select field.

The intended application of this module is to provide a flexible crossbar switch function that allows any input (typically from external GPIO or internal module outputs) to be connected to any output (typically to external GPIO or internal module inputs) under user control. This is used to allow user configuration of data paths between internal modules and between internal modules and GPIO.

A subset of the muxes can be configured to support edge detection and either interrupt or DMA request generation based on detected signal edges on the mux output. This allows signal transitions on the signals feeding the crossbar to trigger interrupts or initiate data transfers via DMA into or out of other system modules.

61.2.1 Block Diagram

The block diagram for XBAR is shown in Figure 61-1.

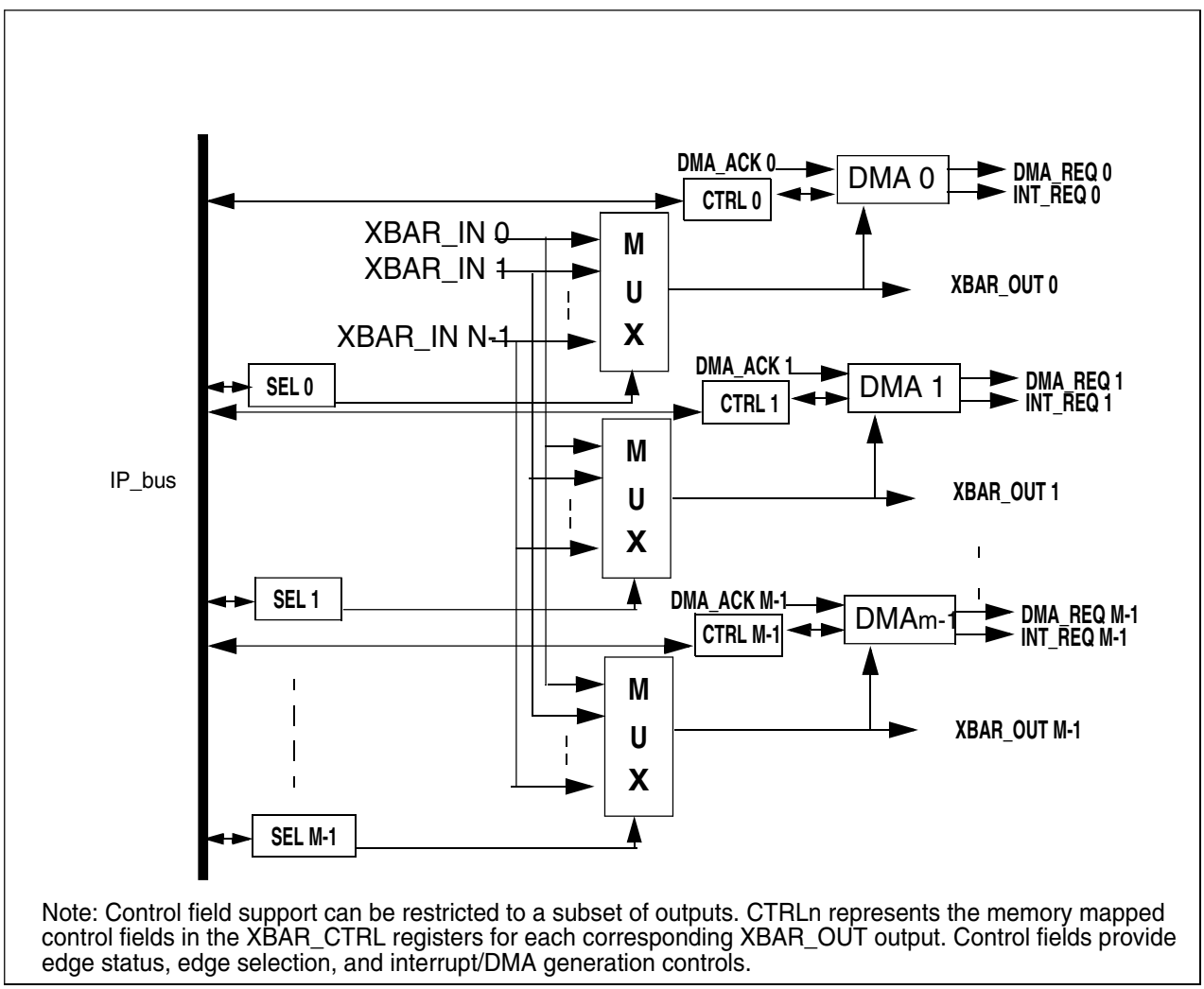


Figure 61-1. XBAR Block Diagram

61.2.2 Features

The XBAR module design includes these distinctive features:

- M identical N-input muxes with individual select fields.
- Edge detection with associated interrupt or DMA request generation for a subset of mux outputs.
- Memory mapped registers with IPBus interface for select and control fields.
- Register write protection input signal.

61.2.3 Modes of Operation

The XBAR module design operates in only a single mode of operation: Functional Mode. The various operation modes are detailed in Functional Mode.

61.3 Functional Description

61.3.1 General

The XBAR module has only one mode of operation, functional mode.

61.3.2 Functional Mode

The value of each mux output is XBAR_OUT[n] = XBAR_IN[SELn]. The SELn select values are configured in the XBAR_SEL registers. All muxes share the same inputs in the same order.

The following is an example to show how to specify the specific input for the specific output:

To select XBAR_IN03 to connect with XBAR_OUT07, according to the index of XBAR_OUT07, select the SEL07 (same number with the selected XBAR_OUT index) fields in the MSBs of the Select Register 3 XBAR_SEL3, and assign the index (0x03, Hexadecimal) of the selected XBAR_IN in it.

```
XBAR_SEL3 &= 0x0011; /*Clear the SEL7 fields
XBAR_SEL3 |= 0x0300; /*Assign the 0x03 to the SEL7 fields
```

A subset of XBAR_OUT[*] outputs has dedicated control fields in a Crossbar Control (XBAR_CTRL) register. Control fields provide the ability to perform edge detection on the corresponding XBAR_OUT output. Edge detection in turn can optionally be used to trigger an interrupt or DMA request. The intention is that, by detecting specified edges on signals propagating through the Crossbar, interrupts or DMA requests can be triggered to perform data transfers to or from other system components.

Control fields include an edge status field (STS), an detected edge type field (EDGE), and interrupt and DMA enable fields (IEN and DEN). STSn is set to 1 when an edge consistent with EDGEn occurs on XBAR_OUT[n]. STSn is cleared by writing 1 to it. Writing 0 as no effect. See Interrupts and DMA Requests for details on the use of STSn for DMA and interrupt request generation.

61.3.3 Clocks

All sequential functionality is controlled by the Bus Clock.

61.3.4 Resets

The XBAR module can be reset by only a hard reset, which forces all registers to their reset state.

61.3.5 Interrupts and DMA Requests

For each XBAR_OUT[*] output with XBAR_CTRL register support, DMA or interrupt functionality can be enabled by setting the corresponding XBAR_CTRL register bit DENn or IENn to 1. DENn and IENn should not be set to 1 at the same time for the same output XBAR_OUT[n].

Setting DENn to 1 enables DMA functionality for XBAR_OUT[n]. When DMA functionality is enabled, the output DMA_REQ[n] reflects the value of STSn. Thus the DMA request asserts when the edge specified by EDGEn is detected on XBAR_OUT[n]. Also, a rising edge on DMA_ACK[n] sets STSn to zero and thus clears the DMA request. When DEN is 0, DMA_REQ[n] is held low and DMA_ACK[n] is ignored.

Setting IENn to 1 enables interrupt functionality for XBAR_OUT[n]. When interrupt functionality is enabled, the output INT_REQ[n] reflects the value of STSn. Thus the interrupt request asserts when the edge specified by EDGEn is detected on XBAR_OUT[n]. The interrupt request is cleared by writing a 1 to STSn. When IENn is 0, INT_REQ[n] is held low.

61.4 External Signals

The following table summarizes the module's external signals.

Table 61-2. Control Signal Properties

Name	I/O Type	Function	Reset State	Notes
XBAR_OUT [0:NUMOUT-1]	O	Mux Outputs with configurable width	XBAR_IN[0]	
XBAR_IN [0:NUMIN-1]	I	Mux Inputs with configurable width	*	
DMA_REQ	O	DMA request	0	
INT_REQ	O	Interrupt request	0	
DMA_ACK	I	DMA acknowledge	0	

At reset, each output XBAR_OUT[*] contains the reset value of the signal driving XBAR_IN[0].

61.4.1 XBAR_OUT[0:NUM_OUT-1] - MUX Outputs

This is a one-dimensional array of the mux outputs. The value on each output XBAR_OUT[n] is determined by the setting of the corresponding memory mapped register SELn such that XBAR_OUT[n] = XBAR_IN[SELn].

61.4.2 XBAR_IN[0:NUM_IN-1] - MUX Inputs

This is a one-dimensional array consisting of the inputs shared by each mux. All muxes share the same inputs in the same order.

61.4.3 DMA_REQ[n] - DMA Request Output(s)

DMA_REQ[n] is a DMA request to the DMA controller.

61.4.4 DMA_ACK[n] - DMA Acknowledge Input(s)

DMA_ACK[n] is a DMA acknowledge input from the DMA controller.

61.4.5 INT_REQ[n] - Interrupt Request Output(s)

INT_REQ[n] is an interrupt request output to the interrupt controller.

61.5 Memory Map and Register Descriptions

The XBAR module has select registers and control registers.

In the XBAR select registers, the SELn fields select which of the shared inputs (XBAR_IN[*]) is muxed to each mux output (XBAR_OUT[*]). There is one SELn field per mux and therefore one per XBAR_OUT output. Crossbar output XBAR_OUT[n] presents the value of XBAR_IN[SELn]. Each select register contains two SELn fields. In the first select register, the LSBs contain the select field for mux 0, and the MSBs contain the select field for mux 1. The pattern repeats in subsequent select registers.

The actual signals connected to XBAR_IN and XBAR_OUT are application specific and are described in the Interrupts, DMA Events, and XBAR Assignments chapter details.

The XBAR control registers configure edge detection, interrupt, and DMA features for a subset of the XBAR_OUT[*] outputs.

XBARA memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
403B_C000	Crossbar A Select Register 0 (XBARA1_SEL0)	16	R/W	0000h	61.5.1/3313
403B_C002	Crossbar A Select Register 1 (XBARA1_SEL1)	16	R/W	0000h	61.5.2/3314
403B_C004	Crossbar A Select Register 2 (XBARA1_SEL2)	16	R/W	0000h	61.5.3/3314
403B_C006	Crossbar A Select Register 3 (XBARA1_SEL3)	16	R/W	0000h	61.5.4/3315
403B_C008	Crossbar A Select Register 4 (XBARA1_SEL4)	16	R/W	0000h	61.5.5/3315
403B_C00A	Crossbar A Select Register 5 (XBARA1_SEL5)	16	R/W	0000h	61.5.6/3316
403B_C00C	Crossbar A Select Register 6 (XBARA1_SEL6)	16	R/W	0000h	61.5.7/3316
403B_C00E	Crossbar A Select Register 7 (XBARA1_SEL7)	16	R/W	0000h	61.5.8/3317
403B_C010	Crossbar A Select Register 8 (XBARA1_SEL8)	16	R/W	0000h	61.5.9/3317
403B_C012	Crossbar A Select Register 9 (XBARA1_SEL9)	16	R/W	0000h	61.5.10/ 3318
403B_C014	Crossbar A Select Register 10 (XBARA1_SEL10)	16	R/W	0000h	61.5.11/ 3318
403B_C016	Crossbar A Select Register 11 (XBARA1_SEL11)	16	R/W	0000h	61.5.12/ 3319

Table continues on the next page...

XBARA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
403B_C018	Crossbar A Select Register 12 (XBARA1_SEL12)	16	R/W	0000h	61.5.13/ 3319
403B_C01A	Crossbar A Select Register 13 (XBARA1_SEL13)	16	R/W	0000h	61.5.14/ 3320
403B_C01C	Crossbar A Select Register 14 (XBARA1_SEL14)	16	R/W	0000h	61.5.15/ 3320
403B_C01E	Crossbar A Select Register 15 (XBARA1_SEL15)	16	R/W	0000h	61.5.16/ 3321
403B_C020	Crossbar A Select Register 16 (XBARA1_SEL16)	16	R/W	0000h	61.5.17/ 3321
403B_C022	Crossbar A Select Register 17 (XBARA1_SEL17)	16	R/W	0000h	61.5.18/ 3322
403B_C024	Crossbar A Select Register 18 (XBARA1_SEL18)	16	R/W	0000h	61.5.19/ 3322
403B_C026	Crossbar A Select Register 19 (XBARA1_SEL19)	16	R/W	0000h	61.5.20/ 3323
403B_C028	Crossbar A Select Register 20 (XBARA1_SEL20)	16	R/W	0000h	61.5.21/ 3323
403B_C02A	Crossbar A Select Register 21 (XBARA1_SEL21)	16	R/W	0000h	61.5.22/ 3324
403B_C02C	Crossbar A Select Register 22 (XBARA1_SEL22)	16	R/W	0000h	61.5.23/ 3324
403B_C02E	Crossbar A Select Register 23 (XBARA1_SEL23)	16	R/W	0000h	61.5.24/ 3325
403B_C030	Crossbar A Select Register 24 (XBARA1_SEL24)	16	R/W	0000h	61.5.25/ 3325
403B_C032	Crossbar A Select Register 25 (XBARA1_SEL25)	16	R/W	0000h	61.5.26/ 3326
403B_C034	Crossbar A Select Register 26 (XBARA1_SEL26)	16	R/W	0000h	61.5.27/ 3326
403B_C036	Crossbar A Select Register 27 (XBARA1_SEL27)	16	R/W	0000h	61.5.28/ 3327
403B_C038	Crossbar A Select Register 28 (XBARA1_SEL28)	16	R/W	0000h	61.5.29/ 3327
403B_C03A	Crossbar A Select Register 29 (XBARA1_SEL29)	16	R/W	0000h	61.5.30/ 3328
403B_C03C	Crossbar A Select Register 30 (XBARA1_SEL30)	16	R/W	0000h	61.5.31/ 3328
403B_C03E	Crossbar A Select Register 31 (XBARA1_SEL31)	16	R/W	0000h	61.5.32/ 3329
403B_C040	Crossbar A Select Register 32 (XBARA1_SEL32)	16	R/W	0000h	61.5.33/ 3329
403B_C042	Crossbar A Select Register 33 (XBARA1_SEL33)	16	R/W	0000h	61.5.34/ 3330

Table continues on the next page...

XBARA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
403B_C044	Crossbar A Select Register 34 (XBARA1_SEL34)	16	R/W	0000h	61.5.35/ 3330
403B_C046	Crossbar A Select Register 35 (XBARA1_SEL35)	16	R/W	0000h	61.5.36/ 3331
403B_C048	Crossbar A Select Register 36 (XBARA1_SEL36)	16	R/W	0000h	61.5.37/ 3331
403B_C04A	Crossbar A Select Register 37 (XBARA1_SEL37)	16	R/W	0000h	61.5.38/ 3332
403B_C04C	Crossbar A Select Register 38 (XBARA1_SEL38)	16	R/W	0000h	61.5.39/ 3332
403B_C04E	Crossbar A Select Register 39 (XBARA1_SEL39)	16	R/W	0000h	61.5.40/ 3333
403B_C050	Crossbar A Select Register 40 (XBARA1_SEL40)	16	R/W	0000h	61.5.41/ 3333
403B_C052	Crossbar A Select Register 41 (XBARA1_SEL41)	16	R/W	0000h	61.5.42/ 3334
403B_C054	Crossbar A Select Register 42 (XBARA1_SEL42)	16	R/W	0000h	61.5.43/ 3334
403B_C056	Crossbar A Select Register 43 (XBARA1_SEL43)	16	R/W	0000h	61.5.44/ 3335
403B_C058	Crossbar A Select Register 44 (XBARA1_SEL44)	16	R/W	0000h	61.5.45/ 3335
403B_C05A	Crossbar A Select Register 45 (XBARA1_SEL45)	16	R/W	0000h	61.5.46/ 3336
403B_C05C	Crossbar A Select Register 46 (XBARA1_SEL46)	16	R/W	0000h	61.5.47/ 3336
403B_C05E	Crossbar A Select Register 47 (XBARA1_SEL47)	16	R/W	0000h	61.5.48/ 3337
403B_C060	Crossbar A Select Register 48 (XBARA1_SEL48)	16	R/W	0000h	61.5.49/ 3337
403B_C062	Crossbar A Select Register 49 (XBARA1_SEL49)	16	R/W	0000h	61.5.50/ 3338
403B_C064	Crossbar A Select Register 50 (XBARA1_SEL50)	16	R/W	0000h	61.5.51/ 3338
403B_C066	Crossbar A Select Register 51 (XBARA1_SEL51)	16	R/W	0000h	61.5.52/ 3339
403B_C068	Crossbar A Select Register 52 (XBARA1_SEL52)	16	R/W	0000h	61.5.53/ 3339
403B_C06A	Crossbar A Select Register 53 (XBARA1_SEL53)	16	R/W	0000h	61.5.54/ 3340
403B_C06C	Crossbar A Select Register 54 (XBARA1_SEL54)	16	R/W	0000h	61.5.55/ 3340
403B_C06E	Crossbar A Select Register 55 (XBARA1_SEL55)	16	R/W	0000h	61.5.56/ 3341

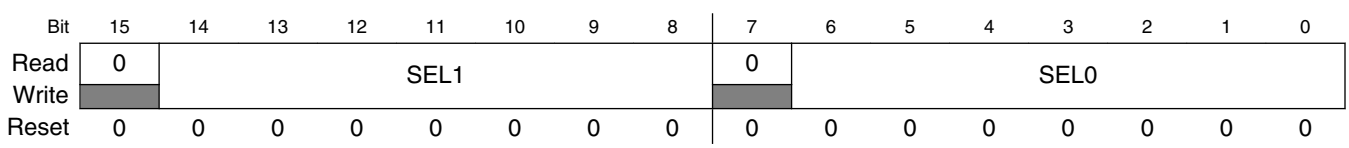
Table continues on the next page...

XBARA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
403B_C070	Crossbar A Select Register 56 (XBARA1_SEL56)	16	R/W	0000h	61.5.57/ 3341
403B_C072	Crossbar A Select Register 57 (XBARA1_SEL57)	16	R/W	0000h	61.5.58/ 3342
403B_C074	Crossbar A Select Register 58 (XBARA1_SEL58)	16	R/W	0000h	61.5.59/ 3342
403B_C076	Crossbar A Select Register 59 (XBARA1_SEL59)	16	R/W	0000h	61.5.60/ 3343
403B_C078	Crossbar A Select Register 60 (XBARA1_SEL60)	16	R/W	0000h	61.5.61/ 3343
403B_C07A	Crossbar A Select Register 61 (XBARA1_SEL61)	16	R/W	0000h	61.5.62/ 3344
403B_C07C	Crossbar A Select Register 62 (XBARA1_SEL62)	16	R/W	0000h	61.5.63/ 3344
403B_C07E	Crossbar A Select Register 63 (XBARA1_SEL63)	16	R/W	0000h	61.5.64/ 3345
403B_C080	Crossbar A Select Register 64 (XBARA1_SEL64)	16	R/W	0000h	61.5.65/ 3345
403B_C082	Crossbar A Select Register 65 (XBARA1_SEL65)	16	R/W	0000h	61.5.66/ 3346
403B_C084	Crossbar A Control Register 0 (XBARA1_CTRL0)	16	R/W	0000h	61.5.67/ 3346
403B_C086	Crossbar A Control Register 1 (XBARA1_CTRL1)	16	R/W	0000h	61.5.68/ 3348

61.5.1 Crossbar A Select Register 0 (XBARAx_SEL0)

Address: 403B_C000h base + 0h offset = 403B_C000h



XBARAx_SEL0 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL1	Input (XBARA_INn) to be muxed to XBARA_OUT1 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL0	Input (XBARA_INn) to be muxed to XBARA_OUT0 (refer to Functional Description section for input/output assignment)

61.5.2 Crossbar A Select Register 1 (XBARA*x*_SEL1)

Address: 403B_C000h base + 2h offset = 403B_C002h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL3							0	SEL2						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA*x*_SEL1 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL3	Input (XBARA_INn) to be muxed to XBARA_OUT3 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL2	Input (XBARA_INn) to be muxed to XBARA_OUT2 (refer to Functional Description section for input/output assignment)

61.5.3 Crossbar A Select Register 2 (XBARA*x*_SEL2)

Address: 403B_C000h base + 4h offset = 403B_C004h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL5							0	SEL4						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA*x*_SEL2 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL5	Input (XBARA_INn) to be muxed to XBARA_OUT5 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL4	Input (XBARA_INn) to be muxed to XBARA_OUT4 (refer to Functional Description section for input/output assignment)

61.5.4 Crossbar A Select Register 3 (XBARA*x*_SEL3)

Address: 403B_C000h base + 6h offset = 403B_C006h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL7							0	SEL6						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA*x*_SEL3 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL7	Input (XBARA_INn) to be muxed to XBARA_OUT7 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL6	Input (XBARA_INn) to be muxed to XBARA_OUT6 (refer to Functional Description section for input/output assignment)

61.5.5 Crossbar A Select Register 4 (XBARA*x*_SEL4)

Address: 403B_C000h base + 8h offset = 403B_C008h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL9							0	SEL8						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA*x*_SEL4 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL9	Input (XBARA_INn) to be muxed to XBARA_OUT9 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL8	Input (XBARA_INn) to be muxed to XBARA_OUT8 (refer to Functional Description section for input/output assignment)

61.5.6 Crossbar A Select Register 5 (XBARA*x*_SEL5)

Address: 403B_C000h base + Ah offset = 403B_C00Ah

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL11							0	SEL10						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA*x*_SEL5 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL11	Input (XBARA_INn) to be muxed to XBARA_OUT11 (refer to Functional Description section for input/ output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL10	Input (XBARA_INn) to be muxed to XBARA_OUT10 (refer to Functional Description section for input/ output assignment)

61.5.7 Crossbar A Select Register 6 (XBARA*x*_SEL6)

Address: 403B_C000h base + Ch offset = 403B_C00Ch

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL13							0	SEL12						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA*x*_SEL6 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL13	Input (XBARA_INn) to be muxed to XBARA_OUT13 (refer to Functional Description section for input/ output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL12	Input (XBARA_INn) to be muxed to XBARA_OUT12 (refer to Functional Description section for input/ output assignment)

61.5.8 Crossbar A Select Register 7 (XBARA*x*_SEL7)

Address: 403B_C000h base + Eh offset = 403B_C00Eh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL15							0	SEL14						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA*x*_SEL7 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL15	Input (XBARA_INn) to be muxed to XBARA_OUT15 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL14	Input (XBARA_INn) to be muxed to XBARA_OUT14 (refer to Functional Description section for input/output assignment)

61.5.9 Crossbar A Select Register 8 (XBARA*x*_SEL8)

Address: 403B_C000h base + 10h offset = 403B_C010h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL17							0	SEL16						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARA*x*_SEL8 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL17	Input (XBARA_INn) to be muxed to XBARA_OUT17 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL16	Input (XBARA_INn) to be muxed to XBARA_OUT16 (refer to Functional Description section for input/output assignment)

61.5.10 Crossbar A Select Register 9 (XBARAx_SEL9)

Address: 403B_C000h base + 12h offset = 403B_C012h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL19							0	SEL18						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL9 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL19	Input (XBARA_INn) to be muxed to XBARA_OUT19 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL18	Input (XBARA_INn) to be muxed to XBARA_OUT18 (refer to Functional Description section for input/output assignment)

61.5.11 Crossbar A Select Register 10 (XBARAx_SEL10)

Address: 403B_C000h base + 14h offset = 403B_C014h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL21							0	SEL20						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL10 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL21	Input (XBARA_INn) to be muxed to XBARA_OUT21 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL20	Input (XBARA_INn) to be muxed to XBARA_OUT20 (refer to Functional Description section for input/output assignment)

61.5.12 Crossbar A Select Register 11 (XBARAx_SEL11)

Address: 403B_C000h base + 16h offset = 403B_C016h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL23							0	SEL22						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL11 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL23	Input (XBARA_INn) to be muxed to XBARA_OUT23 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL22	Input (XBARA_INn) to be muxed to XBARA_OUT22 (refer to Functional Description section for input/output assignment)

61.5.13 Crossbar A Select Register 12 (XBARAx_SEL12)

Address: 403B_C000h base + 18h offset = 403B_C018h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL25							0	SEL24						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL12 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL25	Input (XBARA_INn) to be muxed to XBARA_OUT25 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL24	Input (XBARA_INn) to be muxed to XBARA_OUT24 (refer to Functional Description section for input/output assignment)

61.5.14 Crossbar A Select Register 13 (XBARAx_SEL13)

Address: 403B_C000h base + 1Ah offset = 403B_C01Ah

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL27							0	SEL26						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL13 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL27	Input (XBARA_INn) to be muxed to XBARA_OUT27 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL26	Input (XBARA_INn) to be muxed to XBARA_OUT26 (refer to Functional Description section for input/output assignment)

61.5.15 Crossbar A Select Register 14 (XBARAx_SEL14)

Address: 403B_C000h base + 1Ch offset = 403B_C01Ch

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL29							0	SEL28						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL14 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL29	Input (XBARA_INn) to be muxed to XBARA_OUT29 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL28	Input (XBARA_INn) to be muxed to XBARA_OUT28 (refer to Functional Description section for input/output assignment)

61.5.16 Crossbar A Select Register 15 (XBARAx_SEL15)

Address: 403B_C000h base + 1Eh offset = 403B_C01Eh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL31							0	SEL30						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL15 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL31	Input (XBARA_INn) to be muxed to XBARA_OUT31 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL30	Input (XBARA_INn) to be muxed to XBARA_OUT30 (refer to Functional Description section for input/output assignment)

61.5.17 Crossbar A Select Register 16 (XBARAx_SEL16)

Address: 403B_C000h base + 20h offset = 403B_C020h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL33							0	SEL32						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL16 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL33	Input (XBARA_INn) to be muxed to XBARA_OUT33 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL32	Input (XBARA_INn) to be muxed to XBARA_OUT32 (refer to Functional Description section for input/output assignment)

61.5.18 Crossbar A Select Register 17 (XBARAx_SEL17)

Address: 403B_C000h base + 22h offset = 403B_C022h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL35							0	SEL34						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL17 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL35	Input (XBARA_INn) to be muxed to XBARA_OUT35 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL34	Input (XBARA_INn) to be muxed to XBARA_OUT34 (refer to Functional Description section for input/output assignment)

61.5.19 Crossbar A Select Register 18 (XBARAx_SEL18)

Address: 403B_C000h base + 24h offset = 403B_C024h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL37							0	SEL36						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL18 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL37	Input (XBARA_INn) to be muxed to XBARA_OUT37 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL36	Input (XBARA_INn) to be muxed to XBARA_OUT36 (refer to Functional Description section for input/output assignment)

61.5.20 Crossbar A Select Register 19 (XBARAx_SEL19)

Address: 403B_C000h base + 26h offset = 403B_C026h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL39							0	SEL38						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL19 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL39	Input (XBARA_INn) to be muxed to XBARA_OUT39 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL38	Input (XBARA_INn) to be muxed to XBARA_OUT38 (refer to Functional Description section for input/output assignment)

61.5.21 Crossbar A Select Register 20 (XBARAx_SEL20)

Address: 403B_C000h base + 28h offset = 403B_C028h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL41							0	SEL40						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL20 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL41	Input (XBARA_INn) to be muxed to XBARA_OUT41 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL40	Input (XBARA_INn) to be muxed to XBARA_OUT40 (refer to Functional Description section for input/output assignment)

61.5.22 Crossbar A Select Register 21 (XBARAx_SEL21)

Address: 403B_C000h base + 2Ah offset = 403B_C02Ah

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL43							0	SEL42						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL21 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL43	Input (XBARA_INn) to be muxed to XBARA_OUT43 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL42	Input (XBARA_INn) to be muxed to XBARA_OUT42 (refer to Functional Description section for input/output assignment)

61.5.23 Crossbar A Select Register 22 (XBARAx_SEL22)

Address: 403B_C000h base + 2Ch offset = 403B_C02Ch

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL45							0	SEL44						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL22 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL45	Input (XBARA_INn) to be muxed to XBARA_OUT45 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL44	Input (XBARA_INn) to be muxed to XBARA_OUT44 (refer to Functional Description section for input/output assignment)

61.5.24 Crossbar A Select Register 23 (XBARAx_SEL23)

Address: 403B_C000h base + 2Eh offset = 403B_C02Eh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL47							0	SEL46						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL23 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL47	Input (XBARA_INn) to be muxed to XBARA_OUT47 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL46	Input (XBARA_INn) to be muxed to XBARA_OUT46 (refer to Functional Description section for input/output assignment)

61.5.25 Crossbar A Select Register 24 (XBARAx_SEL24)

Address: 403B_C000h base + 30h offset = 403B_C030h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL49							0	SEL48						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL24 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL49	Input (XBARA_INn) to be muxed to XBARA_OUT49 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL48	Input (XBARA_INn) to be muxed to XBARA_OUT48 (refer to Functional Description section for input/output assignment)

61.5.26 Crossbar A Select Register 25 (XBARAx_SEL25)

Address: 403B_C000h base + 32h offset = 403B_C032h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL51							0	SEL50						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL25 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL51	Input (XBARA_INn) to be muxed to XBARA_OUT51 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL50	Input (XBARA_INn) to be muxed to XBARA_OUT50 (refer to Functional Description section for input/output assignment)

61.5.27 Crossbar A Select Register 26 (XBARAx_SEL26)

Address: 403B_C000h base + 34h offset = 403B_C034h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL53							0	SEL52						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL26 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL53	Input (XBARA_INn) to be muxed to XBARA_OUT53 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL52	Input (XBARA_INn) to be muxed to XBARA_OUT52 (refer to Functional Description section for input/output assignment)

61.5.28 Crossbar A Select Register 27 (XBARAx_SEL27)

Address: 403B_C000h base + 36h offset = 403B_C036h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL55							0	SEL54						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL27 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL55	Input (XBARA_INn) to be muxed to XBARA_OUT55 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL54	Input (XBARA_INn) to be muxed to XBARA_OUT54 (refer to Functional Description section for input/output assignment)

61.5.29 Crossbar A Select Register 28 (XBARAx_SEL28)

Address: 403B_C000h base + 38h offset = 403B_C038h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL57							0	SEL56						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL28 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL57	Input (XBARA_INn) to be muxed to XBARA_OUT57 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL56	Input (XBARA_INn) to be muxed to XBARA_OUT56 (refer to Functional Description section for input/output assignment)

61.5.30 Crossbar A Select Register 29 (XBARAx_SEL29)

Address: 403B_C000h base + 3Ah offset = 403B_C03Ah

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL59							0	SEL58						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL29 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL59	Input (XBARA_INn) to be muxed to XBARA_OUT59 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL58	Input (XBARA_INn) to be muxed to XBARA_OUT58 (refer to Functional Description section for input/output assignment)

61.5.31 Crossbar A Select Register 30 (XBARAx_SEL30)

Address: 403B_C000h base + 3Ch offset = 403B_C03Ch

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL61							0	SEL60						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL30 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL61	Input (XBARA_INn) to be muxed to XBARA_OUT61 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL60	Input (XBARA_INn) to be muxed to XBARA_OUT60 (refer to Functional Description section for input/output assignment)

61.5.32 Crossbar A Select Register 31 (XBARAx_SEL31)

Address: 403B_C000h base + 3Eh offset = 403B_C03Eh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL63							0	SEL62						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL31 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL63	Input (XBARA_INn) to be muxed to XBARA_OUT63 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL62	Input (XBARA_INn) to be muxed to XBARA_OUT62 (refer to Functional Description section for input/output assignment)

61.5.33 Crossbar A Select Register 32 (XBARAx_SEL32)

Address: 403B_C000h base + 40h offset = 403B_C040h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL65							0	SEL64						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL32 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL65	Input (XBARA_INn) to be muxed to XBARA_OUT65 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL64	Input (XBARA_INn) to be muxed to XBARA_OUT64 (refer to Functional Description section for input/output assignment)

61.5.34 Crossbar A Select Register 33 (XBARAx_SEL33)

Address: 403B_C000h base + 42h offset = 403B_C042h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL67							0	SEL66						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL33 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL67	Input (XBARA_INn) to be muxed to XBARA_OUT67 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL66	Input (XBARA_INn) to be muxed to XBARA_OUT66 (refer to Functional Description section for input/output assignment)

61.5.35 Crossbar A Select Register 34 (XBARAx_SEL34)

Address: 403B_C000h base + 44h offset = 403B_C044h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL69							0	SEL68						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL34 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL69	Input (XBARA_INn) to be muxed to XBARA_OUT69 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL68	Input (XBARA_INn) to be muxed to XBARA_OUT68 (refer to Functional Description section for input/output assignment)

61.5.36 Crossbar A Select Register 35 (XBARAx_SEL35)

Address: 403B_C000h base + 46h offset = 403B_C046h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL71							0	SEL70						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL35 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL71	Input (XBARA_INn) to be muxed to XBARA_OUT71 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL70	Input (XBARA_INn) to be muxed to XBARA_OUT70 (refer to Functional Description section for input/output assignment)

61.5.37 Crossbar A Select Register 36 (XBARAx_SEL36)

Address: 403B_C000h base + 48h offset = 403B_C048h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL73							0	SEL72						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL36 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL73	Input (XBARA_INn) to be muxed to XBARA_OUT73 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL72	Input (XBARA_INn) to be muxed to XBARA_OUT72 (refer to Functional Description section for input/output assignment)

61.5.38 Crossbar A Select Register 37 (XBARAx_SEL37)

Address: 403B_C000h base + 4Ah offset = 403B_C04Ah

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL75							0	SEL74						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL37 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL75	Input (XBARA_INn) to be muxed to XBARA_OUT75 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL74	Input (XBARA_INn) to be muxed to XBARA_OUT74 (refer to Functional Description section for input/output assignment)

61.5.39 Crossbar A Select Register 38 (XBARAx_SEL38)

Address: 403B_C000h base + 4Ch offset = 403B_C04Ch

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL77							0	SEL76						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL38 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL77	Input (XBARA_INn) to be muxed to XBARA_OUT77 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL76	Input (XBARA_INn) to be muxed to XBARA_OUT76 (refer to Functional Description section for input/output assignment)

61.5.40 Crossbar A Select Register 39 (XBARAx_SEL39)

Address: 403B_C000h base + 4Eh offset = 403B_C04Eh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL79							0	SEL78						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL39 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL79	Input (XBARA_INn) to be muxed to XBARA_OUT79 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL78	Input (XBARA_INn) to be muxed to XBARA_OUT78 (refer to Functional Description section for input/output assignment)

61.5.41 Crossbar A Select Register 40 (XBARAx_SEL40)

Address: 403B_C000h base + 50h offset = 403B_C050h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL81							0	SEL80						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL40 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL81	Input (XBARA_INn) to be muxed to XBARA_OUT81 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL80	Input (XBARA_INn) to be muxed to XBARA_OUT80 (refer to Functional Description section for input/output assignment)

61.5.42 Crossbar A Select Register 41 (XBARAx_SEL41)

Address: 403B_C000h base + 52h offset = 403B_C052h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL83							0	SEL82						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL41 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL83	Input (XBARA_INn) to be muxed to XBARA_OUT83 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL82	Input (XBARA_INn) to be muxed to XBARA_OUT82 (refer to Functional Description section for input/output assignment)

61.5.43 Crossbar A Select Register 42 (XBARAx_SEL42)

Address: 403B_C000h base + 54h offset = 403B_C054h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL85							0	SEL84						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL42 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL85	Input (XBARA_INn) to be muxed to XBARA_OUT85 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL84	Input (XBARA_INn) to be muxed to XBARA_OUT84 (refer to Functional Description section for input/output assignment)

61.5.44 Crossbar A Select Register 43 (XBARAx_SEL43)

Address: 403B_C000h base + 56h offset = 403B_C056h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL87							0	SEL86						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL43 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL87	Input (XBARA_INn) to be muxed to XBARA_OUT87 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL86	Input (XBARA_INn) to be muxed to XBARA_OUT86 (refer to Functional Description section for input/output assignment)

61.5.45 Crossbar A Select Register 44 (XBARAx_SEL44)

Address: 403B_C000h base + 58h offset = 403B_C058h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL89							0	SEL88						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL44 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL89	Input (XBARA_INn) to be muxed to XBARA_OUT89 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL88	Input (XBARA_INn) to be muxed to XBARA_OUT88 (refer to Functional Description section for input/output assignment)

61.5.46 Crossbar A Select Register 45 (XBARAx_SEL45)

Address: 403B_C000h base + 5Ah offset = 403B_C05Ah

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL91							0	SEL90						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL45 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL91	Input (XBARA_INn) to be muxed to XBARA_OUT91 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL90	Input (XBARA_INn) to be muxed to XBARA_OUT90 (refer to Functional Description section for input/output assignment)

61.5.47 Crossbar A Select Register 46 (XBARAx_SEL46)

Address: 403B_C000h base + 5Ch offset = 403B_C05Ch

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL93							0	SEL92						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL46 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL93	Input (XBARA_INn) to be muxed to XBARA_OUT93 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL92	Input (XBARA_INn) to be muxed to XBARA_OUT92 (refer to Functional Description section for input/output assignment)

61.5.48 Crossbar A Select Register 47 (XBARAx_SEL47)

Address: 403B_C000h base + 5Eh offset = 403B_C05Eh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL95							0	SEL94						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL47 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL95	Input (XBARA_INn) to be muxed to XBARA_OUT95 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL94	Input (XBARA_INn) to be muxed to XBARA_OUT94 (refer to Functional Description section for input/output assignment)

61.5.49 Crossbar A Select Register 48 (XBARAx_SEL48)

Address: 403B_C000h base + 60h offset = 403B_C060h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL97							0	SEL96						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL48 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL97	Input (XBARA_INn) to be muxed to XBARA_OUT97 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL96	Input (XBARA_INn) to be muxed to XBARA_OUT96 (refer to Functional Description section for input/output assignment)

61.5.50 Crossbar A Select Register 49 (XBARAx_SEL49)

Address: 403B_C000h base + 62h offset = 403B_C062h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL99							0	SEL98						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL49 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL99	Input (XBARA_INn) to be muxed to XBARA_OUT99 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL98	Input (XBARA_INn) to be muxed to XBARA_OUT98 (refer to Functional Description section for input/output assignment)

61.5.51 Crossbar A Select Register 50 (XBARAx_SEL50)

Address: 403B_C000h base + 64h offset = 403B_C064h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL101							0	SEL100						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL50 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL101	Input (XBARA_INn) to be muxed to XBARA_OUT101 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL100	Input (XBARA_INn) to be muxed to XBARA_OUT100 (refer to Functional Description section for input/output assignment)

61.5.52 Crossbar A Select Register 51 (XBARAx_SEL51)

Address: 403B_C000h base + 66h offset = 403B_C066h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL103							0	SEL102						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL51 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL103	Input (XBARA_INn) to be muxed to XBARA_OUT103 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL102	Input (XBARA_INn) to be muxed to XBARA_OUT102 (refer to Functional Description section for input/output assignment)

61.5.53 Crossbar A Select Register 52 (XBARAx_SEL52)

Address: 403B_C000h base + 68h offset = 403B_C068h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL105							0	SEL104						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL52 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL105	Input (XBARA_INn) to be muxed to XBARA_OUT105 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL104	Input (XBARA_INn) to be muxed to XBARA_OUT104 (refer to Functional Description section for input/output assignment)

61.5.54 Crossbar A Select Register 53 (XBARAx_SEL53)

Address: 403B_C000h base + 6Ah offset = 403B_C06Ah

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL107							0	SEL106						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL53 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL107	Input (XBARA_INn) to be muxed to XBARA_OUT107 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL106	Input (XBARA_INn) to be muxed to XBARA_OUT106 (refer to Functional Description section for input/output assignment)

61.5.55 Crossbar A Select Register 54 (XBARAx_SEL54)

Address: 403B_C000h base + 6Ch offset = 403B_C06Ch

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL109							0	SEL108						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL54 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL109	Input (XBARA_INn) to be muxed to XBARA_OUT109 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL108	Input (XBARA_INn) to be muxed to XBARA_OUT108 (refer to Functional Description section for input/output assignment)

61.5.56 Crossbar A Select Register 55 (XBARAx_SEL55)

Address: 403B_C000h base + 6Eh offset = 403B_C06Eh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL111							0	SEL110						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL55 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL111	Input (XBARA_INn) to be muxed to XBARA_OUT111 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL110	Input (XBARA_INn) to be muxed to XBARA_OUT110 (refer to Functional Description section for input/output assignment)

61.5.57 Crossbar A Select Register 56 (XBARAx_SEL56)

Address: 403B_C000h base + 70h offset = 403B_C070h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL113							0	SEL112						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL56 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL113	Input (XBARA_INn) to be muxed to XBARA_OUT113 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL112	Input (XBARA_INn) to be muxed to XBARA_OUT112 (refer to Functional Description section for input/output assignment)

61.5.58 Crossbar A Select Register 57 (XBARAx_SEL57)

Address: 403B_C000h base + 72h offset = 403B_C072h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL115							0	SEL114						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL57 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL115	Input (XBARA_INn) to be muxed to XBARA_OUT115 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL114	Input (XBARA_INn) to be muxed to XBARA_OUT114 (refer to Functional Description section for input/output assignment)

61.5.59 Crossbar A Select Register 58 (XBARAx_SEL58)

Address: 403B_C000h base + 74h offset = 403B_C074h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL117							0	SEL116						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL58 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL117	Input (XBARA_INn) to be muxed to XBARA_OUT117 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL116	Input (XBARA_INn) to be muxed to XBARA_OUT116 (refer to Functional Description section for input/output assignment)

61.5.60 Crossbar A Select Register 59 (XBARAx_SEL59)

Address: 403B_C000h base + 76h offset = 403B_C076h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL119							0	SEL118						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL59 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL119	Input (XBARA_INn) to be muxed to XBARA_OUT119 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL118	Input (XBARA_INn) to be muxed to XBARA_OUT118 (refer to Functional Description section for input/output assignment)

61.5.61 Crossbar A Select Register 60 (XBARAx_SEL60)

Address: 403B_C000h base + 78h offset = 403B_C078h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL121							0	SEL120						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL60 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL121	Input (XBARA_INn) to be muxed to XBARA_OUT121 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL120	Input (XBARA_INn) to be muxed to XBARA_OUT120 (refer to Functional Description section for input/output assignment)

61.5.62 Crossbar A Select Register 61 (XBARAx_SEL61)

Address: 403B_C000h base + 7Ah offset = 403B_C07Ah

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL123							0	SEL122						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL61 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL123	Input (XBARA_INn) to be muxed to XBARA_OUT123 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL122	Input (XBARA_INn) to be muxed to XBARA_OUT122 (refer to Functional Description section for input/output assignment)

61.5.63 Crossbar A Select Register 62 (XBARAx_SEL62)

Address: 403B_C000h base + 7Ch offset = 403B_C07Ch

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL125							0	SEL124						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL62 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL125	Input (XBARA_INn) to be muxed to XBARA_OUT125 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL124	Input (XBARA_INn) to be muxed to XBARA_OUT124 (refer to Functional Description section for input/output assignment)

61.5.64 Crossbar A Select Register 63 (XBARAx_SEL63)

Address: 403B_C000h base + 7Eh offset = 403B_C07Eh

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL127							0	SEL126						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL63 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL127	Input (XBARA_INn) to be muxed to XBARA_OUT127 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL126	Input (XBARA_INn) to be muxed to XBARA_OUT126 (refer to Functional Description section for input/output assignment)

61.5.65 Crossbar A Select Register 64 (XBARAx_SEL64)

Address: 403B_C000h base + 80h offset = 403B_C080h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL129							0	SEL128						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL64 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL129	Input (XBARA_INn) to be muxed to XBARA_OUT129 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL128	Input (XBARA_INn) to be muxed to XBARA_OUT128 (refer to Functional Description section for input/output assignment)

61.5.66 Crossbar A Select Register 65 (XBARAx_SEL65)

Address: 403B_C000h base + 82h offset = 403B_C082h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0	SEL131							0	SEL130						
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARAx_SEL65 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
14–8 SEL131	Input (XBARA_INn) to be muxed to XBARA_OUT131 (refer to Functional Description section for input/output assignment)
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL130	Input (XBARA_INn) to be muxed to XBARA_OUT130 (refer to Functional Description section for input/output assignment)

61.5.67 Crossbar A Control Register 0 (XBARAx_CTRL0)

Use this register to configure edge detection, interrupt, and DMA features for the XBAR_OUT0 and XBAR_OUT1 outputs.

The XBAR_CTRL registers are organized similarly to the XBAR_SEL registers, with control fields for two XBAR_OUT outputs in each register. In control register 0, the LSBs contain the control fields for XBAR_OUT0, and the MSBs contain the control fields for XBAR_OUT1.

Address: 403B_C000h base + 84h offset = 403B_C084h

Bit	15	14	13	12	11	10	9	8
Read	0			STS1	EDGE1		IEN1	DEN1
Write				w1c				
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	0			STS0	EDGE0		IEN0	DEN0
Write				w1c				
Reset	0	0	0	0	0	0	0	0

XBARAx_CTRL0 field descriptions

Field	Description
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 STS1	Edge detection status for XBAR_OUT1 This bit reflects the results of edge detection for XBAR_OUT1. This field is set to 1 when an edge consistent with the current setting of EDGE1 is detected on XBAR_OUT1. This field is cleared by writing 1 to it or by a DMA_ACK1 reception when DEN1 is set. Writing 0 to the field has no effect. When interrupt or DMA functionality is enabled for XBAR_OUT1, this field is 1 when the interrupt or DMA request is asserted and 0 when the interrupt or DMA request has been cleared. 0 Active edge not yet detected on XBAR_OUT1 1 Active edge detected on XBAR_OUT1
11–10 EDGE1	Active edge for edge detection on XBAR_OUT1 This field selects which edges on XBAR_OUT1 cause STS1 to assert. 00 STS1 never asserts 01 STS1 asserts on rising edges of XBAR_OUT1 10 STS1 asserts on falling edges of XBAR_OUT1 11 STS1 asserts on rising and falling edges of XBAR_OUT1
9 IEN1	Interrupt Enable for XBAR_OUT1 This bit enables the interrupt function on the corresponding XBAR_OUT1 output. When the interrupt is enabled, the output INT_REQ1 reflects the value STS1. When the interrupt is disabled, INT_REQ1 remains low. The interrupt request is cleared by writing a 1 to STS1. **Restriction:** IEN1 and DEN1 should not both be set to 1. 0 Interrupt disabled 1 Interrupt enabled
8 DEN1	DMA Enable for XBAR_OUT1 This bit enables the DMA function on the corresponding XBAR_OUT1 output. When enabled, DMA_REQ1 presents the value STS1. When disabled, the DMA_REQ1 output remains low. **Restriction:** IEN1 and DEN1 should not both be set to 1. 0 DMA disabled 1 DMA enabled
7–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 STS0	Edge detection status for XBAR_OUT0 This bit reflects the results of edge detection for XBAR_OUT0. This field is set to 1 when an edge consistent with the current setting of EDGE0 is detected on XBAR_OUT0. This field is cleared by writing 1 to it or by a DMA_ACK0 reception when DEN0 is set. Writing 0 to the field has no effect. When interrupt or DMA functionality is enabled for XBAR_OUT0, this field is 1 when the interrupt or DMA request is asserted and 0 when the interrupt or DMA request has been cleared.

Table continues on the next page...

XBARAx_CTRL0 field descriptions (continued)

Field	Description
	0 Active edge not yet detected on XBAR_OUT0 1 Active edge detected on XBAR_OUT0
3–2 EDGE0	Active edge for edge detection on XBAR_OUT0 This field selects which edges on XBAR_OUT0 cause STS0 to assert. 00 STS0 never asserts 01 STS0 asserts on rising edges of XBAR_OUT0 10 STS0 asserts on falling edges of XBAR_OUT0 11 STS0 asserts on rising and falling edges of XBAR_OUT0
1 IEN0	Interrupt Enable for XBAR_OUT0 This bit enables the interrupt function on the corresponding XBAR_OUT0 output. When the interrupt is enabled, the output INT_REQ0 reflects the value STS0. When the interrupt is disabled, INT_REQ0 remains low. The interrupt request is cleared by writing a 1 to STS0. **Restriction:** IEN0 and DEN0 should not both be set to 1. 0 Interrupt disabled 1 Interrupt enabled
0 DEN0	DMA Enable for XBAR_OUT0 This bit enables the DMA function on the corresponding XBAR_OUT0 output. When enabled, DMA_REQ0 presents the value STS0. When disabled, the DMA_REQ0 output remains low. **Restriction:** IEN0 and DEN0 should not both be set to 1. 0 DMA disabled 1 DMA enabled

61.5.68 Crossbar A Control Register 1 (XBARAx_CTRL1)

Use this register to configure edge detection, interrupt, and DMA features for the XBAR_OUT2 and XBAR_OUT3 outputs.

The XBAR_CTRL registers are organized similarly to the XBAR_SEL registers, with control fields for two XBAR_OUT outputs in each register. In control register 1, the LSBs contain the control fields for XBAR_OUT2, and the MSBs contain the control fields for XBAR_OUT3.

Address: 403B_C000h base + 86h offset = 403B_C086h

Bit	15	14	13	12	11	10	9	8
Read	0			STS3	EDGE3		IEN3	DEN3
Write				w1c				
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
Read	0			STS2	EDGE2		IEN2	DEN2
Write				w1c				
Reset	0	0	0	0	0	0	0	0

XBARAx_CTRL1 field descriptions

Field	Description
15–13 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
12 STS3	Edge detection status for XBAR_OUT3 This bit reflects the results of edge detection for XBAR_OUT3. This field is set to 1 when an edge consistent with the current setting of EDGE3 is detected on XBAR_OUT3. This field is cleared by writing 1 to it or by a DMA_ACK3 reception when DEN3 is set. Writing 0 to the field has no effect. When interrupt or DMA functionality is enabled for XBAR_OUT3, this field is 1 when the interrupt or DMA request is asserted and 0 when the interrupt or DMA request has been cleared. 0 Active edge not yet detected on XBAR_OUT3 1 Active edge detected on XBAR_OUT3
11–10 EDGE3	Active edge for edge detection on XBAR_OUT3 This field selects which edges on XBAR_OUT3 cause STS3 to assert. 00 STS3 never asserts 01 STS3 asserts on rising edges of XBAR_OUT3 10 STS3 asserts on falling edges of XBAR_OUT3 11 STS3 asserts on rising and falling edges of XBAR_OUT3
9 IEN3	Interrupt Enable for XBAR_OUT3 This bit enables the interrupt function on the corresponding XBAR_OUT3 output. When the interrupt is enabled, the output INT_REQ3 reflects the value STS3. When the interrupt is disabled, INT_REQ3 remains low. The interrupt request is cleared by writing a 1 to STS3. **Restriction:** IEN3 and DEN3 should not both be set to 1. 0 Interrupt disabled 1 Interrupt enabled
8 DEN3	DMA Enable for XBAR_OUT3 This bit enables the DMA function on the corresponding XBAR_OUT3 output. When enabled, DMA_REQ3 presents the value STS3. When disabled, the DMA_REQ3 output remains low. **Restriction:** IEN3 and DEN3 should not both be set to 1. 0 DMA disabled 1 DMA enabled
7–5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 STS2	Edge detection status for XBAR_OUT2 This bit reflects the results of edge detection for XBAR_OUT2.

Table continues on the next page...

XBARAx_CTRL1 field descriptions (continued)

Field	Description
	This field is set to 1 when an edge consistent with the current setting of EDGE2 is detected on XBAR_OUT2. This field is cleared by writing 1 to it or by a DMA_ACK2 reception when DEN2 is set. Writing 0 to the field has no effect. When interrupt or DMA functionality is enabled for XBAR_OUT2, this field is 1 when the interrupt or DMA request is asserted and 0 when the interrupt or DMA request has been cleared. 0 Active edge not yet detected on XBAR_OUT2 1 Active edge detected on XBAR_OUT2
3–2 EDGE2	Active edge for edge detection on XBAR_OUT2 This field selects which edges on XBAR_OUT2 cause STS2 to assert. 00 STS2 never asserts 01 STS2 asserts on rising edges of XBAR_OUT2 10 STS2 asserts on falling edges of XBAR_OUT2 11 STS2 asserts on rising and falling edges of XBAR_OUT2
1 IEN2	Interrupt Enable for XBAR_OUT2 This bit enables the interrupt function on the corresponding XBAR_OUT2 output. When the interrupt is enabled, the output INT_REQ2 reflects the value STS2. When the interrupt is disabled, INT_REQ2 remains low. The interrupt request is cleared by writing a 1 to STS2. **Restriction:** IEN2 and DEN2 should not both be set to 1. 0 Interrupt disabled 1 Interrupt enabled
0 DEN2	DMA Enable for XBAR_OUT2 This bit enables the DMA function on the corresponding XBAR_OUT2 output. When enabled, DMA_REQ2 presents the value STS2. When disabled, the DMA_REQ2 output remains low. **Restriction:** IEN2 and DEN2 should not both be set to 1. 0 DMA disabled 1 DMA enabled

Chapter 62
Inter-Peripheral Crossbar Switch B (XBARB)

62.1 Chip-specific XBAR information

Table 62-1. Reference links to related information

Topic	Related module(s)	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
Interrupts, DMA Events and XBAR Assignments	-	Interrupts, DMA Events and XBAR Assignments

On this device, for XBAR1 (see the XBARA chapter), the number of inputs is 88, and the number of outputs is 132. For XBAR2 and XBAR3 (see the XBARB chapter), the number of inputs is 64, and the number of outputs is 16.

NOTE

The XBAR_INn and XBAR_OUTn signals (n=4 to 19) share the same IOs. The user needs to configure the corresponding IOMUXC_GPR_GPR6[IOMUXC_XBAR_DIR_SEL_n] bit to use either XBAR_IN or XBAR_OUT.

62.2 Overview

For a description of the general features and functionality of this module, refer to the XBARA description.

62.3 Functional Description

For a functional description of this module, refer to the XBARA functional description.

62.4 External Signals

The following table summarizes the module's external signals.

Table 62-2. Control Signal Properties

Name	I/O Type	Function	Reset State	Notes
XBAR_OUT [0:NUMOUT-1]	O	Mux Outputs with configurable width	XBAR_IN[0]	
XBAR_IN [0:NUMIN-1]	I	Mux Inputs with configurable width	*	

At reset, each output XBAR_OUT[*] contains the reset value of the signal driving XBAR_IN[0].

62.4.1 XBAR_OUT[0:NUM_OUT-1] - MUX Outputs

This is a one-dimensional array of the mux outputs. The value on each output XBAR_OUT[n] is determined by the setting of the corresponding memory mapped register SELn such that XBAR_OUT[n] = XBAR_IN[SELn].

62.4.2 XBAR_IN[0:NUM_IN-1] - MUX Inputs

This is a one-dimensional array consisting of the inputs shared by each mux. All muxes share the same inputs in the same order.

62.5 Memory Map and Register Descriptions

This section provides information about the XBARB instance of the inter-peripheral crossbar switch.

This XBAR module has only select registers.

In the XBAR select registers, the SELn fields select which of the shared inputs (XBAR_IN[*]) is muxed to each mux output (XBAR_OUT[*]). There is one SELn field per mux and therefore one per XBAR_OUT output. Crossbar output XBAR_OUT[n] presents the value of XBAR_IN[SELn]. Each select register contains two SELn fields. In the first select register, the LSBs contain the select field for mux 0, and the MSBs contain the select field for mux 1. The pattern repeats in subsequent select registers.

The actual signals connected to XBAR_IN and XBAR_OUT are application specific and are described in the Interrupts, DMA Events, and XBAR Assignments chapter details.

XBARB memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
403C_0000	Crossbar B Select Register 0 (XBARB2_SEL0)	16	R/W	0000h	62.5.1/3353
403C_0002	Crossbar B Select Register 1 (XBARB2_SEL1)	16	R/W	0000h	62.5.2/3354
403C_0004	Crossbar B Select Register 2 (XBARB2_SEL2)	16	R/W	0000h	62.5.3/3354
403C_0006	Crossbar B Select Register 3 (XBARB2_SEL3)	16	R/W	0000h	62.5.4/3355
403C_0008	Crossbar B Select Register 4 (XBARB2_SEL4)	16	R/W	0000h	62.5.5/3355
403C_000A	Crossbar B Select Register 5 (XBARB2_SEL5)	16	R/W	0000h	62.5.6/3356
403C_000C	Crossbar B Select Register 6 (XBARB2_SEL6)	16	R/W	0000h	62.5.7/3356
403C_000E	Crossbar B Select Register 7 (XBARB2_SEL7)	16	R/W	0000h	62.5.8/3357
403C_4000	Crossbar B Select Register 0 (XBARB3_SEL0)	16	R/W	0000h	62.5.1/3353
403C_4002	Crossbar B Select Register 1 (XBARB3_SEL1)	16	R/W	0000h	62.5.2/3354
403C_4004	Crossbar B Select Register 2 (XBARB3_SEL2)	16	R/W	0000h	62.5.3/3354
403C_4006	Crossbar B Select Register 3 (XBARB3_SEL3)	16	R/W	0000h	62.5.4/3355
403C_4008	Crossbar B Select Register 4 (XBARB3_SEL4)	16	R/W	0000h	62.5.5/3355
403C_400A	Crossbar B Select Register 5 (XBARB3_SEL5)	16	R/W	0000h	62.5.6/3356
403C_400C	Crossbar B Select Register 6 (XBARB3_SEL6)	16	R/W	0000h	62.5.7/3356
403C_400E	Crossbar B Select Register 7 (XBARB3_SEL7)	16	R/W	0000h	62.5.8/3357

62.5.1 Crossbar B Select Register 0 (XBARBx_SEL0)

Address: Base address + 0h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL1						0		SEL0					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARBx_SEL0 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL1	Input (XBARB_INn) to be muxed to XBARB_OUT1 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL0	Input (XBARB_INn) to be muxed to XBARB_OUT0 (refer to Functional Description section for input/output assignment)

62.5.2 Crossbar B Select Register 1 (XBARBx_SEL1)

Address: Base address + 2h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL3						0		SEL2					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARBx_SEL1 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL3	Input (XBARB_INn) to be muxed to XBARB_OUT3 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL2	Input (XBARB_INn) to be muxed to XBARB_OUT2 (refer to Functional Description section for input/output assignment)

62.5.3 Crossbar B Select Register 2 (XBARBx_SEL2)

Address: Base address + 4h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL5						0		SEL4					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARBx_SEL2 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

XBARBx_SEL2 field descriptions (continued)

Field	Description
13–8 SEL5	Input (XBARB_INn) to be muxed to XBARB_OUT5 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL4	Input (XBARB_INn) to be muxed to XBARB_OUT4 (refer to Functional Description section for input/output assignment)

62.5.4 Crossbar B Select Register 3 (XBARBx_SEL3)

Address: Base address + 6h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL7						0		SEL6					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARBx_SEL3 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL7	Input (XBARB_INn) to be muxed to XBARB_OUT7 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL6	Input (XBARB_INn) to be muxed to XBARB_OUT6 (refer to Functional Description section for input/output assignment)

62.5.5 Crossbar B Select Register 4 (XBARBx_SEL4)

Address: Base address + 8h offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL9						0		SEL8					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARBx_SEL4 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL9	Input (XBARB_INn) to be muxed to XBARB_OUT9 (refer to Functional Description section for input/output assignment)

Table continues on the next page...

XBARBx_SEL4 field descriptions (continued)

Field	Description
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL8	Input (XBARB_INn) to be muxed to XBARB_OUT8 (refer to Functional Description section for input/output assignment)

62.5.6 Crossbar B Select Register 5 (XBARBx_SEL5)

Address: Base address + Ah offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL11						0		SEL10					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARBx_SEL5 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL11	Input (XBARB_INn) to be muxed to XBARB_OUT11 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL10	Input (XBARB_INn) to be muxed to XBARB_OUT10 (refer to Functional Description section for input/output assignment)

62.5.7 Crossbar B Select Register 6 (XBARBx_SEL6)

Address: Base address + Ch offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL13						0		SEL12					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARBx_SEL6 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL13	Input (XBARB_INn) to be muxed to XBARB_OUT13 (refer to Functional Description section for input/output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

XBARBx_SEL6 field descriptions (continued)

Field	Description
SEL12	Input (XBARB_INn) to be muxed to XBARB_OUT12 (refer to Functional Description section for input/ output assignment)

62.5.8 Crossbar B Select Register 7 (XBARBx_SEL7)

Address: Base address + Eh offset

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read	0		SEL15						0		SEL14					
Write																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

XBARBx_SEL7 field descriptions

Field	Description
15–14 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
13–8 SEL15	Input (XBARB_INn) to be muxed to XBARB_OUT15 (refer to Functional Description section for input/ output assignment)
7–6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
SEL14	Input (XBARB_INn) to be muxed to XBARB_OUT14 (refer to Functional Description section for input/ output assignment)

Chapter 63
And-Or-Inverter (AOI)

63.1 Chip-specific AOI information

Table 63-1. Reference links to related information

Topic	Related module(s)	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
XBAR resource assignments	XBAR	XBAR Resource Assignments

63.2 Overview

The AND/OR/INVERT module (known simply as the AOI module) supports the generation of a configurable number of EVENT signals. Each output EVENTn is a configurable and/or/invert function of four associated AOI inputs: An, Bn, Cn, and Dn.

This module is designed to be integrated in conjuction with one or more inter-peripheral crossbar switch (XBAR) modules. A crossbar switch is typically used to select the 4*n AOI inputs from among available peripheral outputs and GPIO signals. The n EVENTn outputs from the AOI module are typically used as additional inputs to a second crossbar switch, adding to it the ability to connect to its outputs an arbitrary 4-input boolean function of its other inputs.

The AOI controller is a slave peripheral module connecting event input indicators from a variety of device modules and generating event output signals that can be routed to an inter-peripheral crossbar switch or other peripherals. Its programming model is accessed through the standard IPS (Sky Blue) slave interface. The module is designed to be very configurable in terms of the functionality of its integrated AOI functions.

63.2.1 Block diagram

The AOI module supports a configurable number of event outputs, where each event output represents a user-programmed combinational boolean function based on four event inputs. The key features of this module include:

- Four dedicated inputs for each event output
- User-programmable combinational boolean function evaluation for each event output
- Memory-mapped device connected to a slave peripheral (IPS) bus
- Configurable number of event outputs

NOTE

The connections from the AOI module outputs to other functions is SoC-specific.

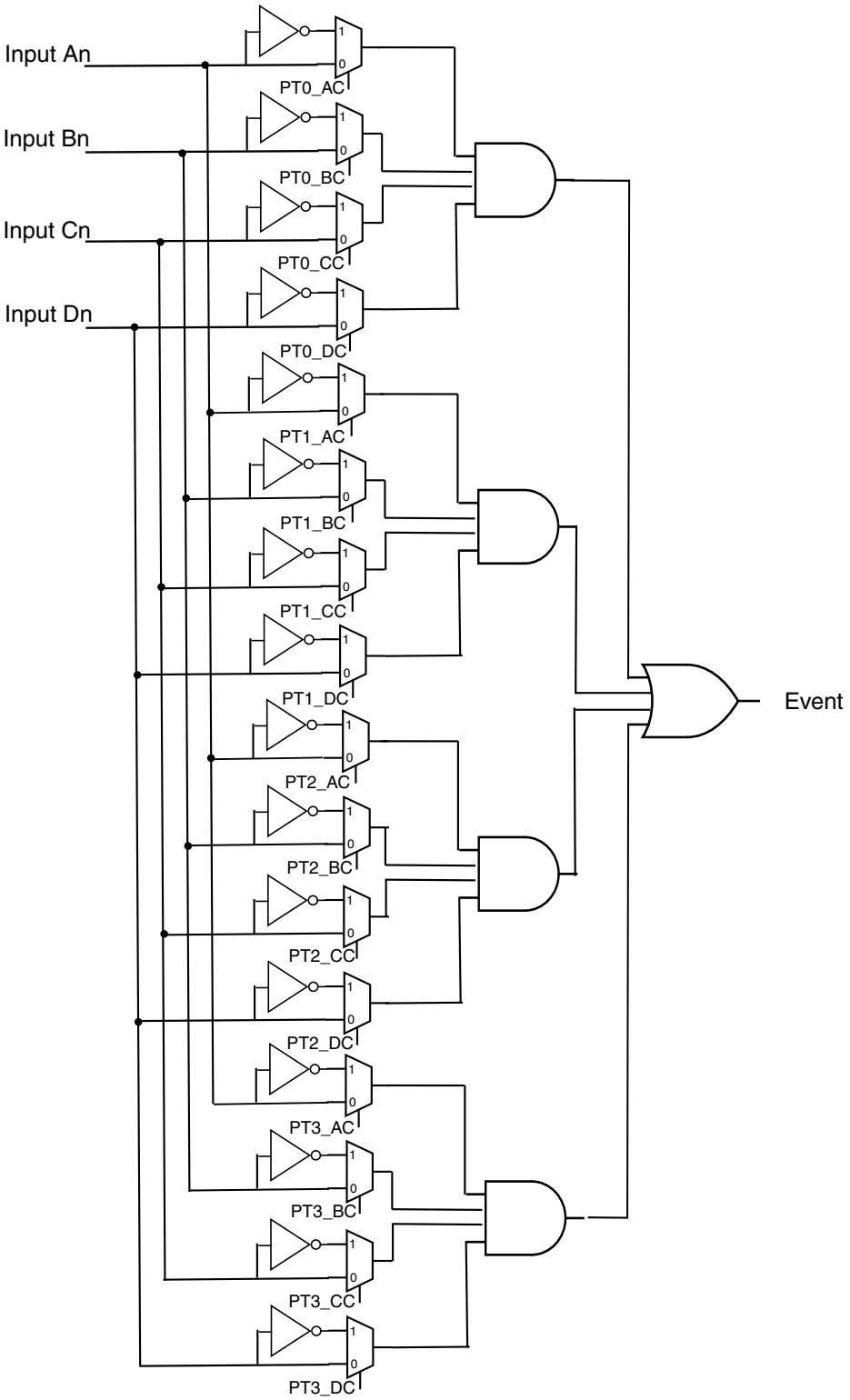


Figure 63-1. Simplified AOI Block Diagram

63.2.2 Features

The major features of the AOI module are summarized below:

- Highly programmable module for creating combinational boolean events for use as hardware triggers
  - Each channel has four event inputs and one output
  - Evaluates a combinational boolean expression as the sum of four products where each product term includes all four selected input sources available as true or complement values
  - Event output is formed as purely combinational logic and operates as a hardware trigger
- Memory-mapped device connected to the slave peripheral (IPS) bus
  - Programming model organized per channel for simplified software

63.2.3 Modes of Operation

The AOI module does not support any special modes of operation. As shown in Figure 63-1, its operation is primarily controlled by the selected event inputs and outputs. Additionally, as a memory-mapped device located on the slave peripheral bus, it responds based strictly on memory address for accesses to its programming model.

The AOI module resides in the slave peripheral *bus clock domain*.

63.3 Functional Description

The AOI is a highly programmable module for creating combinational boolean outputs for use as hardware triggers. Each AOI output channel, as shown in Figure 63-1, has one logic function:

- Evaluation of a combinational boolean expression as a sum of four products where each product term includes all four selected input sources available as true or complement values

A typical application of the AOI module is to be integrated with one or more inter-peripheral crossbar switch modules as illustrated in the following figure. The 20 external inputs are shared by two crossbar switch modules. The crossbar switch on the top is used to select the inputs to four 4-input AOI functions in the AOI module. The outputs of these four AOI functions are output from the AOI module and are added to the original 20 external inputs to provide a total of 24 inputs to the bottom crossbar switch. As a result, the bottom crossbar can not only direct any of the original 20 external inputs to any of its outputs, it can also now direct any one of four 4-input AOI functions of those external inputs to any of its outputs.

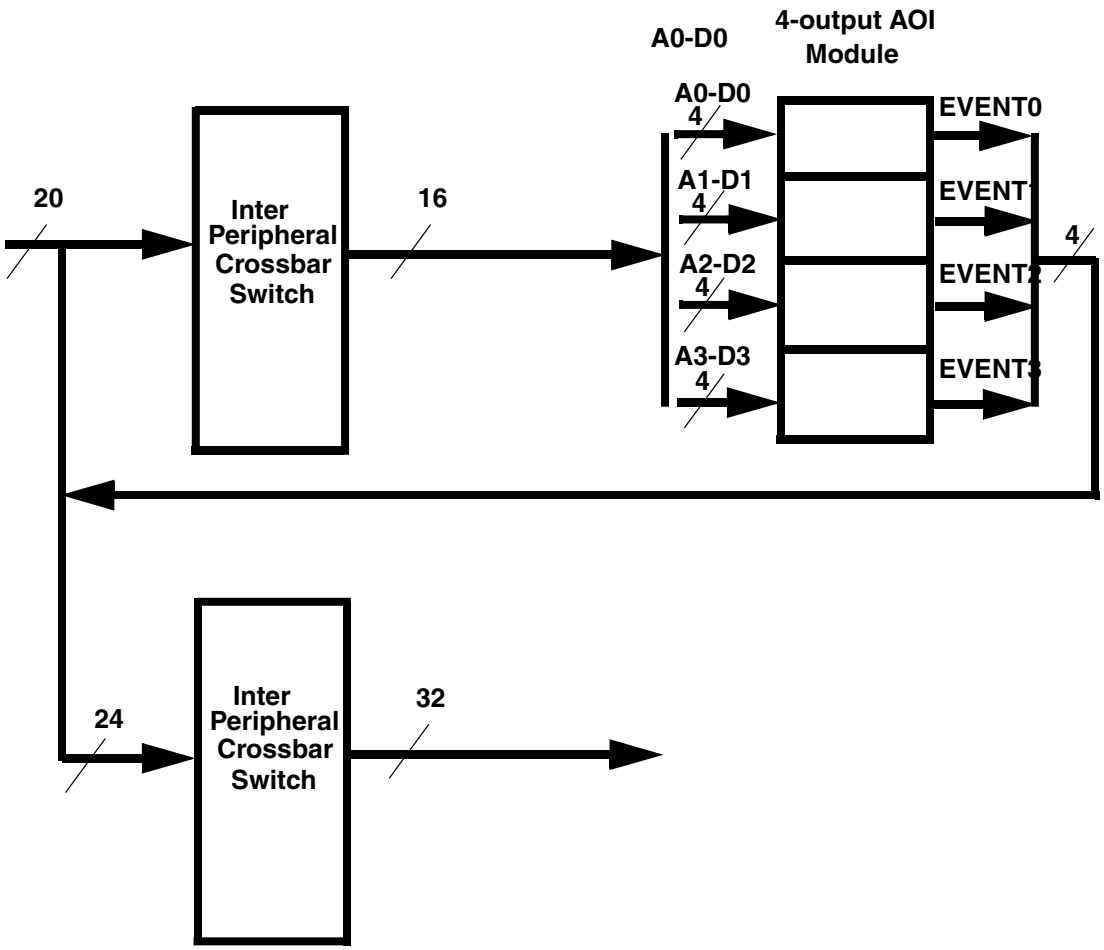


Figure 63-2. Integration Example of AOI with two Inter-Peripheral Crossbar Switches

63.3.1 Configuration Examples for the Boolean Function Evaluation

This section presents examples of the programming model configuration for simple boolean expressions.

The AOI module provides a universal boolean function generator using a four-term sum of products expression with each product term containing true or complement values of the four selected event inputs (A, B, C, D). Specifically, the event output is defined by the following “4 x 4” boolean expression:

```
EVENTn
    = (0,An,~An,1) & (0,Bn,~Bn,1) & (0,Cn,~Cn,1) & (0,Dn,~Dn,1)// product term 0
    | (0,An,~An,1) & (0,Bn,~Bn,1) & (0,Cn,~Cn,1) & (0,Dn,~Dn,1)// product term 1
```

```
    | (0,An,~An,1) & (0,Bn,~Bn,1) & (0,Cn,~Cn,1) & (0,Dn,~Dn,1)// product term 2
    | (0,An,~An,1) & (0,Bn,~Bn,1) & (0,Cn,~Cn,1) & (0,Dn,~Dn,1)// product term 3
```

where each selected input term in each product term can be configured to produce a logical 0 or 1 or pass the true or complement of the selected event input. Each product term uses eight bits of configuration information, two bits for each of the four selected event inputs. The actual boolean expression implemented in each channel is:

```
EVENTn
    = (PT0_AC[0] & An |  PT0_AC[1] & ~An)// product term 0
                      & (PT0_BC[0] &  Bn | PTO_BC[1] & ~Bn)
                      & (PT0_CC[0] &  Cn | PTO_CC[1] & ~Cn)
                      & (PT0_DC[0] &  Dn | PTO_DC[1] & ~Dn)

    | (PT1_AC[0] & An |  PT1_AC[1] & ~An)// product term 1
                      & (PT1_BC[0] &  Bn | PT1_BC[1] & ~Bn)
                      & (PT1_CC[0] &  Cn | PT1_CC[1] & ~Cn)
                      & (PT1_DC[0] &  Dn | PT1_DC[1] & ~Dn)

    | (PT2_AC[0] & An |  PT2_AC[1] & ~An)// product term 2
                      & (PT2_BC[0] &  Bn | PT2_BC[1] & ~Bn)
                      & (PT2_CC[0] &  Cn | PT2_CC[1] & ~Cn)
                      & (PT2_DC[0] &  Dn | PT2_DC[1] & ~Dn)

    | (PT3_AC[0] & An |  PT3_AC[1] & ~An)// product term 3
                      & (PT3_BC[0] &  Bn | PT3_BC[1] & ~Bn)
                      & (PT3_CC[0] &  Cn | PT3_CC[1] & ~Cn)
                      & (PT3_DC[0] &  Dn | PT3_DC[1] & ~Dn)
```

where the bits of the combined {BFCRT01n,BFCRT23n} registers correspond to the `PT{0-3}_{A,B,C,D}C[1:0]` terms in the equation.

Consider the settings of the combined 32-bit {BFCRT01n,BFCRT23n} registers for several simple boolean expressions as shown in Table 63-2.

Table 63-2. BFCRTn Values for Simple Boolean Expressions

Event Output Expression	PT0	PT1	PT2	PT3	{BFCRT01, BFCRT23}
A & B	A & B	0	0	0	01011111_00000000_00000000_00000000
A & B & C	A & B & C	0	0	0	01010111_00000000_00000000_00000000
(A & B & C) + D	A & B & C	D	0	0	01010111_11111101_00000000_00000000
A + B + C + D	A	B	C	D	01111111_11011111_11110111_11111101
(A & ~B) + (~A & B)	A & ~B	~A & B	0	0	01101111_10011111_00000000_00000000

As can be seen in these examples, the resulting logic provides a simple yet powerful boolean function evaluation for defining an event output.

63.3.2 AOI Timing Between Inputs and Outputs

Each EVENTn output of the AOI module is a combination function of its four dedicated inputs An, Bn, Cn, and Dn. Propagation through the AOI and any associated inter-peripheral crossbar switch modules is intended to be single bus clock cycle.

63.3.3 Clocks

The bus clock is used for register accesses by the system, but AOI functions themselves are asynchronous and do not use a clock.

63.4 External Signal Description

The AOI module does not directly support any external interfaces. There may be package input signals (indirectly) connected to the module as event inputs, but since the *AOI does not include any input synchronization hardware*, this function must be handled before the event input signals are routed into the module.

63.5 Memory Map and Register Descriptions

The AOI module supports access to its programming model via a 16-bit peripheral bus connection. The module is designed to support 16-bit accesses only. Functionality for accesses of other widths is undefined.

63.5.1 AOI register descriptions

The AOI module supports a specific number of event outputs. Each output EVENTn outputs a four-term AOI function of four binary inputs: An, Bn, Cn, and Dn. A pair of 16-bit registers configures this four-term AOI function: The two registers BFCRT01n and BFCRT23n define the configuration for the evaluation of the Boolean function defining EVENTn, where *n* is the event output channel number. The BFCRT01n register defines the configuration of product terms 0 and 1, and the BFCRT23n register defines the configuration of product terms 2 and 3.

The AOI module provides a universal Boolean function generator using a four-term sum of products expression with each product term containing true or complement values of the four selected event inputs (An, Bn, Cn, Dn). Specifically, the EVENTn output is defined by the following "4 x 4" Boolean expression:

```
EVENTn
    = (0,An,~An,1) & (0,Bn,~Bn,1) & (0,Cn,~Cn,1) & (0,Dn,~Dn,1)// product term 0
    | (0,An,~An,1) & (0,Bn,~Bn,1) & (0,Cn,~Cn,1) & (0,Dn,~Dn,1)// product term 1
    | (0,An,~An,1) & (0,Bn,~Bn,1) & (0,Cn,~Cn,1) & (0,Dn,~Dn,1)// product term 2
    | (0,An,~An,1) & (0,Bn,~Bn,1) & (0,Cn,~Cn,1) & (0,Dn,~Dn,1)// product term 3
```

where each selected input of each product term can be configured to produce a logical 0 or 1 or pass the true or complement of the selected event input. Each product term uses 8 bits of configuration information, 2 bits for each of the four selected event inputs. The resulting logic provides a simple yet powerful Boolean function evaluation for defining an event output.

These AOI functions are combinational in nature and are intended to be sampled and used synchronously.

63.5.1.1 AOI memory map

AOI base address: 403B_4000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Boolean Function Term 0 and 1 Configuration Register for EVENTn (BFCRT010)	16	RW	0000h
2h	Boolean Function Term 2 and 3 Configuration Register for EVENTn (BFCRT230)	16	RW	0000h
4h	Boolean Function Term 0 and 1 Configuration Register for EVENTn (BFCRT011)	16	RW	0000h
6h	Boolean Function Term 2 and 3 Configuration Register for EVENTn (BFCRT231)	16	RW	0000h
8h	Boolean Function Term 0 and 1 Configuration Register for EVENTn (BFCRT012)	16	RW	0000h
Ah	Boolean Function Term 2 and 3 Configuration Register for EVENTn (BFCRT232)	16	RW	0000h
Ch	Boolean Function Term 0 and 1 Configuration Register for EVENTn (BFCRT013)	16	RW	0000h
Eh	Boolean Function Term 2 and 3 Configuration Register for EVENTn (BFCRT233)	16	RW	0000h

63.5.1.2 Boolean Function Term 0 and 1 Configuration Register for EVENTn (BFCRT010 - BFCRT013)

63.5.1.2.1 Offset

Register	Offset
BFCRT010	0h
BFCRT011	4h
BFCRT012	8h
BFCRT013	Ch

63.5.1.2.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PT0_AC		PT0_BC		PT0_CC		PT0_DC		PT1_AC		PT1_BC		PT1_CC		PT1_DC	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

63.5.1.2.3 Fields

Field	Function
15-14 PT0_AC	Product term 0, A input configuration This 2-bit field defines the Boolean evaluation associated with the selected input A in product term 0. 00b - Force the A input in this product term to a logical zero 01b - Pass the A input in this product term 10b - Complement the A input in this product term 11b - Force the A input in this product term to a logical one
13-12 PT0_BC	Product term 0, B input configuration This 2-bit field defines the Boolean evaluation associated with the selected input B in product term 0. 00b - Force the B input in this product term to a logical zero 01b - Pass the B input in this product term 10b - Complement the B input in this product term 11b - Force the B input in this product term to a logical one
11-10 PT0_CC	Product term 0, C input configuration This 2-bit field defines the Boolean evaluation associated with the selected input C in product term 0. 00b - Force the C input in this product term to a logical zero 01b - Pass the C input in this product term 10b - Complement the C input in this product term 11b - Force the C input in this product term to a logical one
9-8 PT0_DC	Product term 0, D input configuration This 2-bit field defines the Boolean evaluation associated with the selected input D in product term 0. 00b - Force the D input in this product term to a logical zero

Table continues on the next page...

Field	Function
	01b - Pass the D input in this product term 10b - Complement the D input in this product term 11b - Force the D input in this product term to a logical one
7-6 PT1_AC	Product term 1, A input configuration This 2-bit field defines the Boolean evaluation associated with the selected input A in product term 1. 00b - Force the A input in this product term to a logical zero 01b - Pass the A input in this product term 10b - Complement the A input in this product term 11b - Force the A input in this product term to a logical one
5-4 PT1_BC	Product term 1, B input configuration This 2-bit field defines the Boolean evaluation associated with the selected input B in product term 1. 00b - Force the B input in this product term to a logical zero 01b - Pass the B input in this product term 10b - Complement the B input in this product term 11b - Force the B input in this product term to a logical one
3-2 PT1_CC	Product term 1, C input configuration This 2-bit field defines the Boolean evaluation associated with the selected input C in product term 1. 00b - Force the C input in this product term to a logical zero 01b - Pass the C input in this product term 10b - Complement the C input in this product term 11b - Force the C input in this product term to a logical one
1-0 PT1_DC	Product term 1, D input configuration This 2-bit field defines the Boolean evaluation associated with the selected input D in product term 1. 00b - Force the D input in this product term to a logical zero 01b - Pass the D input in this product term 10b - Complement the D input in this product term 11b - Force the D input in this product term to a logical one

63.5.1.3 Boolean Function Term 2 and 3 Configuration Register for EVENTn (BFCRT230 - BFCRT233)

63.5.1.3.1 Offset

Register	Offset
BFCRT230	2h
BFCRT231	6h
BFCRT232	Ah
BFCRT233	Eh

63.5.1.3.2 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	PT2_AC		PT2_BC		PT2_CC		PT2_DC		PT3_AC		PT3_BC		PT3_CC		PT3_DC	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

63.5.1.3.3 Fields

Field	Function
15-14 PT2_AC	Product term 2, A input configuration This 2-bit field defines the Boolean evaluation associated with the selected input A in product term 2. 00b - Force the A input in this product term to a logical zero 01b - Pass the A input in this product term 10b - Complement the A input in this product term 11b - Force the A input in this product term to a logical one
13-12 PT2_BC	Product term 2, B input configuration This 2-bit field defines the Boolean evaluation associated with the selected input B in product term 2. 00b - Force the B input in this product term to a logical zero 01b - Pass the B input in this product term 10b - Complement the B input in this product term 11b - Force the B input in this product term to a logical one
11-10 PT2_CC	Product term 2, C input configuration This 2-bit field defines the Boolean evaluation associated with the selected input C in product term 2. 00b - Force the C input in this product term to a logical zero 01b - Pass the C input in this product term 10b - Complement the C input in this product term 11b - Force the C input in this product term to a logical one
9-8 PT2_DC	Product term 2, D input configuration This 2-bit field defines the Boolean evaluation associated with the selected input D in product term 2. 00b - Force the D input in this product term to a logical zero 01b - Pass the D input in this product term 10b - Complement the D input in this product term 11b - Force the D input in this product term to a logical one
7-6 PT3_AC	Product term 3, A input configuration This 2-bit field defines the Boolean evaluation associated with the selected input A in product term 3. 00b - Force the A input in this product term to a logical zero 01b - Pass the A input in this product term 10b - Complement the A input in this product term 11b - Force the A input in this product term to a logical one
5-4 PT3_BC	Product term 3, B input configuration This 2-bit field defines the Boolean evaluation associated with the selected input B in product term 3. 00b - Force the B input in this product term to a logical zero 01b - Pass the B input in this product term 10b - Complement the B input in this product term 11b - Force the B input in this product term to a logical one
3-2	Product term 3, C input configuration

Table continues on the next page...

Field	Function
PT3_CC	This 2-bit field defines the Boolean evaluation associated with the selected input C in product term 3. 00b - Force the C input in this product term to a logical zero 01b - Pass the C input in this product term 10b - Complement the C input in this product term 11b - Force the C input in this product term to a logical one
1-0	Product term 3, D input configuration
PT3_DC	This 2-bit field defines the Boolean evaluation associated with the selected input D in product term 3. 00b - Force the D input in this product term to a logical zero 01b - Pass the D input in this product term 10b - Complement the D input in this product term 11b - Force the D input in this product term to a logical one

Chapter 64
Analog Overview

64.1 Overview

The following analog modules are integrated in this chip:

- Analog-Digital-Converter (ADC) - Supports up to 1MS/s sampling rate and up to 16 single-ended external analog inputs
- Touch Screen Controller (TSC) - Responsible for providing control of the ADC and the touch screen analog block to form a touch screen system
- ADC External Trigger Control (ADC_ETC) - Enables multiple users to share a ADC module in a Time-Division-Multiplexing (TDM) method
- Comparator (CMP) - Operational over the entire supply range and features a integrated 6-bit DAC and a integrated 8 to 1 channel mux

Below is the simplified block diagram:

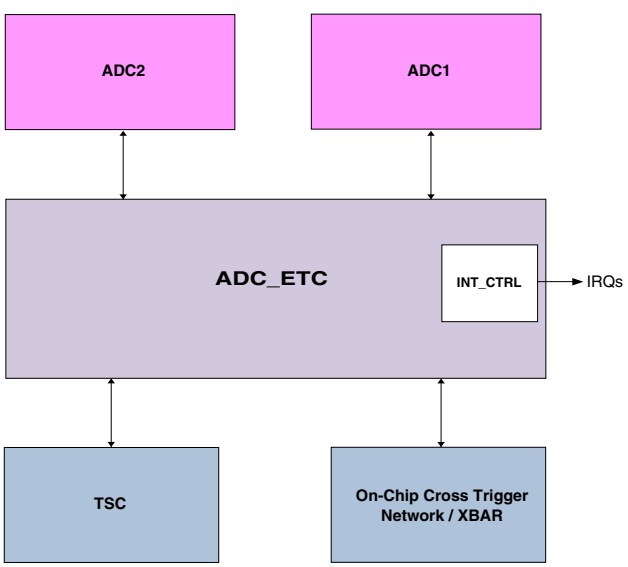


Figure 64-1. Block Diagram (Simplified)

64.1.1 Analog-to-Digital Converter (ADC)

The ADC module features a linear successive approximation algorithm with up to 12-bit resolution with 10/11 bit accuracy.

It has three possible states:

- Disabled State - The ADC module is disabled
- Idle state - The ADC module is idle, waiting on a conversion to be initiated
- Conversion State - In this mode, the ADC module can perform analog-to-digital conversion on any of the software selectable channels, using the successive approximation algorithm

64.1.2 Touch Screen Control (TSC)

The TSC helps achieve the function of touch detection and touch location detection. The controller utilizes the ADC hardware trigger function and control switches in the touch screen analog block to form a touch screen system . The controller only supports 4-wire or 5-wire screen touch modes.

64.1.3 ADC External Trigger Control (ADC_ETC)

The ADC_ETC module enables multiple users to share a ADC module. It manages the external trigger mode (HWT) of the ADCs and supports up to x8 HWT for each ADC. It is capable of triggering dual ADCs in SyncMode (controlled by the same trigger source) or AsyncMode (controlled by separate trigger source). Every ADC HWT can be configured with a Back-to-Back (B2B) and Interrupt Enable (IE) settings and each external trigger can be configured with a fixed priority.

64.1.4 Comparator (CMP)

The analog comparator (CMP) allows for comparing two analog input voltages while the Analog MUX allows for selecting an analog input signal from eight channels. The CMP is operational across the full range of the supply voltage. The integrated 6-bit DAC provides a selectable voltage reference for applications where voltage reference is needed.

The CMPs share the same 3.3V reference voltage with the ADCs.

Chapter 65
Analog Comparator (ACMP)

65.1 Chip-specific CMP information

Table 65-1. Reference links to related information

Topic	Related module(s)	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
Interrupts, DMA Events and XBAR Assignments	-	Interrupts, DMA Events and XBAR Assignments

NOTE

V_{in1} and V_{in2} are all connected to V_{DDA}, on this device.

65.2 Introduction

The comparator (CMP) module provides a circuit for comparing two analog input voltages. The comparator circuit is designed to operate across the full range of the supply voltage, known as rail-to-rail operation.

The Analog MUX (ANMUX) provides a circuit for selecting an analog input signal from eight channels. One signal is provided by the 6-bit digital-to-analog converter (DAC). The mux circuit is designed to operate across the full range of the supply voltage.

The 6-bit DAC is 64-tap resistor ladder network which provides a selectable voltage reference for applications where voltage reference is needed. The 64-tap resistor ladder network divides the supply reference V_{in} into 64 voltage levels. A 6-bit digital signal

input selects the output voltage level, which varies from V_{in} to $V_{in}/64$. V_{in} can be selected from two voltage sources, V_{in1} and V_{in2}. The 6-bit DAC from a comparator is available as an on-chip internal signal only and is not available externally to a pin.

65.2.1 CMP features

The CMP has the following features:

- Operational over the entire supply range
- Inputs may range from rail to rail
- Programmable hysteresis control
- Selectable interrupt on rising-edge, falling-edge, or both rising or falling edges of the comparator output
- Selectable inversion on comparator output
- Capability to produce a wide range of outputs such as:
  - Sampled
  - Windowed, which is ideal for certain PWM zero-crossing-detection applications
  - Digitally filtered:
    - Filter can be bypassed
    - Can be clocked via external SAMPLE signal or scaled bus clock
- External hysteresis can be used at the same time that the output filter is used for internal functions
- Two software selectable performance levels:
  - Shorter propagation delay at the expense of higher power
  - Low power, with longer propagation delay
- DMA transfer support
  - A comparison event can be selected to trigger a DMA transfer
- Functional in all modes of operation
- The window and filter functions are not available in the following modes:
  - Stop
  - VLPS

65.2.2 6-bit DAC key features

The 6-bit DAC has the following features:

- 6-bit resolution
- Selectable supply reference source
- Power Down mode to conserve power when not in use
- Option to route the output to internal comparator input

65.2.3 ANMUX key features

The ANMUX has the following features:

- Two 8-to-1 channel mux
- Operational over the entire supply range

65.2.4 CMP, DAC and ANMUX diagram

The following figure shows the block diagram for the High-Speed Comparator, DAC, and ANMUX modules.

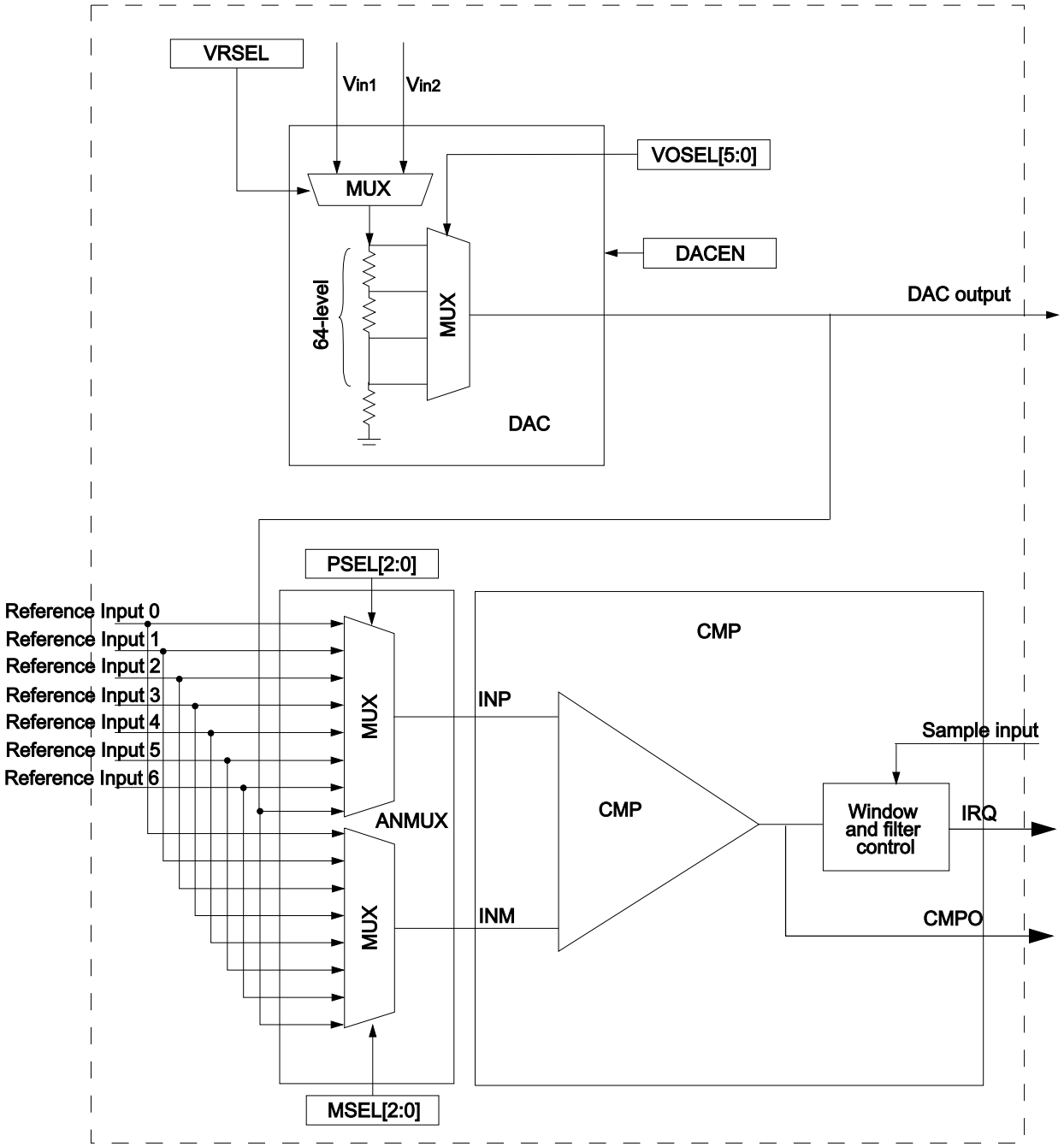


Figure 65-1. CMP, DAC and ANMUX block diagram

65.2.5 CMP block diagram

The following figure shows the block diagram for the CMP module.

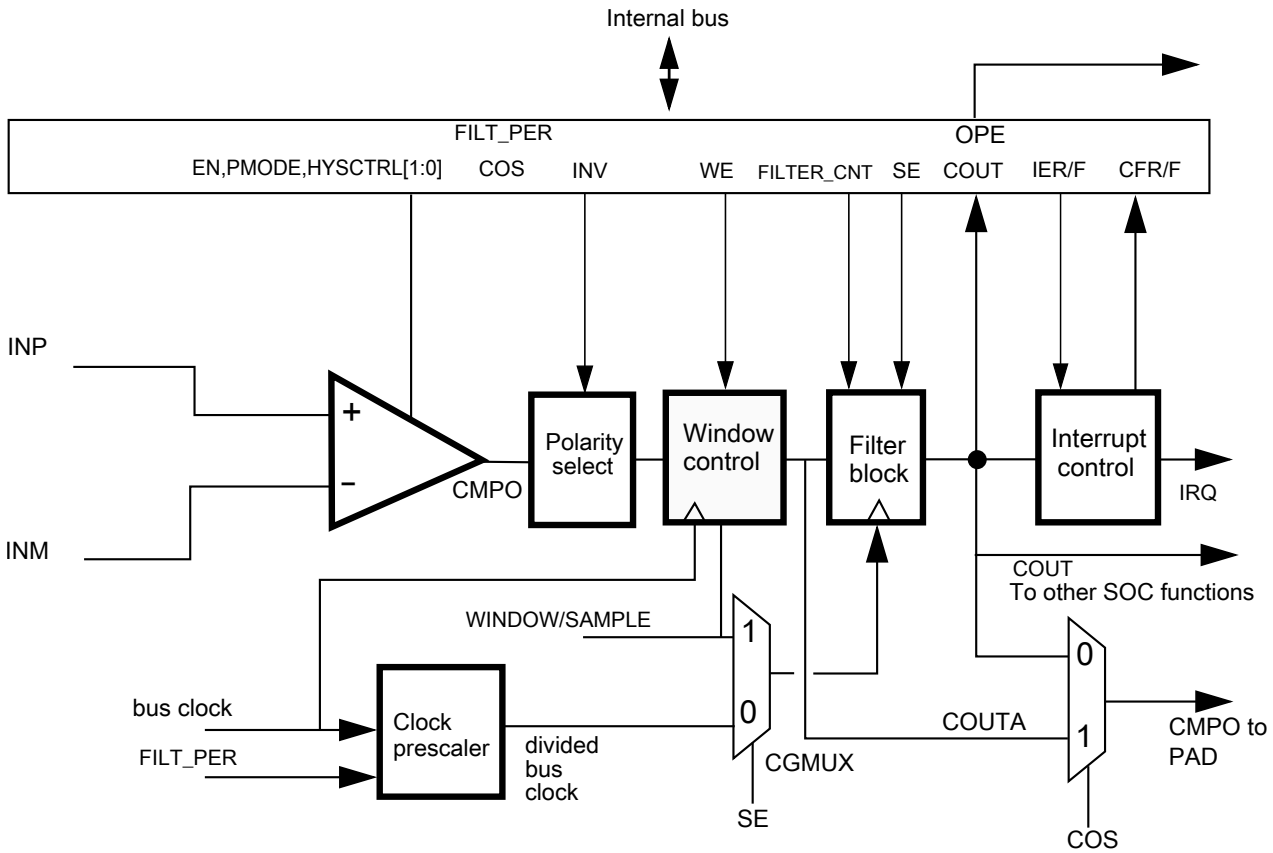


Figure 65-2. Comparator module block diagram

In the CMP block diagram:

- The Window Control block is bypassed when CR1[WE] = 0
- If CR1[WE] = 1, the comparator output will be sampled on every bus clock when WINDOW=1 to generate COUTA. Sampling does NOT occur when WINDOW = 0.
- The Filter block is bypassed when not in use.
- The Filter block acts as a simple sampler if the filter is bypassed and CR0[FILTER_CNT] is set to 0x01.
- The Filter block filters based on multiple samples when the filter is bypassed and CR0[FILTER_CNT] is set greater than 0x01.
  - If CR1[SE] = 1, the external SAMPLE input is used as sampling clock
  - If CR1[SE] = 0, the divided bus clock is used as sampling clock

- If enabled, the Filter block will incur up to one bus clock additional latency penalty on COUT due to the fact that COUT, which is crossing clock domain boundaries, must be resynchronized to the bus clock.
- CR1[WE] and CR1[SE] are mutually exclusive.

65.3 Memory map/register definitions

CMP memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4009_4000	CMP Control Register 0 (CMP1_CR0)	8	R/W	00h	65.3.1/3379
4009_4001	CMP Control Register 1 (CMP1_CR1)	8	R/W	00h	65.3.2/3380
4009_4002	CMP Filter Period Register (CMP1_FPR)	8	R/W	00h	65.3.3/3381
4009_4003	CMP Status and Control Register (CMP1_SCR)	8	R/W	00h	65.3.4/3381
4009_4004	DAC Control Register (CMP1_DACCR)	8	R/W	00h	65.3.5/3383
4009_4005	MUX Control Register (CMP1_MUXCR)	8	R/W	00h	65.3.6/3383
4009_4008	CMP Control Register 0 (CMP2_CR0)	8	R/W	00h	65.3.1/3379
4009_4009	CMP Control Register 1 (CMP2_CR1)	8	R/W	00h	65.3.2/3380
4009_400A	CMP Filter Period Register (CMP2_FPR)	8	R/W	00h	65.3.3/3381
4009_400B	CMP Status and Control Register (CMP2_SCR)	8	R/W	00h	65.3.4/3381
4009_400C	DAC Control Register (CMP2_DACCR)	8	R/W	00h	65.3.5/3383
4009_400D	MUX Control Register (CMP2_MUXCR)	8	R/W	00h	65.3.6/3383
4009_4010	CMP Control Register 0 (CMP3_CR0)	8	R/W	00h	65.3.1/3379
4009_4011	CMP Control Register 1 (CMP3_CR1)	8	R/W	00h	65.3.2/3380
4009_4012	CMP Filter Period Register (CMP3_FPR)	8	R/W	00h	65.3.3/3381
4009_4013	CMP Status and Control Register (CMP3_SCR)	8	R/W	00h	65.3.4/3381
4009_4014	DAC Control Register (CMP3_DACCR)	8	R/W	00h	65.3.5/3383
4009_4015	MUX Control Register (CMP3_MUXCR)	8	R/W	00h	65.3.6/3383
4009_4018	CMP Control Register 0 (CMP4_CR0)	8	R/W	00h	65.3.1/3379
4009_4019	CMP Control Register 1 (CMP4_CR1)	8	R/W	00h	65.3.2/3380
4009_401A	CMP Filter Period Register (CMP4_FPR)	8	R/W	00h	65.3.3/3381
4009_401B	CMP Status and Control Register (CMP4_SCR)	8	R/W	00h	65.3.4/3381
4009_401C	DAC Control Register (CMP4_DACCR)	8	R/W	00h	65.3.5/3383
4009_401D	MUX Control Register (CMP4_MUXCR)	8	R/W	00h	65.3.6/3383

65.3.1 CMP Control Register 0 (CMPx_CR0)

Address: Base address + 0h offset

Bit	7	6	5	4	3	2	1	0
Read	0	FILTER_CNT			0	0	HYSTCTR	
Write								
Reset	0	0	0	0	0	0	0	0

CMPx_CR0 field descriptions

Field	Description
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6–4 FILTER_CNT	Filter Sample Count Represents the number of consecutive samples that must agree prior to the comparator ouput filter accepting a new output state. For information regarding filter programming and latency, see the Functional description. 000 Filter is disabled. If SE = 1, then COUT is a logic 0. This is not a legal state, and is not recommended. If SE = 0, COUT = COUTA. 001 One sample must agree. The comparator output is simply sampled. 010 2 consecutive samples must agree. 011 3 consecutive samples must agree. 100 4 consecutive samples must agree. 101 5 consecutive samples must agree. 110 6 consecutive samples must agree. 111 7 consecutive samples must agree.
3 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
2 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
HYSTCTR	Comparator hard block hysteresis control Defines the programmable hysteresis level. The hysteresis values associated with each level are device-specific. See the Data Sheet of the device for the exact values. 00 Level 0 01 Level 1 10 Level 2 11 Level 3

65.3.2 CMP Control Register 1 (CMPx_CR1)

Address: Base address + 1h offset

Bit	7	6	5	4	3	2	1	0
Read	SE	WE	0	PMODE	INV	COS	OPE	EN
Write								
Reset	0	0	0	0	0	0	0	0

CMPx_CR1 field descriptions

Field	Description
7 SE	Sample Enable At any given time, either SE or WE can be set. If a write to this register attempts to set both, then SE is set and WE is cleared. However, avoid writing 1s to both field locations because this "11" case is reserved and may change in future implementations. 0 Sampling mode is not selected. 1 Sampling mode is selected.
6 WE	Windowing Enable At any given time, either SE or WE can be set. If a write to this register attempts to set both, then SE is set and WE is cleared. However, avoid writing 1s to both field locations because this "11" case is reserved and may change in future implementations. 0 Windowing mode is not selected. 1 Windowing mode is selected.
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 PMODE	Power Mode Select See the electrical specifications table in the device Data Sheet for details. 0 Low-Speed (LS) Comparison mode selected. In this mode, CMP has slower output propagation delay and lower current consumption. 1 High-Speed (HS) Comparison mode selected. In this mode, CMP has faster output propagation delay and higher current consumption.
3 INV	Comparator INVERT Allows selection of the polarity of the analog comparator function. It is also driven to the COUT output, on both the device pin and as SCR[COUT], when OPE=0. 0 Does not invert the comparator output. 1 Inverts the comparator output.
2 COS	Comparator Output Select 0 Set the filtered comparator output (CMPO) to equal COUT. 1 Set the unfiltered comparator output (CMPO) to equal COUTA.
1 OPE	Comparator Output Pin Enable

Table continues on the next page...

CMPx_CR1 field descriptions (continued)

Field	Description
	0 CMPO is not available on the associated CMPO output pin. If the comparator does not own the pin, this field has no effect. 1 CMPO is available on the associated CMPO output pin. The comparator output (CMPO) is driven out on the associated CMPO output pin if the comparator owns the pin. If the comparator does not own the field, this bit has no effect.
0 EN	Comparator Module Enable Enables the Analog Comparator module. When the module is not enabled, it remains in the off state, and consumes no power. When the user selects the same input from analog mux to the positive and negative port, the comparator is disabled automatically. 0 Analog Comparator is disabled. 1 Analog Comparator is enabled.

65.3.3 CMP Filter Period Register (CMPx_FPR)

Address: Base address + 2h offset

Bit	7	6	5	4	3	2	1	0
Read Write	FILT_PER							
Reset	0	0	0	0	0	0	0	0

CMPx_FPR field descriptions

Field	Description
FILT_PER	Filter Sample Period Specifies the sampling period, in bus clock cycles, of the comparator output filter, when CR1[SE]=0. Setting FILT_PER to 0x0 disables the filter. Filter programming and latency details appear in the Functional description. This field has no effect when CR1[SE]=1. In that case, the external SAMPLE signal is used to determine the sampling period.

65.3.4 CMP Status and Control Register (CMPx_SCR)

Address: Base address + 3h offset

Bit	7	6	5	4	3	2	1	0
Read	0	DMAEN	0	IER	IEF	CFR	CFF	COUT
Write						w1c	w1c	
Reset	0	0	0	0	0	0	0	0

CMPx_SCR field descriptions

Field	Description
7 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
6 DMAEN	DMA Enable Control Enables the DMA transfer triggered from the CMP module. When this field is set, a DMA request is asserted when CFR or CFF is set. 0 DMA is disabled. 1 DMA is enabled.
5 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
4 IER	Comparator Interrupt Enable Rising Enables the CFR interrupt from the CMP. When this field is set, an interrupt will be asserted when CFR is set. 0 Interrupt is disabled. 1 Interrupt is enabled.
3 IEF	Comparator Interrupt Enable Falling Enables the CFF interrupt from the CMP. When this field is set, an interrupt will be asserted when CFF is set. 0 Interrupt is disabled. 1 Interrupt is enabled.
2 CFR	Analog Comparator Flag Rising Detects a rising-edge on COUT, when set, during normal operation. CFR is cleared by writing 1 to it. During Stop modes, CFR is level sensitive . 0 Rising-edge on COUT has not been detected. 1 Rising-edge on COUT has occurred.
1 CFF	Analog Comparator Flag Falling Detects a falling-edge on COUT, when set, during normal operation. CFF is cleared by writing 1 to it. During Stop modes, CFF is level sensitive . 0 Falling-edge on COUT has not been detected. 1 Falling-edge on COUT has occurred.
0 COUT	Analog Comparator Output Returns the current value of the Analog Comparator output, when read. The field is reset to 0 and will read as CR1[INV] when the Analog Comparator module is disabled, that is, when CR1[EN] = 0. Writes to this field are ignored.

65.3.5 DAC Control Register (CMPx_DACCR)

Address: Base address + 4h offset

Bit	7	6	5	4	3	2	1	0
Read Write	DACEN	VRSEL	VOSEL					
Reset	0	0	0	0	0	0	0	0

CMPx_DACCR field descriptions

Field	Description
7 DACEN	DAC Enable Enables the DAC. When the DAC is disabled, it is powered down to conserve power. 0 DAC is disabled. 1 DAC is enabled.
6 VRSEL	Supply Voltage Reference Source Select 0 V_{in1} is selected as resistor ladder network supply reference. 1 V_{in2} is selected as resistor ladder network supply reference.
VOSEL	DAC Output Voltage Select Selects an output voltage from one of 64 distinct levels. `DACO = (V in /64) * (VOSEL[5:0] + 1)` , so the DACO range is from V_{in} /64 to V_{in} .

65.3.6 MUX Control Register (CMPx_MUXCR)

Address: Base address + 5h offset

Bit	7	6	5	4	3	2	1	0
Read Write	Reserved	0	PSEL			MSEL		
Reset	0	0	0	0	0	0	0	0

CMPx_MUXCR field descriptions

Field	Description
7 Reserved	Bit can be programmed to zero only . This field is reserved.
6 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.
5–3 PSEL	Plus Input Mux Control Determines which input is selected for the plus input of the comparator. For INx inputs, see CMP, DAC, and ANMUX block diagrams.

Table continues on the next page...

CMPx_MUXCR field descriptions (continued)

Field	Description
	NOTE: When an inappropriate operation selects the same input for both muxes, the comparator automatically shuts down to prevent itself from becoming a noise generator. 000 IN0 001 IN1 010 IN2 011 IN3 100 IN4 101 IN5 110 IN6 111 IN7
MSEL	Minus Input Mux Control Determines which input is selected for the minus input of the comparator. For INx inputs, see CMP, DAC, and ANMUX block diagrams. **NOTE:** When an inappropriate operation selects the same input for both muxes, the comparator automatically shuts down to prevent itself from becoming a noise generator. 000 IN0 001 IN1 010 IN2 011 IN3 100 IN4 101 IN5 110 IN6 111 IN7

65.4 Functional description

The CMP module can be used to compare two analog input voltages applied to INP and INM.

CMPO is high when the non-inverting input is greater than the inverting input, and is low when the non-inverting input is less than the inverting input. This signal can be selectively inverted by setting CR1[INV] = 1.

SCR[IER] and SCR[IEF] are used to select the condition which will cause the CMP module to assert an interrupt to the processor. SCR[CFF] is set on a falling-edge and SCR[CFR] is set on rising-edge of the comparator output. The optionally filtered CMPO can be read directly through SCR[COUT].

65.4.1 CMP functional modes

There are the following main sub-blocks to the CMP module:

- The comparator itself
- The window function
- The filter function

The filter, CR0[FILTER_CNT], can be clocked from an internal or external clock source. The filter is programmable with respect to the number of samples that must agree before a change in the output is registered. In the simplest case, only one sample must agree. In this case, the filter acts as a simple sampler.

The external sample input is enabled using CR1[SE]. When set, the output of the comparator is sampled only on rising edges of the sample input.

The "windowing mode" is enabled by setting CR1[WE]. When set, the comparator output is sampled only when WINDOW=1. This feature can be used to ignore the comparator output during time periods in which the input voltages are not valid. This is especially useful when implementing zero-crossing-detection for certain PWM applications.

The comparator filter and sampling features can be combined as shown in the following table. Individual modes are discussed below.

Table 65-2. Comparator sample/filter controls

Mode #	CR1[EN]	CR1[WE]	CR1[SE]	CR0[FILTER_CNT]	FPR[FILT_PER]	Operation
1	0	X	X	X	X	**Disabled** See the Disabled mode (# 1).
2A	1	0	0	0x00	X	**Continuous Mode** See the Continuous mode (#s 2A & 2B).
2B	1	0	0	X	0x00	
3A	1	0	1	0x01	X	**Sampled, Non-Filtered mode** See the Sampled, Non-Filtered mode (#s 3A & 3B).
3B	1	0	0	0x01	> 0x00	
4A	1	0	1	> 0x01	X	**Sampled, Filtered mode** See the Sampled, Filtered mode (#s 4A & 4B).
4B	1	0	0	> 0x01	> 0x00	
5A	1	1	0	0x00	X	**Windowed mode** Comparator output is sampled on every rising bus clock edge when SAMPLE=1 to generate COUTA. See the Windowed mode (#s 5A & 5B).
5B	1	1	0	X	0x00	
6	1	1	0	0x01	0x01–0xFF	**Windowed/Resampled mode**

Table continues on the next page...

Table 65-2. Comparator sample/filter controls (continued)

Mode #	CR1[EN]	CR1[WE]	CR1[SE]	CR0[FILTER_CNT]	FPR[FILT_PER]	Operation
						Comparator output is sampled on every rising bus clock edge when SAMPLE=1 to generate COUTA, which is then resampled on an interval determined by FILT_PER to generate COUT. See the Windowed/Resampled mode (# 6).
7	1	1	0	> 0x01	0x01–0xFF	**Windowed/Filtered mode** Comparator output is sampled on every rising bus clock edge when SAMPLE=1 to generate COUTA, which is then resampled and filtered to generate COUT. See the Windowed/Filtered mode (#7).
All other combinations of CR1[EN], CR1[WE], CR1[SE], CR0[FILTER_CNT], and FPR[FILT_PER] are illegal.						

For cases where a comparator is used to drive a fault input, for example, inverter-control module such as PWM, it must be configured to operate in Continuous mode so that an external fault can immediately pass through the comparator to the target fault circuitry.

Note

Filtering and sampling settings must be changed only after setting CR1[SE]=0 and CR0[FILTER_CNT]=0x00. This resets the filter to a known state.

65.4.1.1 Disabled mode (# 1)

In Disabled mode, the analog comparator is non-functional and consumes no power. CMPO is 0 in this mode.

65.4.1.2 Continuous mode (#s 2A & 2B)

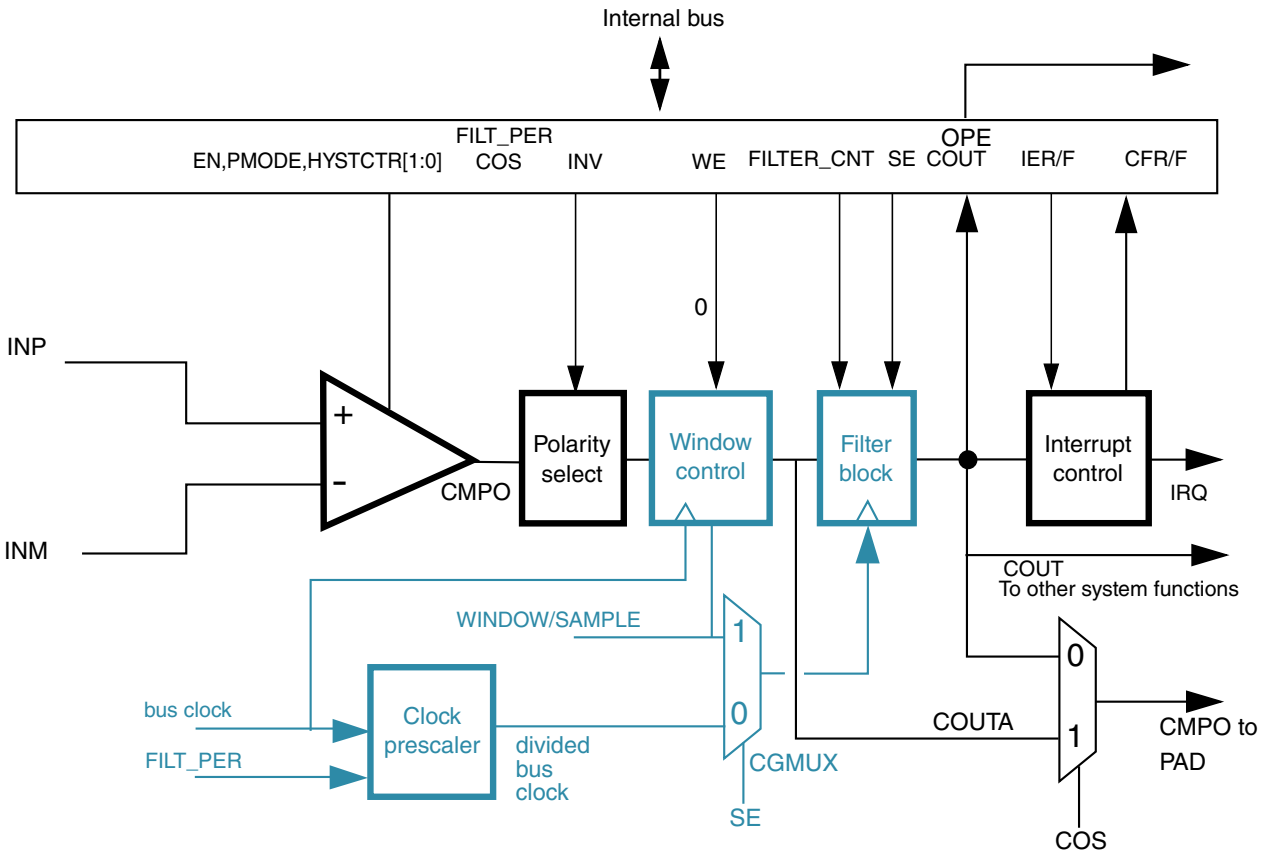


Figure 65-3. Comparator operation in Continuous mode

The analog comparator block is powered and active. CMPO may be optionally inverted, but is not subject to external sampling or filtering. Both window control and filter blocks are completely bypassed. SCR[COUT] is updated continuously. The path from comparator input pins to output pin is operating in combinational unclocked mode. COUT and COUTA are identical.

For control configurations which result in disabling the filter block, see the Filter Block Bypass Logic diagram.

65.4.1.3 Sampled, Non-Filtered mode (#s 3A & 3B)

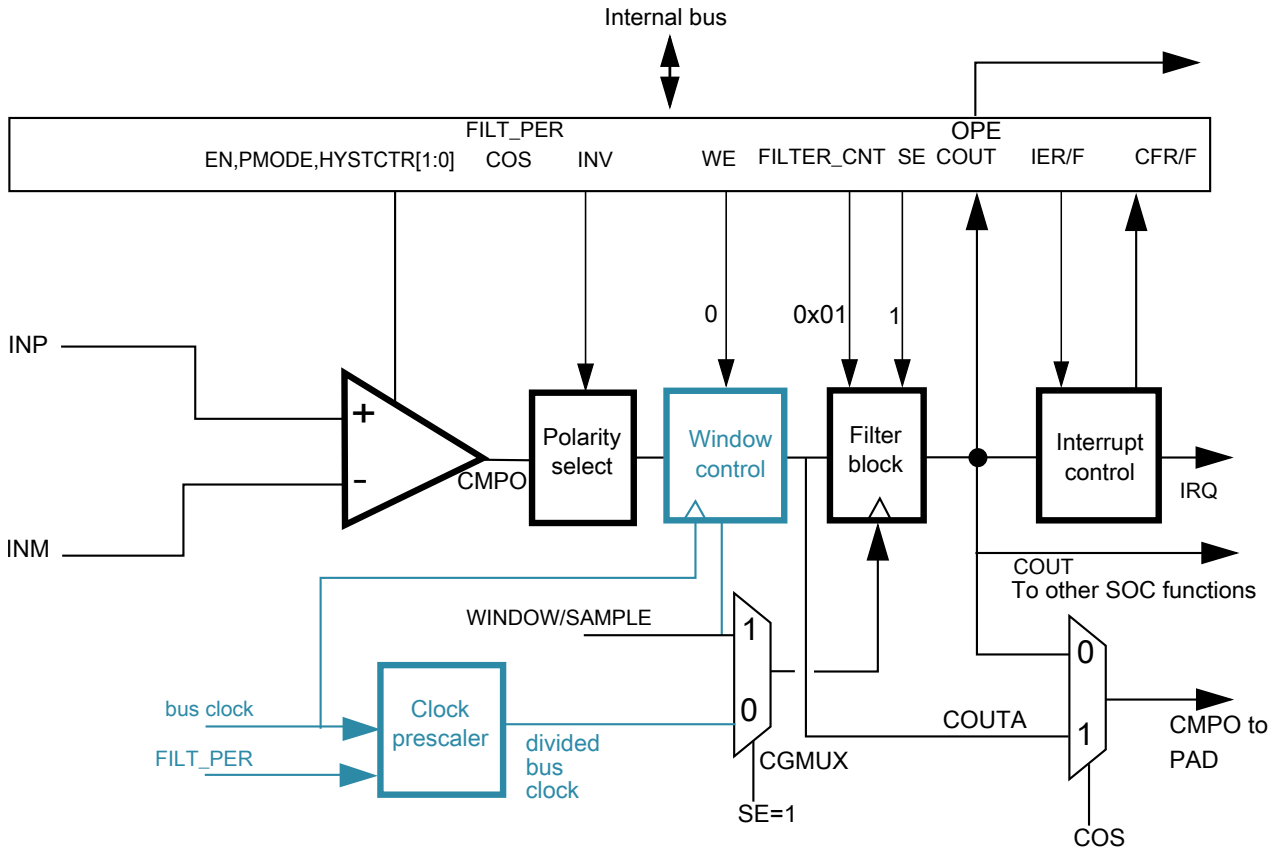


Figure 65-4. Sampled, Non-Filtered (# 3A): sampling point externally driven

In Sampled, Non-Filtered mode, the analog comparator block is powered and active. The path from analog inputs to COUTA is combinational unclocked. Windowing control is completely bypassed. COUTA is sampled whenever a rising-edge is detected on the filter block clock input.

The only difference in operation between Sampled, Non-Filtered (# 3A) and Sampled, Non-Filtered (# 3B) is in how the clock to the filter block is derived. In #3A, the clock to filter block is externally derived while in #3B, the clock to filter block is internally derived.

The comparator filter has no other function than sample/hold of the comparator output in this mode (# 3B).

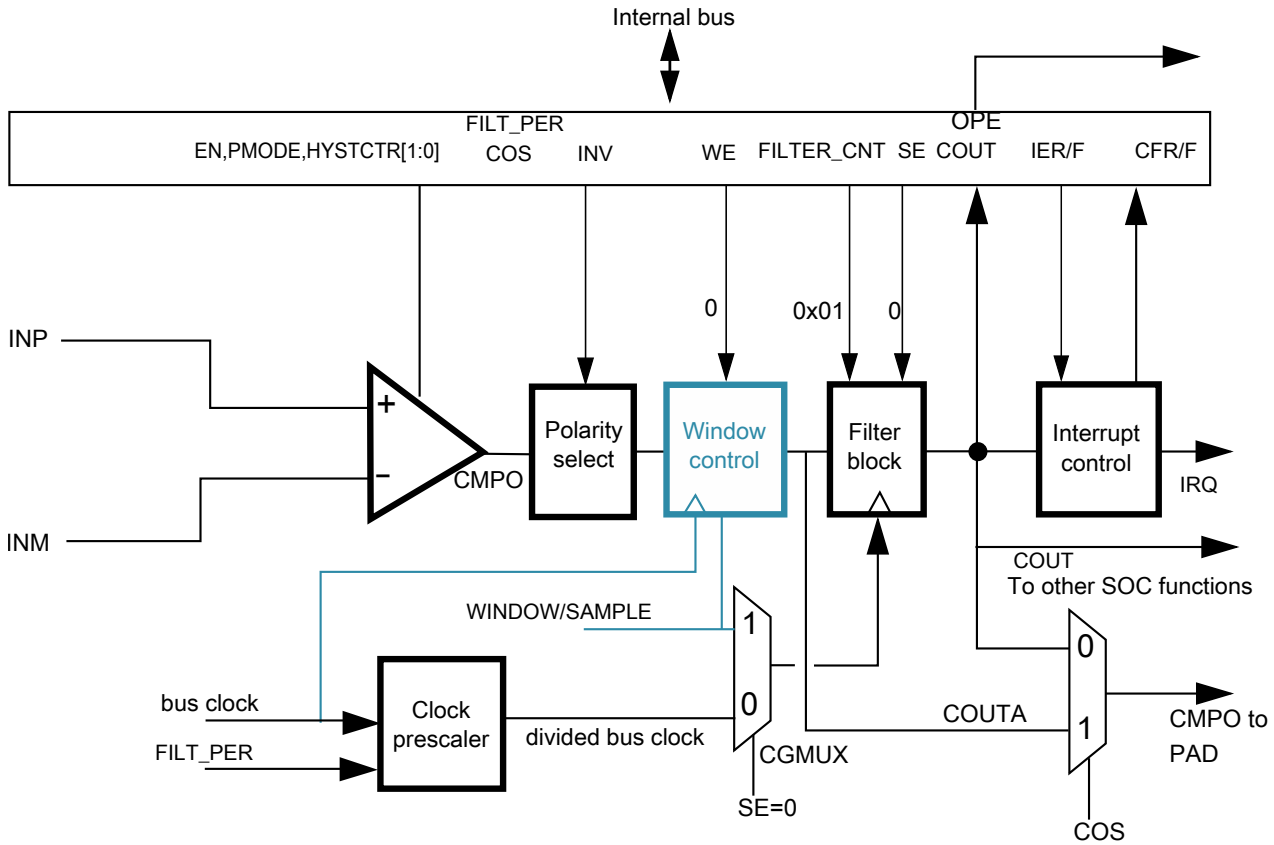


Figure 65-5. Sampled, Non-Filtered (# 3B): sampling interval internally derived

65.4.1.4 Sampled, Filtered mode (#s 4A & 4B)

In Sampled, Filtered mode, the analog comparator block is powered and active. The path from analog inputs to COUTA is combinational unclocked. Windowing control is completely bypassed. COUTA is sampled whenever a rising edge is detected on the filter block clock input.

The only difference in operation between Sampled, Non-Filtered (# 3A) and Sampled, Filtered (# 4A) is that, now, CR0[FILTER_CNT]>1, which activates filter operation.

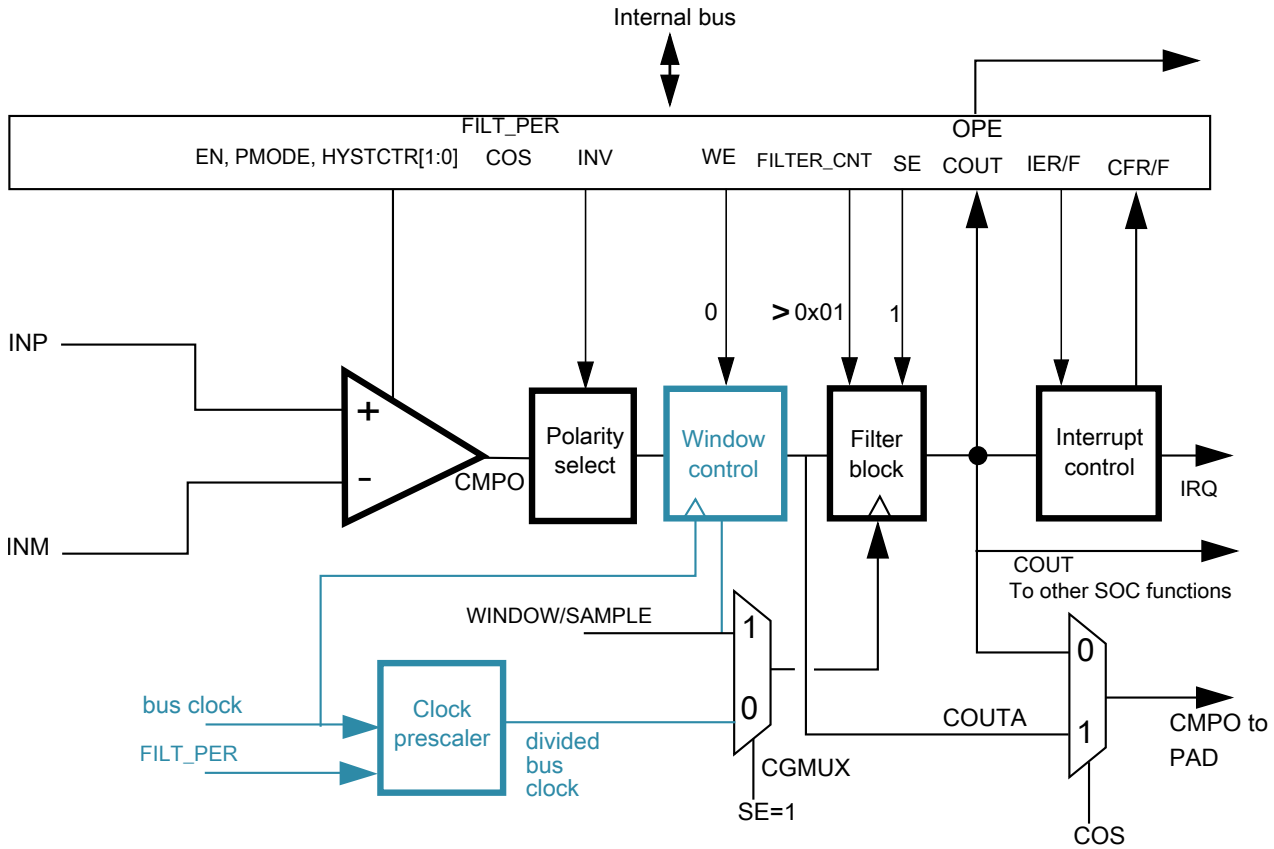


Figure 65-6. Sampled, Filtered (# 4A): sampling point externally driven

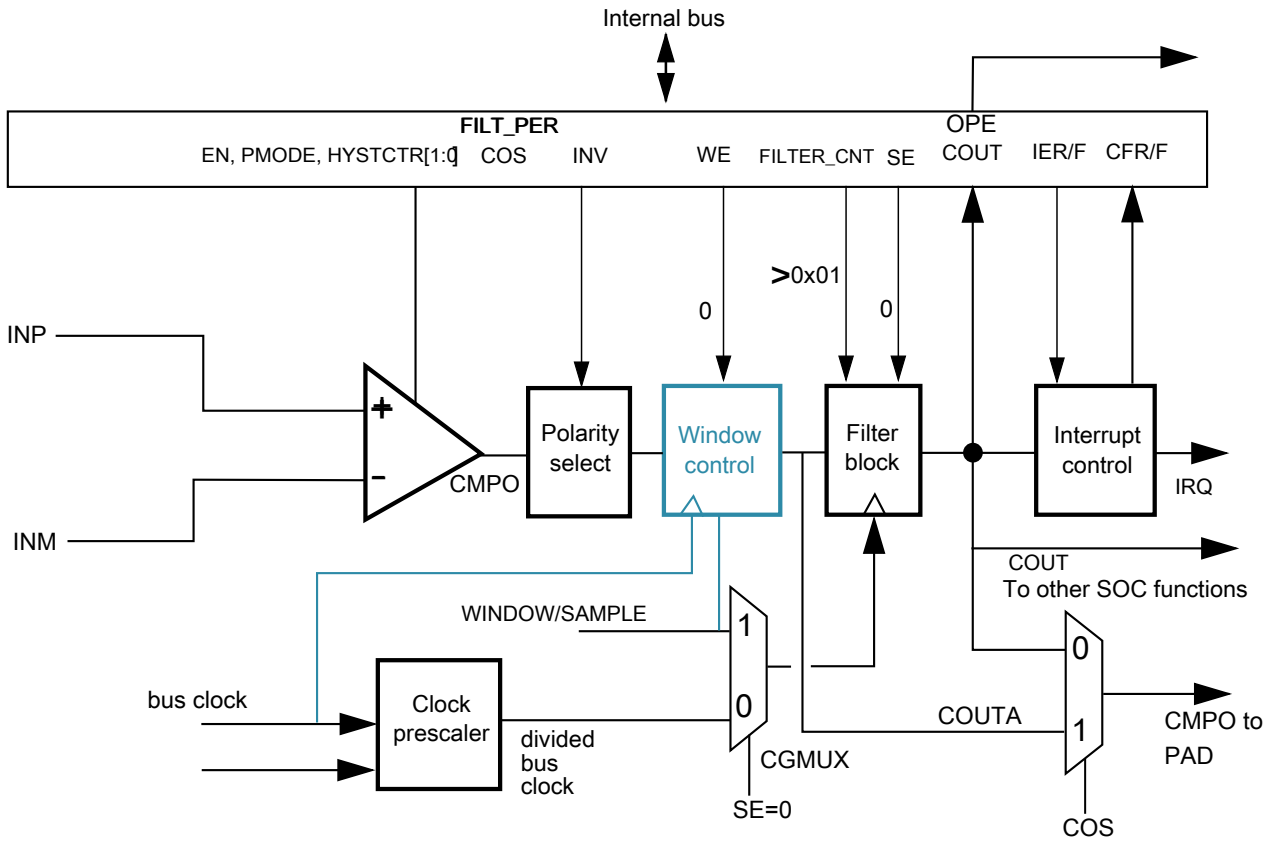


Figure 65-7. Sampled, Filtered (# 4B): sampling point internally derived

The only difference in operation between Sampled, Non-Filtered (# 3B) and Sampled, Filtered (# 4B) is that now, CR0[FILTER_CNT]>1, which activates filter operation.

65.4.1.5 Windowed mode (#s 5A & 5B)

The following figure illustrates comparator operation in the Windowed mode, ignoring latency of the analog comparator, polarity select, and window control block. It also assumes that the polarity select is set to non-inverting state.

NOTE

The analog comparator output is passed to COUTA only when the WINDOW signal is high.

In actual operation, COUTA may lag the analog inputs by up to one bus clock cycle plus the combinational path delay through the comparator and polarity select logic.

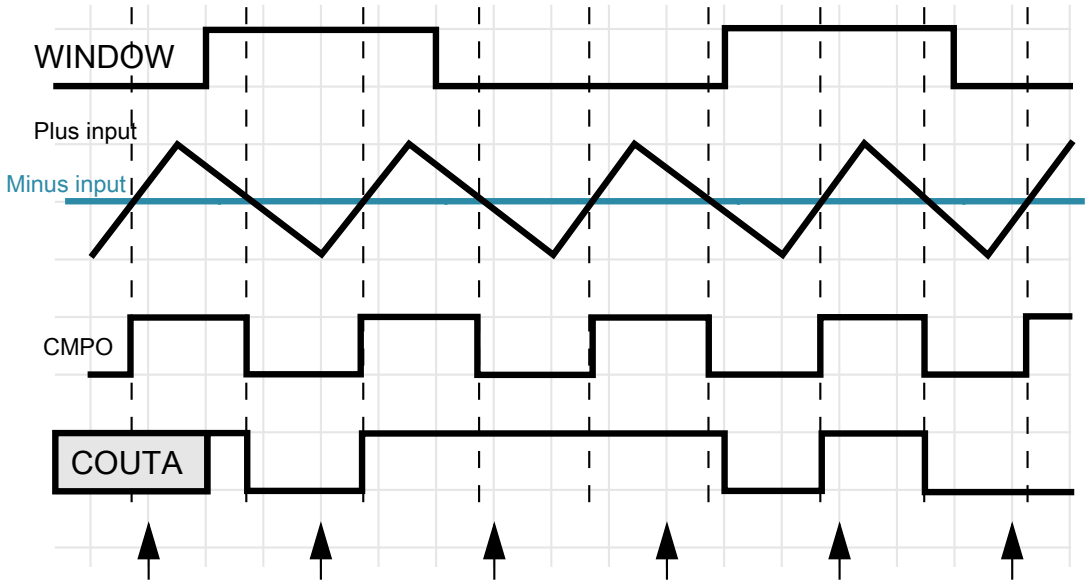


Figure 65-8. Windowed mode operation

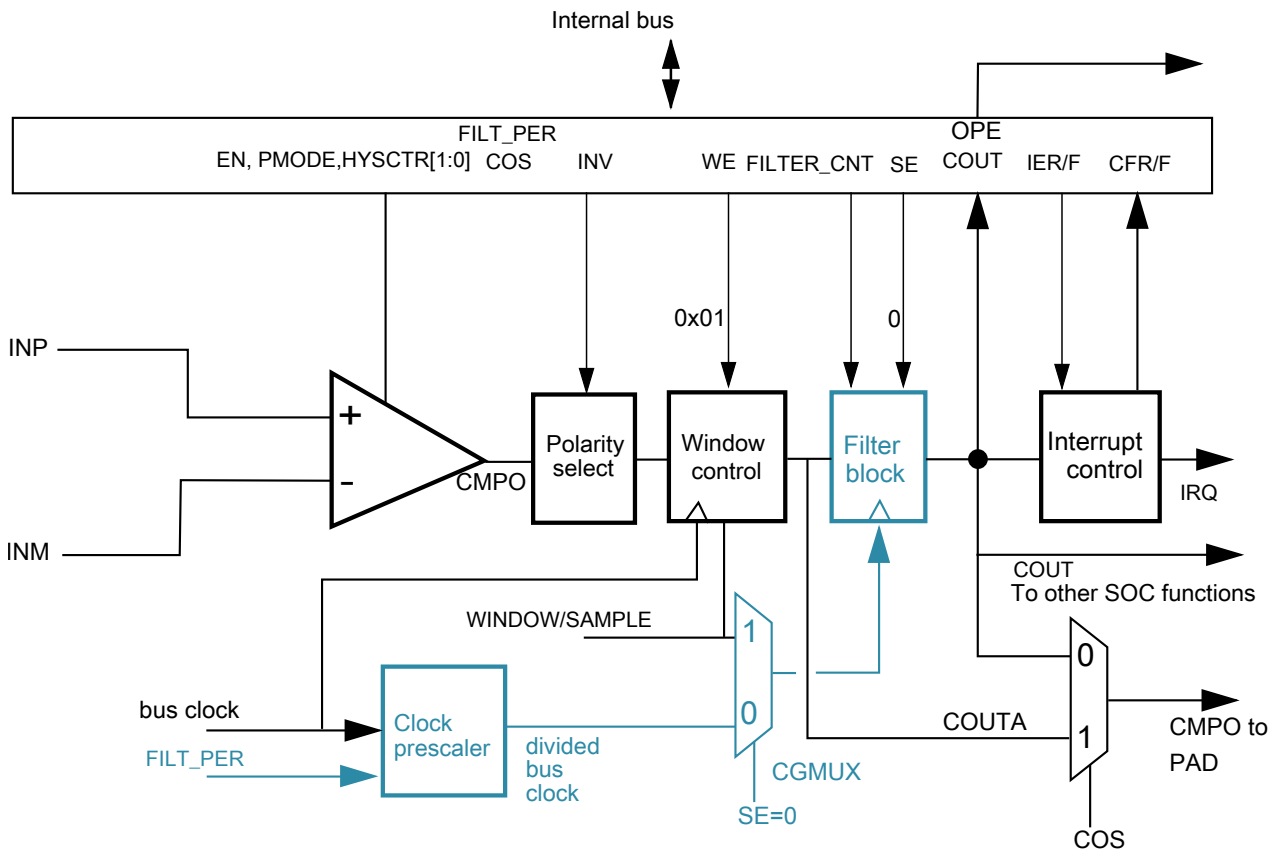


Figure 65-9. Windowed mode

For control configurations which result in disabling the filter block, see Filter Block Bypass Logic diagram.

When any windowed mode is active, COUTA is clocked by the bus clock whenever WINDOW = 1. The last latched value is held when WINDOW = 0.

65.4.1.6 Windowed/Resampled mode (# 6)

The following figure uses the same input stimulus shown in the figure "Windowed mode operation", and adds resampling of COUTA to generate COUT. Samples are taken at the time points indicated by the arrows in the figure. Again, prop delays and latency are ignored for the sake of clarity.

This example was generated solely to demonstrate operation of the comparator in windowed/resampled mode, and does not reflect any specific application. Depending upon the sampling rate and window placement, COUT may not see zero-crossing events detected by the analog comparator. Sampling period and/or window placement must be carefully considered for a given application.

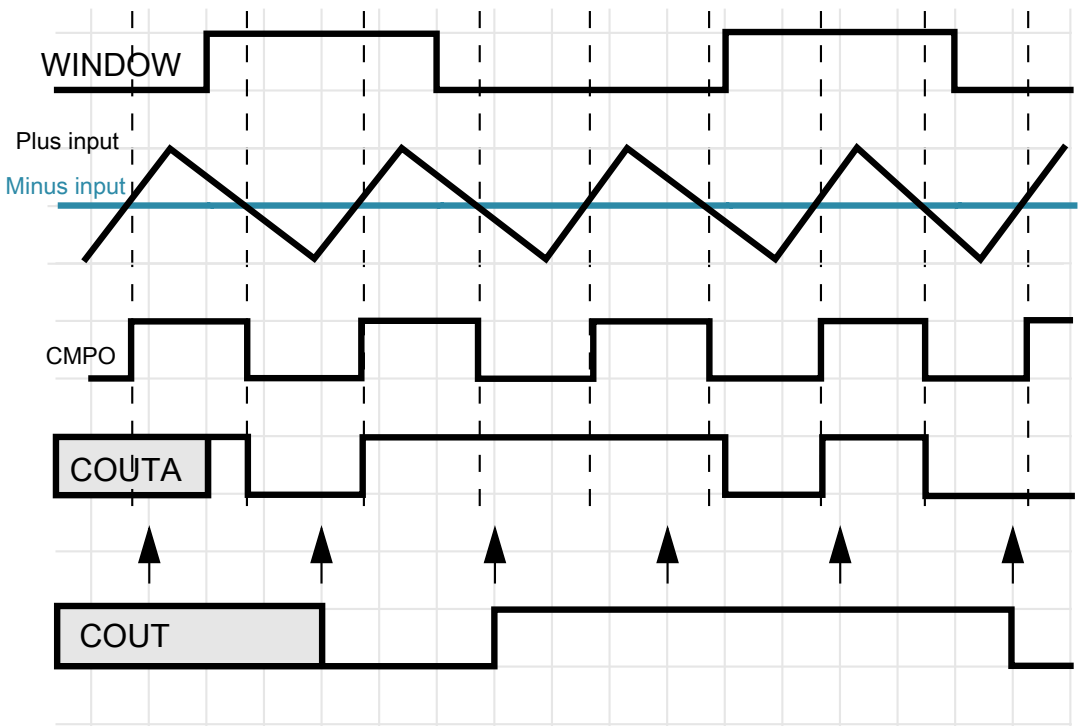


Figure 65-10. Windowed/resampled mode operation

This mode of operation results in an unfiltered string of comparator samples where the interval between the samples is determined by FPR[FILT_PER] and the bus clock rate. Configuration for this mode is virtually identical to that for the Windowed/Filtered Mode shown in the next section. The only difference is that the value of CR0[FILTER_CNT] must be 1.

65.4.1.7 Windowed/Filtered mode (#7)

This is the most complex mode of operation for the comparator block, as it uses both windowing and filtering features. It also has the highest latency of any of the modes. This can be approximated: up to 1 bus clock synchronization in the window function + ((CR0[FILTER_CNT] * FPR[FILT_PER]) + 1) * bus clock for the filter function.

When any windowed mode is active, COUTA is clocked by the bus clock whenever WINDOW = 1. The last latched value is held when WINDOW = 0.

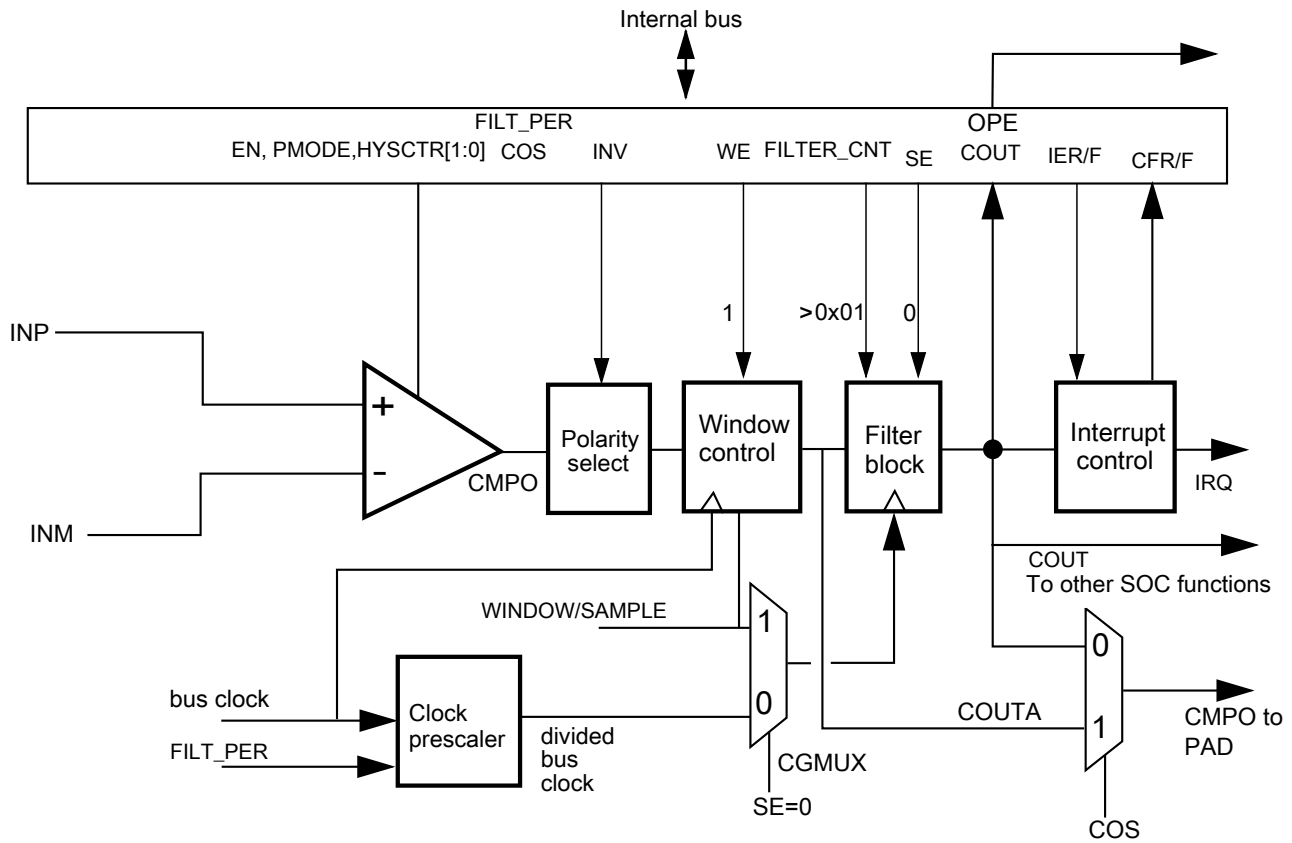


Figure 65-11. Windowed/Filtered mode

65.4.2 Power modes

65.4.2.1 Wait mode operation

During Wait and VLPW modes, the CMP, if enabled, continues to operate normally and a CMP interrupt can wake the MCU.

65.4.2.2 Stop mode operation

Depending on clock restrictions related to the MCU core or core peripherals, the MCU is brought out of stop when a compare event occurs and the corresponding interrupt is enabled. Similarly, if CR1[OPE] is enabled, the comparator output operates as in the normal operating mode and comparator output is placed onto the external pin. In Stop modes, the comparator can be operational in both:

- High-Speed (HS) Comparison mode when CR1[PMODE] = 1
- Low-Speed (LS) Comparison mode when CR1[PMODE] = 0

It is recommended to use the LS mode to minimize power consumption.

If stop is exited with a reset, all comparator registers are put into their reset state.

65.4.3 Startup and operation

A typical startup sequence is listed here.

- The time required to stabilize COUT will be the power-on delay of the comparators plus the largest propagation delay from a selected analog source through the analog comparator, windowing function and filter. See the Data Sheets for power-on delays of the comparators. The windowing function has a maximum of one bus clock period delay. The filter delay is specified in the Low-pass filter.
- During operation, the propagation delay of the selected data paths must always be considered. It may take many bus clock cycles for COUT and SCR[CFR]/SCR[CFF] to reflect an input change or a configuration change to one of the components involved in the data path.
- When programmed for filtering modes, COUT will initially be equal to 0, until sufficient clock cycles have elapsed to fill all stages of the filter. This occurs even if COUTA is at a logic 1.

65.4.4 Low-pass filter

The low-pass filter operates on the unfiltered and unsynchronized and optionally inverted comparator output COUTA and generates the filtered and synchronized output COUT.

Both COUTA and COUT can be configured as module outputs and are used for different purposes within the system.

Synchronization and edge detection are always used to determine status register bit values. They also apply to COUT for all sampling and windowed modes. Filtering can be performed using an internal timebase defined by FPR[FILT_PER], or using an external SAMPLE input to determine sample time.

The need for digital filtering and the amount of filtering is dependent on user requirements. Filtering can become more useful in the absence of an external hysteresis circuit. Without external hysteresis, high-frequency oscillations can be generated at COUTA when the selected INM and INP input voltages differ by less than the offset voltage of the differential comparator.

65.4.4.1 Enabling filter modes

Filter modes can be enabled by:

- Setting CR0[FILTER_CNT] > 0x01 and
- Setting FPR[FILT_PER] to a nonzero value or setting CR1[SE]=1

If using the divided bus clock to drive the filter, it will take samples of COUTA every FPR[FILT_PER] bus clock cycles.

The filter output will be at logic 0 when first initalized, and will subsequently change when all the consecutive CR0[FILTER_CNT] samples agree that the output value has changed. In other words, SCR[COUT] will be 0 for some initial period, even when COUTA is at logic 1.

Setting both CR1[SE] and FPR[FILT_PER] to 0 disables the filter and eliminates switching current associated with the filtering process.

Note

Always switch to this setting prior to making any changes in filter parameters. This resets the filter to a known state. Switching CR0[FILTER_CNT] on the fly without this intermediate step can result in unexpected behavior.

If CR1[SE]=1, the filter takes samples of COUTA on each positive transition of the sample input. The output state of the filter changes when all the consecutive CR0[FILTER_CNT] samples agree that the output value has changed.

65.4.4.2 Latency issues

The value of FPR[FILT_PER] or SAMPLE period must be set such that the sampling period is just longer than the period of the expected noise. This way a noise spike will corrupt only one sample. The value of CR0[FILTER_CNT] must be chosen to reduce the probability of noisy samples causing an incorrect transition to be recognized. The probability of an incorrect transition is defined as the probability of an incorrect sample raised to the power of CR0[FILTER_CNT].

The values of FPR[FILT_PER] or SAMPLE period and CR0[FILTER_CNT] must also be traded off against the desire for minimal latency in recognizing actual comparator output transitions. The probability of detecting an actual output change within the nominal latency is the probability of a correct sample raised to the power of CR0[FILTER_CNT].

The following table summarizes maximum latency values for the various modes of operation *in the absence of noise*. Filtering latency is restarted each time an actual output transition is masked by noise.

Table 65-3. Comparator sample/filter maximum latencies

Mode #	CR1[EN]	CR1[WE]	CR1[SE]	CR0[FILTER_CNT]	FPR[FILT_PER]	Operation	Maximum latency[1]
1	0	X	X	X	X	Disabled	N/A
2A	1	0	0	0x00	X	Continuous Mode	T_{PD}
2B	1	0	0	X	0x00		
3A	1	0	1	0x01	X	Sampled, Non-Filtered mode	$T_{PD} + T_{SAMPLE} + T_{per}$
3B	1	0	0	0x01	> 0x00		T_{PD} + (FPR[FILT_PER] * T_{per}) + T_{per}
4A	1	0	1	> 0x01	X	Sampled, Filtered mode	T_{PD} + (CR0[FILTER_CNT] * T_{SAMPLE}) + T_{per}
4B	1	0	0	> 0x01	> 0x00		T_{PD} + (CR0[FILTER_CNT] * FPR[FILT_PER] x T_{per}) + T_{per}
5A	1	1	0	0x00	X	Windowed mode	$T_{PD} + T_{per}$
5B	1	1	0	X	0x00		$T_{PD} + T_{per}$
6	1	1	0	0x01	0x01 - 0xFF	Windowed / Resampled mode	T_{PD} + (FPR[FILT_PER] * T_{per}) + $2T_{per}$
7	1	1	0	> 0x01	0x01 - 0xFF	Windowed / Filtered mode	T_{PD} + (CR0[FILTER_CNT] * FPR[FILT_PER] x T_{per}) + $2T_{per}$

1. T_{PD} represents the intrinsic delay of the analog component plus the polarity select logic. T_{SAMPLE} is the clock period of the external sample clock. T_{per} is the period of the bus clock.

65.5 CMP interrupts

The CMP module is capable of generating an interrupt on either the rising- or falling-edge of the comparator output, or both.

The following table gives the conditions in which the interrupt request is asserted and deasserted.

When	Then
SCR[IER] and SCR[CFR] are set	The interrupt request is asserted
SCR[IEF] and SCR[CFF] are set	The interrupt request is asserted
SCR[IER] and SCR[CFR] are cleared for a rising-edge interrupt	The interrupt request is deasserted
SCR[IEF] and SCR[CFF] are cleared for a falling-edge interrupt	The interrupt request is deasserted

65.6 DMA support

Normally, the CMP generates a CPU interrupt if there is a change on the COUT. When DMA support is enabled by setting SCR[DMAEN] and the interrupt is enabled by setting SCR[IER], SCR[IEF], or both, the corresponding change on COUT forces a DMA transfer request rather than a CPU interrupt instead. When the DMA has completed the transfer, it sends a transfer completing indicator that deasserts the DMA transfer request and clears the flag to allow a subsequent change on comparator output to occur and force another DMA request.

The comparator can remain functional in STOP modes.

When DMA support is enabled by setting SCR[DMAEN] and the interrupt is enabled by setting SCR[IER], SCR[IEF], or both, the corresponding change on COUT forces a DMA transfer request to wake up the system from STOP modes. After the data transfer has finished, system will go back to STOP modes. Refer to DMA chapters in the device reference manual for the asynchronous DMA function for details.

65.7 Digital-to-analog converter

The figure found here shows the block diagram of the DAC module.

It contains a 64-tap resistor ladder network and a 64-to-1 multiplexer, which selects an output voltage from one of 64 distinct levels that outputs from DACO. It is controlled through the DAC Control Register (DACCR). Its supply reference source can be selected from two sources V_{in1} and V_{in2}. The module can be powered down or disabled when not in use. When in Disabled mode, DACO is connected to the analog ground.

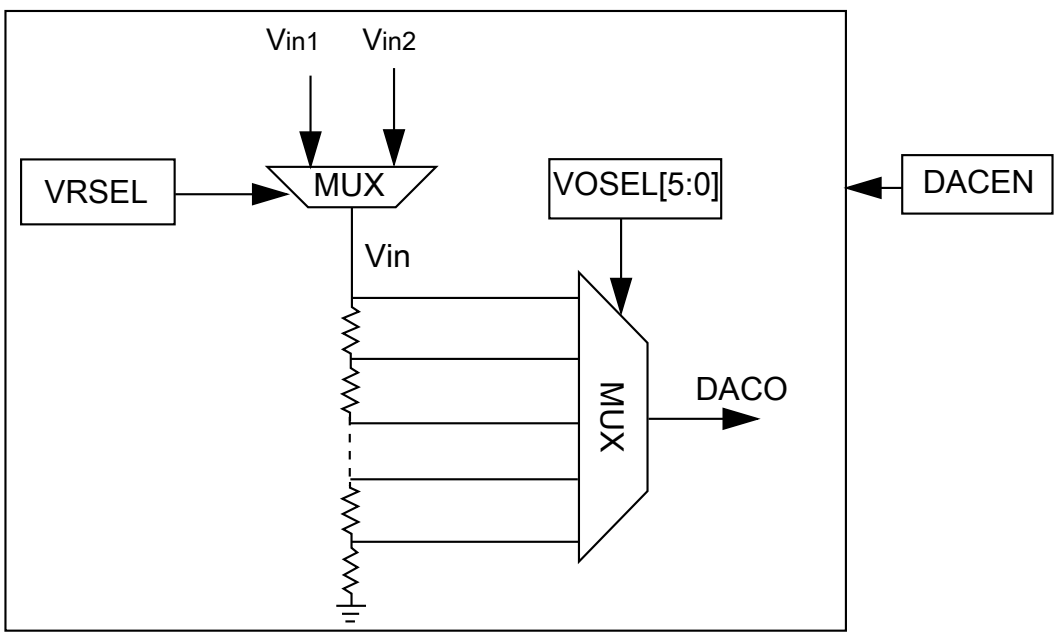


Figure 65-12. 6-bit DAC block diagram

65.8 DAC functional description

This section provides DAC functional description information.

65.8.1 Voltage reference source select

- V_{in1} connects to the primary voltage source as supply reference of 64 tap resistor ladder
- V_{in2} connects to an alternate voltage source

65.9 DAC resets

This module has a single reset input, corresponding to the chip-wide peripheral reset.

65.10 DAC clocks

This module has a single clock input, the bus clock.

65.11 DAC interrupts

This module has no interrupts.

Chapter 66
Analog-to-Digital Converter (ADC)

66.1 Chip-specific ADC information

Table 66-1. Reference links to related information

Topic	Related module(s)	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
Interrupts, DMA Events and XBAR Assignments	-	Interrupts, DMA Events and XBAR Assignments

NOTE

V_{REFH} / V_{REFL} are bonded to VDDA_ADC_3P3 / GND, on this device.

66.2 Overview

The analog-to-digital converter (ADC) is a successive approximation ADC designed for operation within an integrated microcontroller system-on-chip.

66.2.1 Features

The features of the ADC are as follows:

- Configuration registers

    - 32-bit, word aligned, byte enabled registers. (byte and halfword access is not supported)
- Linear successive approximation algorithm with up to 12-bit resolution with 10/11 bit accuracy.
- Up to 10 ENOB (dedicated single ended channels)
- Up to 1MS/s sampling rate
- Up to 16 single-ended external analog inputs
- Single or continuous conversion (automatic return to idle after single conversion)
- Output Modes: (in right-justified unsigned format)
    - 12-bit
    - 10-bit
    - 8-bit
- Configurable sample time and conversion speed/power
- Conversion complete and hardware average complete flag and interrupt
- Input clock selectable from up to three sources
- Asynchronous clock source for lower noise operation with option to output the clock
- Selectable asynchronous hardware conversion trigger with hardware channel select
- Automatic compare with interrupt for less-than, greater-than or equal-to, within range, or out-of-range, programmable value
- Operation in low power modes for lower noise operation
- Hardware average function
- Self-calibration mode

66.2.2 ADC I/F block diagram

The following diagram represents the ADC I/F block.

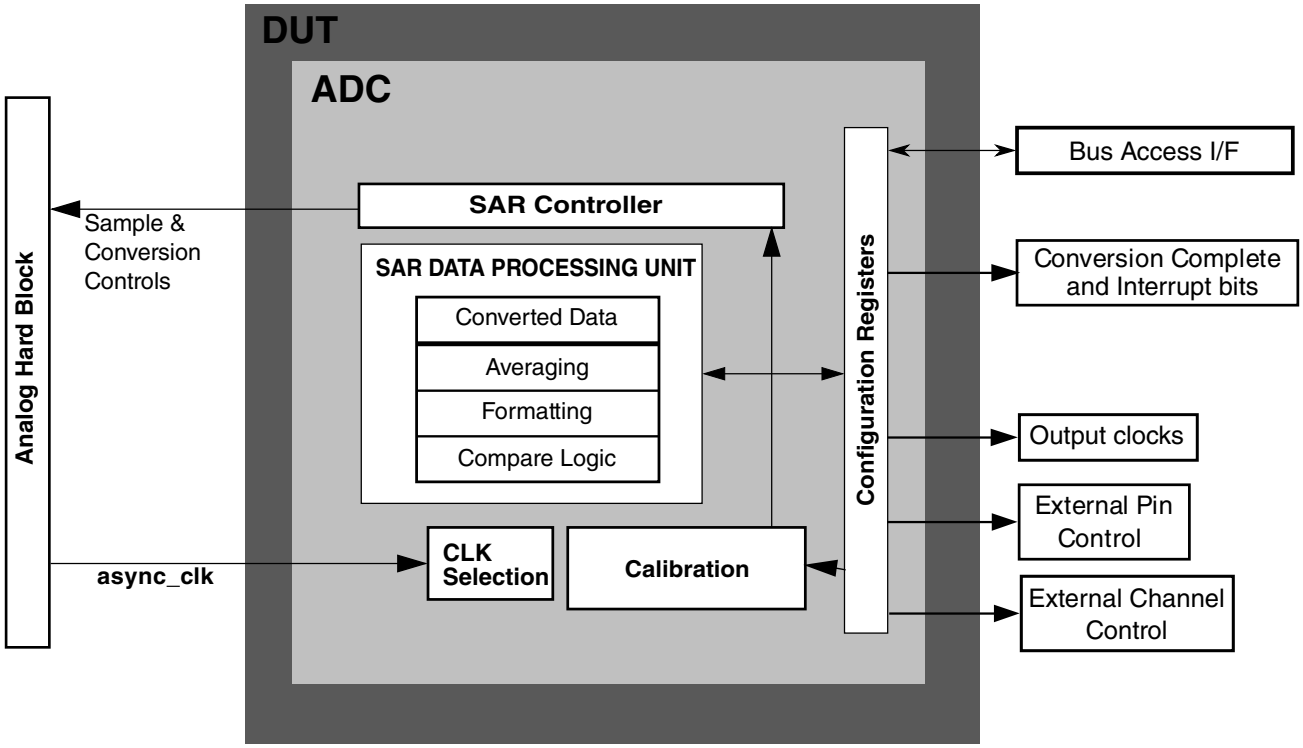


Figure 66-1. ADC I/F block diagram

66.2.3 ADC block diagram

The following figure shows a top-level block diagram of the ADC module.

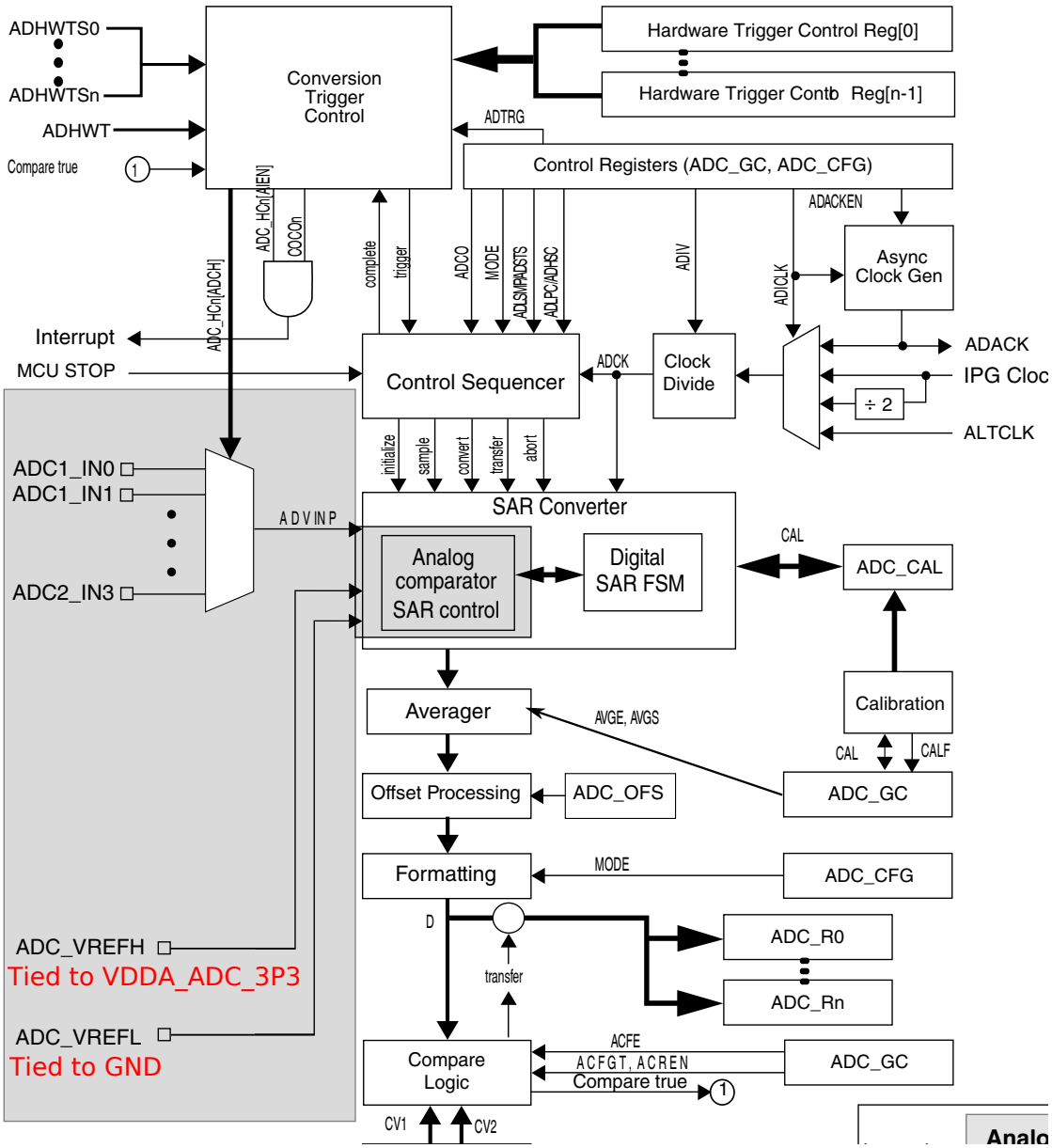


Figure 66-2. ADC block diagram

66.2.4 ADC module interface

The ADC is connected to many interfaces such as the clocks and reset, access bus, voltage references, interrupt controller, hardware triggers, ADC pin control, and analog I/F as shown in the following figure.

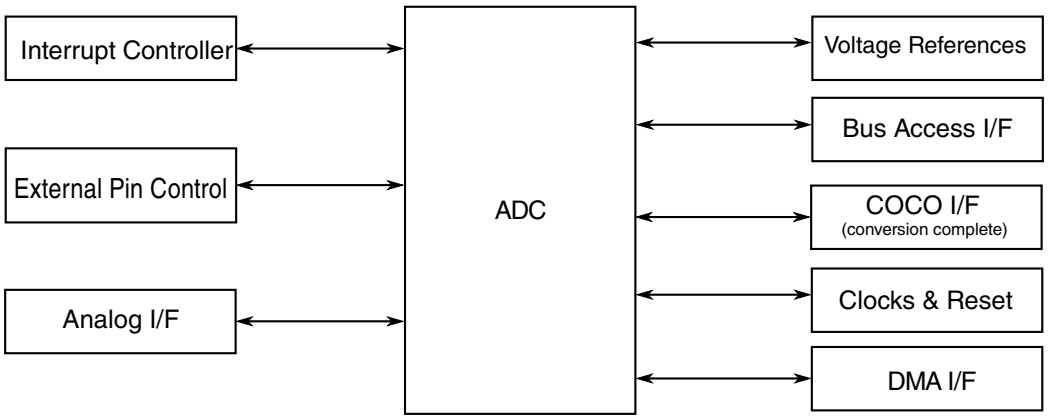


Figure 66-3. ADC module interface

66.2.5 Modes of Operation

By default, the ADC is in disabled mode. In this state, no conversion or other actions occur. All of the ADC control registers are accessible in this state through an access bus interface.To enable the ADC, required configurations should be done by programming the ADC configuration registers.

66.3 External Signals

The following table describes the external signals of ADC:

Table 66-2. ADC external signals

Signal	Description	Pad	Mode	Direction
ADC 1_IN0	Analog channel 1 input 0	GPIO_AD_B1_11 pin 21	-	I
ADC1_IN1	Analog channel 1 input 1	GPIO_AD_B0_12 pin 24	-	I
ADC1_IN2	Analog channel 1 input 2	GPIO_AD_B0_13 pin 25	-	I
ADC1_IN3	Analog channel 1 input 3	GPIO_AD_B0_14	-	I
ADC1_IN4	Analog channel 1 input 4	GPIO_AD_B0_15	-	I
ADC1_IN5	Analog channel 1 input 5	GPIO_AD_B1_00 pin 19	-	I
ADC1_IN6	Analog channel 1 input 6	GPIO_AD_B1_01 pin 18	-	I
ADC1_IN7	Analog channel 1 input 7	GPIO_AD_B1_02 pin 14	-	I

Table continues on the next page...

Table 66-2. ADC external signals (continued)

Signal	Description	Pad	Mode	Direction
ADC1_IN8	Analog channel 1 input 8	GPIO_AD_B1_03 pin 15	-	I
ADC1_IN9	Analog channel 1 input 9	GPIO_AD_B1_04 pin 40 (T4.1)	-	I
ADC1_IN10	Analog channel 1 input 10	GPIO_AD_B1_05 pin 41 (T4.1)	-	I
ADC1_IN11	Analog channel 1 input 11	GPIO_AD_B1_06 pin 17	-	I
ADC1_IN12	Analog channel 1 input 12	GPIO_AD_B1_07 pin 16	-	I
ADC1_IN13	Analog channel 1 input 13	GPIO_AD_B1_08 pin 22	-	I
ADC1_IN14	Analog channel 1 input 14	GPIO_AD_B1_09 pin 23	-	I
ADC1_IN15	Analog channel 1 input 15	GPIO_AD_B1_10 pin 20	-	I
ADC 2_IN0	Analog channel 2 input 0	GPIO_AD_B1_11 pin 21	-	I
ADC2_IN1	Analog channel 2 input 1	GPIO_AD_B1_12 pin 38 (T4.1)	-	I
ADC2_IN2	Analog channel 2 input 2	GPIO_AD_B1_13 pin 39 (T4.1)	-	I
ADC2_IN3	Analog channel 2 input 3	GPIO_AD_B1_14 pin 26	-	I
ADC2_IN4	Analog channel 2 input 4	GPIO_AD_B1_15 pin 27	-	I
ADC2_IN5	Analog channel 2 input 5	GPIO_AD_B1_00 pin 19	-	I
ADC2_IN6	Analog channel 2 input 6	GPIO_AD_B1_01 pin 18	-	I
ADC2_IN7	Analog channel 2 input 7	GPIO_AD_B1_02 pin 14	-	I
ADC2_IN8	Analog channel 2 input 8	GPIO_AD_B1_03 pin 15	-	I
ADC2_IN9	Analog channel 2 input 9	GPIO_AD_B1_04 pin 40 (T4.1)	-	I
ADC2_IN10	Analog channel 2 input 10	GPIO_AD_B1_05 pin 41 (T4.1)	-	I
ADC2_IN11	Analog channel 2 input 11	GPIO_AD_B1_06 pin 17	-	I
ADC2_IN12	Analog channel 2 input 12	GPIO_AD_B1_07 pin 16	-	I
ADC2_IN13	Analog channel 2 input 13	GPIO_AD_B1_08 pin 22	-	I
ADC2_IN14	Analog channel 2 input 14	GPIO_AD_B1_09 pin 23	-	I

Table continues on the next page...

Table 66-2. ADC external signals (continued)

Signal	Description	Pad	Mode	Direction
ADC2_IN15	Analog channel 2 input 15	GPIO_AD_B1_10 pin 20	-	I

NOTE

The ADC input signals connect to GPIO. The GPIO default configuration is enabled for keeper. The keeper causes an undesired jump behavior in ADC. To avoid the problem, disable keeper before starting ADC. For detailed information about keeper, refer to the GPIO block.

66.4 Clocks

The following table describes the clock sources for ADC.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 66-3. ADC Clocks

Clock name	Clock Root	Description
ipg_clk	ipg_clk_root	Peripheral clock

66.5 Functional Description

There are three possible states which ADC module can be in:

1. Disabled State
2. Idle state
3. Performing conversions

Disabled State:

The ADC module is disabled during reset or stop mode (if internal clock is not selected as source of clock), or when the ADCH bits of the hardware control (ADC_HC*n*) registers are all high.

Idle State:

The module is idle when a conversion has completed and another conversion has not been initiated. When idle and the asynchronous clock output enable is disabled (ADACKEN = 0), the module is in its lowest power state.

Conversion State:

The ADC can perform an analog-to-digital conversion on any of the software selectable channels. All modes perform conversion by a successive approximation algorithm.

To meet accuracy specifications the ADC module must be calibrated using the on chip calibration function. Calibration is recommended to be done after any reset.

When the conversion is completed, the result is placed in the data result registers (ADC_R*n*). The conversion complete flag (COCO*n*) field in the Hardware Status register is/are then set and an interrupt is generated, if the respective conversion complete interrupt has been enabled (ADC_HC*n*[AIEN]=1).

The ADC module has the capability of automatically comparing the result of a conversion with the contents of the compare value registers. The compare function is enabled by setting ACFE (ADC Compare Function Enable) in the ADC general control register.

The ADC module has the capability of automatically averaging the result of multiple conversions. The hardware average function is enabled by setting AVGE in the ADC general control register.

66.5.1 Clock Select and Divide Control

The ADC digital module has two clock sources:

- IPG clock
- Internal clock (ADACK) is a dedicated clock used only by the ADC.

ADC digital block generates IPG clock/2 by internally dividing the IPG clock. The final clock is chosen from the following clocks.

- IPG clock
- IPG clock divided by 2
- ADACK

From the three clocks listed above, one is chosen depending on the configuration of ADICLK[1:0] bits of ADC_CFG. This chosen clock is divided depending on the configuration of ADIV[1:0] bits of ADC_CFG. The final generated clock is used as conversion clock for ADC.

ADICLK	Selected Clock Source
00	IPG clock
01	IPG clock divided by 2
10	Reserved
11	Asynchronous clock (ADACK)

- The IPG clock. This is the default selection following reset.
- The IPG clock divided by two. For higher IPG clock rates, this allows a maximum divide by 16 of the IPG clock using the ADIV bits.
- The asynchronous clock (ADACK). This clock is generated from a clock source within the ADC module. Conversions are possible using ADACK as the input clock source while the MCU is in stop mode.

Whichever clock is selected, its frequency must fall within the specified frequency range for ADCK. If the available clocks are too slow, the ADC may not perform according to specifications. If the available clocks are too fast, the clock must be divided to the appropriate frequency. This divider is specified by the ADIV bits and can be divide-by 1, 2, 4, or 8.

66.5.2 Voltage Reference Selection

The ADC can be configured to use reference pairs as the reference voltages used for conversions (V_{REFSH} and V_{REFSL}). Each pair contains a positive reference which must be between the minimum Ref Voltage High and V_{DDAD}, and a ground reference which must be at the same potential as V_{SSAD}. The pairs can be as follows:

- External (V_{REFH} and V_{REFL})

These voltage references are selected by configuring ADC_CFG[REFSEL].

66.5.3 Hardware Triggering and Channel Selection

The ADC module has a trigger input (known as Alternate Trigger) which provides asynchronous hardware conversion trigger when the ADTRG bit in ADC configuration register (ADC_CFG) is set and any of the external hardware trigger select is high.

To be reliably captured, the Alternate Trigger pulse must be high for sufficient time to satisfy clocking requirement of capturing Flop and the external hardware trigger select event must be set for sufficient time before and after the positive edge of Alternate trigger pulse to meet the setup / hold requirement of capturing flop.

If an external hardware trigger select event gets asserted during a conversion it must stay asserted until end of current conversion and remain set until the receipt of the an Alternate Trigger to initiate a new conversion.

When the Alternate Trigger source is available and hardware triggering is enabled (ADC_CFG[ADTRG]=1), a conversion is initiated on the rising edge of the Alternate Trigger after a external hardware trigger select event has occurred.

If a conversion is in progress when a rising edge of a trigger occurs, the rising edge is ignored. In continuous convertion configuration, only the initial rising edge to launch continuous conversions is observed and until conversion gets aborted the ADC will continue to do conversions on the same ADC Hardware Trigger Control register that initiated the conversion. The hardware trigger function operates in conjunction with any of the conversion modes and configurations.

The channel selected for the conversion will depend on the settings of active Hardware Trigger register field ADC_HCn[ADCH] of enabled external hardware trigger.

NOTE

Asserting more than one external hardware trigger select signal at the same time will result in unknown results. To avoid this, only select one external hardware trigger select signal prior to the next intended conversion.

When the conversion is completed, the result is placed in the data registers associated with the external hardware trigger received (active trigger selects ADC_Rn). The conversion complete flag associated with the external hardware trigger received (ADC_HS[COCOn]) is then set and an interrupt is generated if the respective conversion complete interrupt has been enabled (ADC_HCn[AIEN]=1).

66.5.4 Conversion Control

Conversions are performed as determined by ADC_CFG[MODE] field.

Conversions can be initiated by either software or hardware triggers. In addition, the ADC can be configured for low power operation, long sample time, continuous conversion, hardware average, and automatic comparison of conversion results with predetermined values.

66.5.4.1 Initiating Conversions

A conversion is initiated:

- Following a write to ADC_HC0 (with ADCHn bits not all 1's) and when a software triggered operation is selected (ADTRG=0).
- Following a hardware trigger event if hardware triggered operation is selected (ADTRG=1) and a external hardware trigger select event has occurred. The channel selected will depend on the active trigger select signal active selects ADC_HC1; if neither is active the off condition is selected).

Note

Selecting more than one external hardware trigger select signal (ext_hwts[n]) prior to a conversion completion will generate unknown results. To avoid this, only select one hardware trigger select signal (ext_hwts[n]) prior to a conversion completion.

- Following the transfer of the result to the data registers when continuous conversion is enabled (ADCO=1 in ADC_GC register).

If continuous conversion is enabled, a new conversion is automatically initiated after the completion of the current conversion. In software triggered operation (ADTRG=0), continuous conversions begin after ADC_HC0 is written and continue until aborted. In hardware triggered operation(ADTRG=1 and one external hardware trigger select event has occurred), continuous conversions begin after a hardware trigger event and continue until aborted.

If hardware averaging is enabled, a new conversion is automatically initiated after the completion of the current conversion until the correct number of conversions is completed. In software triggered operation, conversions begin after ADC_HC0 is written. In a hardware triggered operation, conversions begin after a hardware trigger. If continuous conversions is also enabled, a new set of conversions to be averaged are initiated following the last of the selected number of conversions.

66.5.4.2 Completing Conversions

A conversion is completed when the result of the conversion is transferred into the data result registers. (provided the compare function & hardware averaging is disabled), this is indicated by the setting of COCOn. If hardware averaging is enabled, COCOn sets only, if the last of the selected number of conversions is complete. If the compare function is enabled, COCOn sets and conversion result data is transferred only if the compare condition is true. If both hardware averaging and compare functions are enabled, then COCOn sets only if the last of the selected number of conversions is complete and the

compare condition is true. An interrupt is generated, if ADC_HCn[AIEN] is high at the time that COCOn is set and if DMAEN is set, DMA request is asserted, if COCOn is set. Both the requests get deasserted when COCOn is low, cleared, which happens when data is read.

In all modes a blocking mechanism prevents a new result from overwriting previous data in ADC_Rn, if the previous data is in the process of being read. When blocking is active (OVWREN=0 in ADC_CFG), the conversion result data transfer is blocked, COCOn is not set, and the new result is lost. In all other cases of operation, when a conversion result data transfer is blocked, another conversion is initiated regardless of the state of ADCO (single or continuous conversions enabled).

Note

If continuous conversions are enabled, the blocking mechanism could result in the loss of data occurring at specific timepoints. To avoid this issue, the data must be read in fewer cycles than an ADC conversion time, accounting for interrupt or software polling loop latency.

If single conversions are enabled, the blocking mechanism could result in several discarded conversions and excess power consumption. To avoid this issue, the data registers must not be read after initiating a single conversion until the conversion completes.

66.5.4.3 Aborting Conversions

Any conversion in progress is aborted when:

- The MCU enters stop mode with ADACK not enabled.
- In software trigger mode, write to ADC_HC0 register, while ADC_HC0 is actively (already) controlling a conversion, aborts the current conversion. Since none of the ADC_HC1 - ADC_HCn registers are used for software trigger operation, writing to any of them will not initiate a new conversion nor abort the software triggered active conversion.

- In hardware trigger mode, writing to any of the ADC_HC0 - ADC_HCn registers, while that specific register is actively controlling a conversion, will abort the current conversion.
- A write to any ADC register other than the ADC_HC0: ADC_HCn registers occurs. This indicates a mode of operation change has occurred and the current conversion is therefore invalid.

Note

When a conversion is aborted, the contents of the data result registers, ADCRn are not altered. The data result registers continue to hold the values, transferred after the completion of the last successful conversion. If the conversion is aborted by a reset or stop (not operated with internal ADACK), ADCRn (data result register) return to their reset states.

66.5.4.4 Power Control

The ADC module remains in its idle state until a conversion is initiated. If ADACK is selected as the conversion clock source but the asynchronous clock output is disabled (ADACKEN=0), the ADACK clock generator will also remain in its idle state (disabled) until a conversion is initiated. If the asynchronous clock output is enabled (ADACKEN=1), it will remain active regardless of the state of the ADC or the MCU power mode.

Power consumption when the ADC is active can be reduced by setting ADLPC.

66.5.4.5 Sample Time and Total Conversion Time

The total conversion time depends upon the following:

- the sample phase time (as determined by ADLSMP and ADSTS bits in ADC_CFG register),
- the compare phase time (determined by MODE bits)
- the frequency of the conversion clock (f_{ADCK}).
- the MCU bus frequency (for Handshaking and selection of clock)

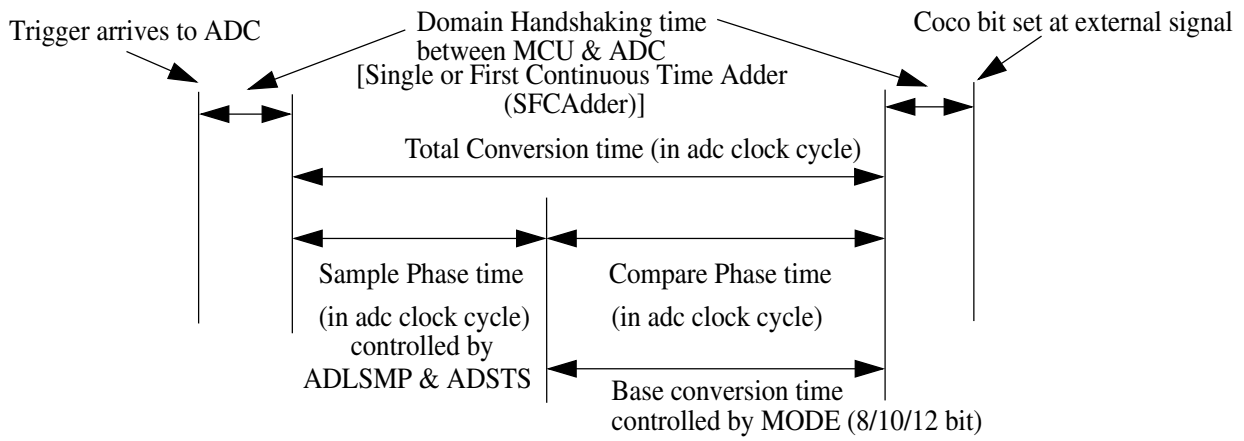


Figure 66-4. ADC conversion time details

After the module becomes active, sampling of the input begins. ADLSMP and ADSTS decide the sample time duration. When sampling is complete, the converter is isolated from the input channel and a successive approximation algorithm is performed to determine the digital value of the analog signal. The result of the conversion is transferred to ADC_Rn upon completion of the conversion algorithm.

If the bus frequency is less than the f_{ADCK} frequency, precise sample time for continuous conversions cannot be guaranteed .

The maximum total conversion time is determined by the clock source chosen and the divide ratio selected. The clock source is selectable by the ADICLK bits in ADC_CFG register, and the divide ratio is specified by the ADIV bits.

The maximum total conversion time for all configurations is summarized in Equation 1 on page 3414. Refer to Table 66-4 through Table 66-7 for the variables referenced in the equation.

$$\text{ConversionTime} = \text{SFCAdder} + \text{AverageNum} \times (\text{BCT} + \text{LSTAdder})$$

Equation 1. Equation of Conversion Time

Table 66-4. Single or First Continuous Time Adder (SFCAdder)

ADACKEN	ADICLK	Single or First Continuous Time Adder (SFCAdder)
x	0x, 10	3 ADCK cycles (before starting of conversion) + 1ADCK (after end of conversion) + 2 bus clock cycles
1	11	3 ADCK cycles (before starting of conversion) + 1 ADCK (after end of conversion) + 2 bus clock cycles
0	11	1.5µs +

Table 66-4. Single or First Continuous Time Adder (SFCAdder)

ADACKEN	ADICLK	Single or First Continuous Time Adder (SFCAdder)
		3 ADCK cycles (before starting of conversion) + 1 ADCK (after end of conversion) + 2 bus clock cycles

Table 66-5. Average Number Factor (AverageNum)

AVGE	AVGS[1:0]	Average Number Factor (AverageNum)
0	xx	1
1	00	4
1	01	8
1	10	16
1	11	32

Table 66-6. Base Conversion Time (BCT) (compare phase duration)

Mode	Base Conversion Time (BCT) (compare phase duration)
8 bit	17 ADCK cycles
10 bit	21 ADCK cycles
12 bit	25 ADCK cycles

Table 66-7. Long Sample Time

ADLSMP	ADSTS	Long Sample Time Adder (LSTAdder)
0	00	3 ADCK cycles
0	01	5 ADCK cycles
0	10	7 ADCK cycles (default)
0	11	9 ADCK cycles
1	00	13 ADCK cycles
1	01	17 ADCK cycles
1	10	21 ADCK cycles
1	11	25 ADCK cycles

Note

The ADCK frequency must be between f_{ADCK} minimum and f_{ADCK} maximum to meet ADC specifications.

66.5.4.6 Conversion Time Examples

The following examples uses Equation 1 on page 3414 and the information provided in tables Table 66-4 through Table 66-7.

66.5.4.6.1 Typical conversion time configuration

A typical configuration for ADC conversion is: 10-bit mode, with the bus clock selected as the input clock source, the input clock divide-by-1 ratio selected, and a bus frequency of 40 MHz, ADLSMP=0,ADLSTS=10 and high speed conversion disabled. The conversion time for a single conversion is calculated by using Equation 1 on page 3414 and the information provided in Table 66-8 through Table 66-10. The table below list the variables of Equation 1 on page 3414.

Table 66-8. Typical Conversion Time

Variable	Time
SFCAdder	4 ADCK cycles + 2 bus clock cycles
AverageNum	1
BCT	21 ADCK cycles
LSTAdder	7

The resulting conversion time is generated using the parameters listed in Table 66-8. So for Bus clock equal to 40 Mhz and ADCK equal to 40 Mhz the resulting conversion time is 0.85 us.

66.5.4.6.2 Long conversion time configuration

A configuration for long ADC conversion is: 12-bit mode, with the bus clock selected as the input clock source, the input clock divide-by-8 ratio selected, and a bus frequency of 40 MHz, long sample time enabled (ADLSMP=1, ADSTS=11) and configured for longest adder and high speed conversion disabled. Average enabled for 32 conversions (AVGE=1, AVGS=11). The conversion time for this conversion is calculated by using equation on Sample Time and Total Conversion Time and the information provided in Table 66-4 through Table 66-7. The table below lists the variables of equation.

Table 66-9. Typical Conversion Time

Variable	Time
SFCAdder	4 ADCK cycles + 2 bus clock cycles
AverageNum	32
BCT	25 ADCK cycles
LSTAdder	25 ADCK cycles

The resulting conversion time is generated using the parameters listed in Table 66-9. So for Bus clock equal to 40 Mhz and ADCK equal to 5 Mhz the resulting conversion time is 10.0226 us (AverageNum). This results in a total conversion time of 320.85 us.

66.5.4.6.3 Short conversion time configuration

A configuration for short ADC conversion is: 8-bit mode, with the bus clock selected as the input clock source, the input clock divide-by-1 ratio selected, and a bus frequency of 40 MHz, long sample time disabled(ADLSMP=0, ADSTS=00) and high speed conversion enabled. The conversion time for this conversion is calculated by using the equation and the information provided in Table 66-4 to Table 66-7. The table below list the variables of equation.

Table 66-10. Typical Conversion Time

Variable	Time
SFCAdder	4 ADCK cycles + 2 bus clock cycles
AverageNum	1
BCT	17 ADCK cycles
LSTAdder	3 ADCK cycles

The resulting conversion time is generated using the parameters listed in Table 66-10. So for Bus clock equal to 40Mhz and ADCK equal to 40Mhz the resulting conversion time is 650 ns.

66.5.4.7 Hardware Average Function

The hardware average function can be enabled (AVGE=1) to perform a hardware average of multiple conversions. The number of conversions is determined by the AVGS[1:0] bits, which select 4, 8, 16 or 32 conversions to be averaged. While the hardware average function is in progress the ADACT bit will be set.

After the selected input is sampled and converted, the result is placed in an accumulator from which an average is calculated after the selected number of conversions has been completed. When hardware averaging is selected the completion of a single conversion will not set the COCOn bit.

If the compare function is either disabled or evaluates true, after the selected number of conversions are completed, the average conversion result is transferred into the data result registers, ADC_Rn, and the COCOn bit is set. An ADC interrupt is generated upon the setting of COCOn if the respective ADC interrupt is enabled (AIENn=1).

66.5.5 Automatic Compare Function

The compare function can be configured to check if the result is less than or greater-than-or-equal-to a single compare value, or if the result falls within or outside a range determined by two compare values. The compare mode is determined by ACFGT, ACREN and the values in the compare value register (ADC_CV). After the input is sampled and converted, the compare values (CV1 and CV2) are used as described in the table below. There are six compare modes as shown in the table below.

Table 66-11. Compare Modes

ACFGT	ACREN	CV1 relative to CV2	Function	Compare Mode Description
0	0	-	Less than threshold	Compare true if the result is less than the CV1 registers.
1	0	-	Greater than or equal to threshold	Compare true if the result is greater than or equal to CV1 registers.
0	1	Less than or equal	Outside range, not inclusive	Compare true if the result is less than CV1 **Or** the result is Greater than CV2
0	1	Greater than	Inside range, not inclusive	Compare true if the result is less than CV1 **And** the result is greater than CV2
1	1	Less Than or equal	Inside range, inclusive	Compare true if the result is greater than or equal to CV1 **And** the result is less than or equal to CV2
1	1	Greater than	Outside range, inclusive	Compare true if the result is greater than or equal to CV1 **Or** the result is less than or equal to CV2

With the ADC range enable bit set, ADCREN =1, if compare value 1(CV1 value) is less than or equal to the compare value 2 (CV2 value), setting ACFGT will select a trigger-if-inside-compare-range, inclusive-of-endpoints function. Clearing ACFGT will select a trigger-if-outside-compare-range, not-inclusive-of-endpoints function.

If CV1 is greater than the CV2, setting ACFGT will select a trigger-if-outside-compare-range, inclusive-of-endpoints function. Clearing ACFGT will select a trigger-if-inside-compare-range, not-inclusive-of-endpoints function.

If the condition selected evaluates true, COCOn is set.

Upon completion of a conversion while the compare function is enabled, if the compare condition is not true, COCOn is not set and the conversion result data will not be transferred to the result register. If the hardware averaging function is enabled, the compare function compares the averaged result to the compare values. The same compare function definitions apply. An ADC interrupt is generated upon the setting of COCOn if the respective ADC interrupt is enabled (ADC_HCn[AIEN]=1).

Note

The compare function can monitor the voltage on a channel while the MCU is in wait or stop3 mode. The ADC interrupt wakes the MCU when the compare condition is met.

66.5.6 Calibration Function

The ADC contains a self-calibration function that is required to achieve the specified accuracy. Calibration should be run or valid calibration values should be written after power up and system reset (as the calibration register will be reset on reset assertion) with specified settings before any conversion is initiated. The calibration function sets the calibration value at the end of running the full calibration sequence in ADC_CAL register. The user must configure the ADC correctly prior to starting the calibration process, and must allow the process to run the full calibration sequence by checking the status of ADC_GC[CAL] and ADC_GS[CALF] so that the generated calibration value can be loaded.

Prior to calibration, the user must configure the ADC's clock source and frequency, low power configuration, voltage reference selection, sample time, averaging, and the high speed configuration according to the application's clock source availability and needs. If the application uses the ADC in a wide variety of configurations, the configuration for which the highest accuracy is required should be selected, or multiple calibrations can be done for the different configurations. The input channel, conversion mode, continuous function and compare function are all ignored during the calibration process.

To initiate calibration, the user sets the CAL bit and the calibration will automatically begin if the ADTRG bit = 0. If ADTRG = 1, the CAL bit will not get set and the calibration fail flag (CALF) will be set. While calibration is active, no ADC register can be written and no stop mode may be entered or the calibration routine will be aborted causing the CAL bit to clear and the CALF bit to set.

At the end of a calibration sequence the COCO[0] bit of the ADC_HS register will be set. The ADC_HCn[AIEN] bit can be used to allow an interrupt to occur at the end of a calibration sequence. If, at the end of calibration routine, the CALF bit is not set, the automatic calibration routine completed successfully.

To complete calibration, the user must follow the below procedure :

- Configure ADC_CFG with actual operating values for maximum accuracy.
- Configure the ADC_GC values along with CAL bit
- Check the status of CALF bit in ADC_GS and the CAL bit in ADC_GC
- When CAL bit becomes '0' then check the CALF status and COCO[0] bit status

When complete the user may reconfigure and use the ADC as desired.

A second calibration may also be performed if desired by clearing and again setting the CAL bit

Overall the calibration routine may take as many as 14000 ADCK cycles and 100 bus cycles, depending on the results and the clock source chosen.

66.5.7 User Defined Offset Function

The ADC Offset Correction Register (ADC_OFS) contains the user configured offset value. This register is 13 bit wide. The value in MSB (13th bit) is the operation bit, if this bit is ‘0’ then the value in rest 12 bit is added with the converted result value to generate final result to be loaded into ADC_Rn and if this bit is ‘1’ then this field is subtracted from converted value to generate final Result (ADC_Rn). If the Final result is above the maximum or below the minimum result value, it is forced to the appropriate limit for the current mode of operation. Forced to 0x0FFF if over and 0x0000 if lower for 12 bit mode.

The offset value has no effect during calibration on the final result.

The formatting of the ADC Offset Register is different from the Data Result Registers (ADC_Rn) to preserve the resolution of the value regardless of the conversion mode selected. Lower order bits are ignored in lower resolution modes. For example, in 8b single-ended mode, the bits OFS[11:4] are subtracted from D[7:0] when bit OFS[12] (sign bit) is ‘1’ ; indicates subtraction and bits OFS[4:0] are ignored. For 12b single-ended mode, bits OFS[11:0] are directly subtracted from the conversion result data CDATA[11:0] when OFS[12] (sign bit) is ‘1’. The similar is the addition operation when OFS[12](sign bit) is 0.

ADC_OFS is manually set according to user requirements after the self calibration sequence is done (CAL is cleared). The user have to write ADC_OFS with desired value.

NOTE

There is an effective limit to the values of Offset that can be set by the user. If the magnitude of the offset is too great the results of the conversions will cap off at the limits.

The offset function may be employed by the user to remove application offsets or DC bias values. An offset correction that results in an out-of-range value will be forced to the minimum or maximum value.

For applications which may change the offset repeatedly during operation, it is recommended to store the initial offset value in flash so that it can be recovered and added to any user offset adjustment value and the sum stored in the ADC_OFS registers.

66.5.8 MCU Wait Mode Operation

Wait mode is a **lower power-consumption standby mode** from which **recovery is fast** because **the clock sources remain active**. If a conversion is in progress when the MCU enters wait mode, it continues until completion. Conversions can be initiated while the MCU is in wait mode by means of the hardware trigger or if continuous conversions are enabled.

The bus clock, bus clock divided by two, and ADACK are available as conversion clock sources while in wait mode.

A conversion complete event sets the COCOn and generates an ADC interrupt to wake the MCU from wait mode.

If the compare and hardware averaging functions are disabled, a conversion complete event sets the COCOn and generates an ADC interrupt to wake the MCU from wait mode if the respective ADC interrupt is enabled (ADC_HCn[AIEN]=1).

If the hardware averaging function is enabled the COCOn will set (and generate an interrupt if enabled) when the selected number of conversions are complete.

If the compare function is enabled the COCOn will set (and generate an interrupt if enabled) only if the compare conditions are met.

If a single conversion is selected and the compare trigger is not met, the ADC will return to its idle state and cannot wake the MCU from wait mode unless a new conversion is initiated by the hardware trigger.

66.5.9 MCU Stop Mode Operation

Stop mode is a low power-consumption standby mode during which most or all clock sources on the MCU are disabled. Stop mode is entered when stop indication comes from the MCU.

66.5.9.1 Stop Mode With ADACK Disabled

If the asynchronous clock, ADACK, is not selected as the conversion clock, executing a stop instruction aborts the current conversion and places the ADC in its idle state. The contents of the ADC registers, including ADC_Rn are unaffected by stop mode. After exiting from stop mode, a software or hardware trigger is required to resume conversions.

66.5.9.2 Stop Mode With ADACK Enabled

If ADACK is selected as the conversion clock, the ADC continues operation during stop mode. For guaranteed ADC operation, the MCU's voltage regulator must remain active during stop mode.

If a conversion is in progress when the MCU enters stop mode, it continues until completion. Conversions can be initiated while the MCU is in stop mode by means of the hardware trigger or if continuous conversions are enabled.

A conversion complete event sets the COCOn and generates an ADC interrupt to wake the MCU from stop mode :

Note

The ADC module can wake the system from low-power stop and cause the MCU to begin consuming run-level currents without generating a system level interrupt. To prevent this scenario, software should ensure the conversion result data transfer blocking mechanism (discussed in Completing Conversions) is cleared when entering stop and continuing ADC conversions.

66.6 Initialization Information

This section gives an example that provides some basic direction on how to initialize and configure the ADC module. User can configure the module for 8, 10, 12 bit resolution, single or continuous conversion, and a polled or interrupt approach, among many other options.

66.6.1 ADC Module Initialization Example

This section describes the initialization sequence along with pseudo-code.

66.6.1.1 Initialization Sequence

Before the ADC module can be used to complete conversions, an initialization procedure must be performed. A typical sequence is as follows:

- Calibrate the ADC by following the calibration instructions in Calibration Function
- Update the configuration register (ADC_CFG) to select the input clock source and the divide ratio used to generate the internal clock, ADCK. This register is also used for selecting sample time and low-power configuration.
- Update General control register (ADC_GC) to select whether conversions will be continuous or completed only once (ADCO) and to select whether to perform hardware averaging, etc.
- Update Trigger control register (ADC_HCn) to select the conversion trigger (hardware or software, i.e. configure ADTRG bit) and compare function options, if enabled.

66.6.1.2 Pseudo-Code Example

In this example, the ADC module is set up with interrupts enabled to perform a single 10-bit conversion at low power with a long sample time on input channel 1, where the internal ADCK clock is derived from the bus clock divided by 1.

ADC_CFG

```
Bit 7    ADLPC   1      Configures for low power (lowers maximum clock speed.
Bit 6:5  ADIV    00     Sets the ADCK to the input clock  ÷  1.
Bit 4    ADLSMP  1      Configures for long sample time.
Bit 3:2  MODE    10     Sets mode at 10-bit conversions.
Bit 1:0  ADICLK  00     Selects bus clock as input clock source.
```

ADC_GC

```
Bit 7    CAL     0      Flag indicates if a conversion is in progress.
Bit 6    ADCO    0      Software trigger selected.
Bit 5    AVGE    0      Compare function disabled.
Bit 4    ACFE    0      Compare function disabled.
Bit 3    ACFGT   0      Not used in this example.
Bit 2    ACREN   0      Not used in this example.
Bit 1    DMAEN   0      Not used in this example.
Bit 0    ADACKEN 0      Not used in this example.
```

ADC_HC0

```
   Bit 7    AIEN   1      Conversion complete interrupt enabled.
Bit 4:0  ADCH   00001  Input channel 1 selected as ADC input channel.
```

ADC_R0

```
    Holds results of conversion. Read high byte (ADCRHA) before low byte (ADCRLA) so that
conversion data cannot be overwritten with data from the next conversion.
```

ADC_CV

```
    Holds compare values when compare function enabled.
```

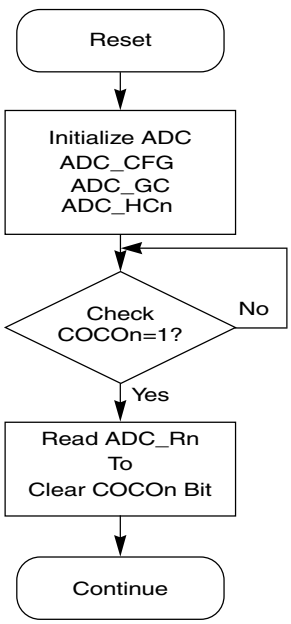


Figure 66-5. Initialization Flowchart for Example

66.7 Application Information

This section contains information for using the ADC module in applications. The ADC has been designed to be integrated into a microcontroller for use in embedded control applications requiring an A/D converter.

66.7.1 Sources of Error

Several sources of error exist for A/D conversions. These are discussed in the following sections.

66.7.1.1 Sampling Error

For proper conversions, the input must be sampled long enough to achieve the proper accuracy. Given the maximum input resistance of approximately 7kΩ and input capacitance of approximately 1.3 pF, sampling to within 1/4LSB (at 12-bit resolution) can be achieved within the nominal sample window (6 cycles @ 40 MHz maximum ADCK frequency) provided the resistance of the external analog source (R_{AS}) is kept below 4 kΩ.

Higher source resistances or higher-accuracy sampling is possible by setting ADLSMP and changing the ADSTS bits (to increase the sample window) or decreasing ADCK frequency to increase sample time.

66.7.1.2 Pin Leakage Error

Leakage on the I/O pins can cause conversion error if the external analog source resistance (R_{AS}) is high. If this error cannot be tolerated by the application, keep R_{AS} lower than $V_{DDAD}/ (2^N*I_{LEAK})$ for less than 1/4LSB leakage error (N = 8 in 8-bit, 10 in 10-bit or 12 in 12-bit mode).

66.7.1.3 Noise-Induced Errors

System noise that occurs during the sample or conversion process can affect the accuracy of the conversion. The ADC accuracy numbers are guaranteed as specified only if the following conditions are met:

- There is a 0.1 μF low-ESR capacitor from V_{REFH} to V_{REFL}.
- There is a 0.1 μF low-ESR capacitor from V_{DDAD} to V_{SSAD}.
- If inductive isolation is used from the primary supply, an additional 1 μF capacitor is placed from V_{DDAD} to V_{SSAD}.
- V_{SSAD} (and V_{REFL}, if connected) is connected to V_{SS} at a quiet point in the ground plane.
- Operate the MCU in wait or stop mode before initiating (hardware triggered conversions) or immediately after initiating (hardware or software triggered conversions) the ADC conversion.

- For software triggered conversions, immediately follow the write to ADCSC1 with a wait instruction or stop instruction.
- For stop mode operation, select ADACK as the clock source. Operation in stop reduces V_{DD} noise but increases effective conversion time due to stop recovery.

- There is no I/O switching, input or output, on the MCU during the conversion.

There are some situations where external system activity causes radiated or conducted noise emissions or excessive V_{DD} noise is coupled into the ADC. In these situations, or when the MCU cannot be placed in wait or stop or I/O activity cannot be halted, these recommended actions may reduce the effect of noise on the accuracy:

- Place a 0.01 μF capacitor (CAS) on the selected input channel to V_{REFL} or V_{SS} (this improves noise issues, but affects the sample rate based on the external analog source resistance).
- Average the result by converting the analog input many times in succession and dividing the sum of the results. Four samples are required to eliminate the effect of a 1LSB, one-time error.
- Reduce the effect of synchronous noise by operating off the asynchronous clock (ADACK) and averaging. Noise that is synchronous to ADCK cannot be averaged out.

66.7.1.4 Code Width and Quantization Error

Note

This will remain the same as long as the result is rounded for 8 and 10-bit modes. If the result is truncated in 8/10b modes then they will match 12b mode where the quantization error is -1 to 0.

The ADC quantizes the ideal straight-line transfer function into 4096 steps (in 12-bit mode). Each step ideally has the same height (1 code) and width. The width is defined as the delta between the transition points to one code and the next. The ideal code width for an N bit converter (in this case N can be 8, 10 or 12), defined as 1LSB, is:

$$1\text{lsb} = (V_{REFH} - V_{REFL}) / 2^N$$

There is an inherent quantization error due to the digitization of the result. For 8-bit or 10-bit conversions the code transitions when the voltage is at the midpoint between the points where the straight line transfer function is exactly represented by the actual

transfer function. Therefore, the quantization error will be ± 1/2 lsb in 8- or 10-bit mode. As a consequence, however, the code width of the first (0x000) conversion is only 1/2 lsb and the code width of the last (0xFF or 0x3FF) is 1.5 lsb.

For 12-bit conversions the code transitions only after the full code width is present, so the quantization error is -1 lsb to 0 lsb and the code width of each step is 1 lsb.

66.7.1.5 Linearity Errors

The ADC may also exhibit non-linearity of several forms. Every effort has been made to reduce these errors but the system should be aware of them because they affect overall accuracy. These errors are:

- Zero-scale error (E_{ZS}) (sometimes called offset) — This error is defined as the difference between the actual code width of the first conversion and the ideal code width (1/2 lsb in 8-bit or 10-bit modes and 1 lsb in 12-bit mode). If the first conversion is 0x001, the difference between the actual 0x001 code width and its ideal (1 lsb) is used.
- Full-scale error (E_{FS}) — This error is defined as the difference between the actual code width of the last conversion and the ideal code width (1.5 lsb in 8-bit or 10-bit modes and 1LSB in 12-bit mode). If the last conversion is 0x3FE, the difference between the actual 0x3FE code width and its ideal (1LSB) is used.
- Differential non-linearity (DNL) — This error is defined as the worst-case difference between the actual code width and the ideal code width for all conversions.
- Integral non-linearity (INL) — This error is defined as the highest-value the (absolute value of the) running sum of DNL achieves. More simply, this is the worst-case difference of the actual transition voltage to a given code and its corresponding ideal transition voltage, for all codes.
- Total unadjusted error (TUE) — This error is defined as the difference between the actual transfer function and the ideal straight-line transfer function and includes all forms of error.

66.7.1.6 Code Jitter, Non-Monotonicity, and Missing Codes

Analog-to-digital converters are susceptible to three special forms of error. These are code jitter, non-monotonicity, and missing codes.

Code jitter is when, at certain points, a given input voltage converts to one of two values when sampled repeatedly. Ideally, when the input voltage is infinitesimally smaller than the transition voltage, the converter yields the lower code (and vice-versa). However, even small amounts of system noise can cause the converter to be indeterminate (between two codes) for a range of input voltages around the transition voltage. This range is normally around ±1/2 lsb in 8-bit or 10-bit mode, or around 2 lsb in 12-bit mode, and increases with noise.

This error may be reduced by repeatedly sampling the input and averaging the result. Additionally the techniques discussed in Noise-Induced Errors reduces this error.

Non-monotonicity is defined as when, except for code jitter, the converter converts to a lower code for a higher input voltage. Missing codes are those values never converted for any input value.

In 8-bit or 10-bit mode, the ADC is guaranteed to be monotonic and have no missing codes.

66.8 Memory map and register definition

The ADC-Digital contains 32-bit, word aligned, byte enables registers; byte or half word access are not supported. All configuration registers are accessible via 32-bit access bus Interface. Write access to reserved locations have no impact while read access to reserved locations always return 0.

NOTE

No protection or indication mechanism is available (for example, 32-bit access starting with address offset value 0x01 or 0x02 or 0x03). The ADC does not check for correctness of the programmed values in the registers and the programmer must ensure that correct values are being written.

ADC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400C_4000	Control register for hardware triggers (ADC1_HC0)	32	R/W	0000_001Fh	66.8.1/3430
400C_4004	Control register for hardware triggers (ADC1_HC1)	32	R/W	0000_001Fh	66.8.2/3432
400C_4008	Control register for hardware triggers (ADC1_HC2)	32	R/W	0000_001Fh	66.8.2/3432
400C_400C	Control register for hardware triggers (ADC1_HC3)	32	R/W	0000_001Fh	66.8.2/3432
400C_4010	Control register for hardware triggers (ADC1_HC4)	32	R/W	0000_001Fh	66.8.2/3432

Table continues on the next page...

ADC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400C_4014	Control register for hardware triggers (ADC1_HC5)	32	R/W	0000_001Fh	66.8.2/3432
400C_4018	Control register for hardware triggers (ADC1_HC6)	32	R/W	0000_001Fh	66.8.2/3432
400C_401C	Control register for hardware triggers (ADC1_HC7)	32	R/W	0000_001Fh	66.8.2/3432
400C_4020	Status register for HW triggers (ADC1_HS)	32	R (reads 0)	0000_0000h	66.8.3/3433
400C_4024	Data result register for HW triggers (ADC1_R0)	32	R	0000_0000h	66.8.4/3435
400C_4028	Data result register for HW triggers (ADC1_R1)	32	R	0000_0000h	66.8.5/3436
400C_402C	Data result register for HW triggers (ADC1_R2)	32	R	0000_0000h	66.8.5/3436
400C_4030	Data result register for HW triggers (ADC1_R3)	32	R	0000_0000h	66.8.5/3436
400C_4034	Data result register for HW triggers (ADC1_R4)	32	R	0000_0000h	66.8.5/3436
400C_4038	Data result register for HW triggers (ADC1_R5)	32	R	0000_0000h	66.8.5/3436
400C_403C	Data result register for HW triggers (ADC1_R6)	32	R	0000_0000h	66.8.5/3436
400C_4040	Data result register for HW triggers (ADC1_R7)	32	R	0000_0000h	66.8.5/3436
400C_4044	Configuration register (ADC1_CFG)	32	R/W	0000_0200h	66.8.6/3437
400C_4048	General control register (ADC1_GC)	32	R/W	0000_0000h	66.8.7/3440
400C_404C	General status register (ADC1_GS)	32	R/W	0000_0000h	66.8.8/3442
400C_4050	Compare value register (ADC1_CV)	32	R/W	0000_0000h	66.8.9/3443
400C_4054	Offset correction value register (ADC1_OFS)	32	R/W	0000_0000h	66.8.10/ 3444
400C_4058	Calibration value register (ADC1_CAL)	32	R/W	0000_0000h	66.8.11/ 3444
400C_8000	Control register for hardware triggers (ADC2_HC0)	32	R/W	0000_001Fh	66.8.1/3430
400C_8004	Control register for hardware triggers (ADC2_HC1)	32	R/W	0000_001Fh	66.8.2/3432
400C_8008	Control register for hardware triggers (ADC2_HC2)	32	R/W	0000_001Fh	66.8.2/3432
400C_800C	Control register for hardware triggers (ADC2_HC3)	32	R/W	0000_001Fh	66.8.2/3432
400C_8010	Control register for hardware triggers (ADC2_HC4)	32	R/W	0000_001Fh	66.8.2/3432
400C_8014	Control register for hardware triggers (ADC2_HC5)	32	R/W	0000_001Fh	66.8.2/3432
400C_8018	Control register for hardware triggers (ADC2_HC6)	32	R/W	0000_001Fh	66.8.2/3432
400C_801C	Control register for hardware triggers (ADC2_HC7)	32	R/W	0000_001Fh	66.8.2/3432
400C_8020	Status register for HW triggers (ADC2_HS)	32	R (reads 0)	0000_0000h	66.8.3/3433
400C_8024	Data result register for HW triggers (ADC2_R0)	32	R	0000_0000h	66.8.4/3435
400C_8028	Data result register for HW triggers (ADC2_R1)	32	R	0000_0000h	66.8.5/3436
400C_802C	Data result register for HW triggers (ADC2_R2)	32	R	0000_0000h	66.8.5/3436
400C_8030	Data result register for HW triggers (ADC2_R3)	32	R	0000_0000h	66.8.5/3436
400C_8034	Data result register for HW triggers (ADC2_R4)	32	R	0000_0000h	66.8.5/3436
400C_8038	Data result register for HW triggers (ADC2_R5)	32	R	0000_0000h	66.8.5/3436
400C_803C	Data result register for HW triggers (ADC2_R6)	32	R	0000_0000h	66.8.5/3436
400C_8040	Data result register for HW triggers (ADC2_R7)	32	R	0000_0000h	66.8.5/3436

Table continues on the next page...

ADC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400C_8044	Configuration register (ADC2_CFG)	32	R/W	0000_0200h	66.8.6/3437
400C_8048	General control register (ADC2_GC)	32	R/W	0000_0000h	66.8.7/3440
400C_804C	General status register (ADC2_GS)	32	R/W	0000_0000h	66.8.8/3442
400C_8050	Compare value register (ADC2_CV)	32	R/W	0000_0000h	66.8.9/3443
400C_8054	Offset correction value register (ADC2_OFS)	32	R/W	0000_0000h	66.8.10/ 3444
400C_8058	Calibration value register (ADC2_CAL)	32	R/W	0000_0000h	66.8.11/ 3444

66.8.1 Control register for hardware triggers (ADCx_HC0)

ADC_HC0 can be used for both software and hardware trigger mode. Other ADC_HC*n* (*n* = 1...) are for use only in hardware trigger mode. The ADC_HC0 to ADC_HC*n* (*n* = 1...) registers have identical fields, and are used to control ADC operation. At any one point in time, only one of the ADC_HC0 to ADC_HC*n* (*n* = 1...) registers is actively controlling ADC conversions. Updating ADC_HC0 while ADC_HC*n* (*n* = 1...) is actively controlling a conversion is allowed (and vice-versa for any of the ADC_HC*n* (*n* = 1...) registers). Writing ADC_HC0 while ADC_HC0 is actively controlling a conversion aborts the current conversion. In software trigger mode (ADTRG=0), writes to ADC_HC0 subsequently initiates a new conversion (if the ADCH bits are equal to a value other than all 1s). Similarly, writing any of the ADC_HC*n* (*n* = 1...) registers while that specific ADC_HC*n* register is actively controlling a conversion aborts the current conversion. ADC_HC*n* (*n* = 1...) register is not used for software trigger operation and therefore writes to any of them do not initiate a new conversion.

Address: Base address + 0h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								AIEN	0		ADCH				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

ADCx_HC0 field descriptions

Field	Description
31–8 Reserved	This read-only field is reserved and always has the value 0.
7 AIEN	Conversion Complete Interrupt Enable/Disable Control An interrupt is generated whenever ADC_HS[COCO0]=1 (conversion ADC_HC0 completed), provided the corresponding interrupt is enabled. 1 Conversion complete interrupt enabled 0 Conversion complete interrupt disabled
6–5 Reserved	This read-only field is reserved and always has the value 0.
ADCH	Input Channel Select This 5-bit field selects one of the input channels. The successive approximation converter subsystem is turned off when the channel select bits are all set (ADCH = 11111b). This feature allows for explicit disabling of the ADC and isolation of the input channel from all sources. Terminating continuous conversions this way prevents an additional single conversion from being performed. 00000-01111 External channels 0 to 15 See External Signals for more information 10000 External channel selection from ADC_ETC 10001-10111 Reserved 11000 Reserved. 11001 VREFSH = internal channel, for ADC self-test, hard connected to VRH internally 11010 Reserved. 11011 Reserved. 11100-11110 Reserved. 11111 Conversion Disabled. Hardware Triggers will not initiate any conversion.

66.8.2 Control register for hardware triggers (ADCx_HC*n*)

ADC_HC*n* (*n* = 1...) are for use only in hardware trigger mode. The ADC_HC0 to ADC_HC*n* registers have identical fields, and are used to control ADC operation. At any one point in time, only one of the ADC_HC0 to ADC_HC*n* registers is actively controlling ADC conversions. Updating ADC_HC0 while ADC_HC*n* is actively controlling a conversion is allowed (and vice-versa for any of the ADC_HC*n* registers). Writing any of the ADC_HC*n* registers while that specific ADC_HC*n* register is actively controlling a conversion aborts the current conversion. Any of the ADC_HC*n* (*n* = 1...) registers are not used for software trigger operation and therefore writes to any of them do not initiate a new conversion.

Address: Base address + 4h offset + (4d × i), where i=0d to 6d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								AIEN	0		ADCH				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

ADCx_HC*n* field descriptions

Field	Description
31–8 Reserved	This read-only field is reserved and always has the value 0.
7 AIEN	Conversion Complete Interrupt Enable/Disable Control An interrupt is generated whenever ADC_HS[COCO0]=1(conversion ADC_HC0 completed), provided the corresponding interrupt is enabled. 1 Conversion complete interrupt enabled 0 Conversion complete interrupt disabled
6–5 Reserved	This read-only field is reserved and always has the value 0.
ADCH	Input Channel Select This 5-bit field selects one of the input channels. The successive approximation converter subsystem is turned off when the channel select bits are all set (ADCH =11111b). This feature allows for explicit disabling of the ADC and isolation of the input channel from all sources. Terminating continuous conversions this way prevents an additional single conversion from being performed. 00000-01111 External channels 0 to 15 See External Signals for more information

Table continues on the next page...

ADCx_HCn field descriptions (continued)

Field	Description	
	10000	External channel selection from ADC_ETC
	10001-10111	Reserved
	11000	Reserved.
	11001	VREFSH = internal channel, for ADC self-test, hard connected to VRH internally
	11010	Reserved.
	11011	Reserved.
	11100-11110	Reserved.
	11111	Conversion Disabled. Hardware Triggers will not initiate any conversion.

66.8.3 Status register for HW triggers (ADCx_HS)

Bit 0 is used for both software and hardware trigger modes of operation. Bit 1 to bit (n-1) indicate the rest of the HW triggers' statuses similar to bit 0, potentially corresponding to multiple ADC_HC registers (for use only in hardware trigger mode).

Address: Base address + 20h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								COCO7	COCO6	COCO5	COCO4	COCO3	COCO2	COCO1	COCO0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADCx_HS field descriptions

Field	Description
31–8 Reserved	This read-only field is reserved and always has the value 0.
7 COCO7	See description for COCO1.
6 COCO6	See description for COCO1.
5 COCO5	See description for COCO1.
4 COCO4	See description for COCO1.
3 COCO3	See description for COCO1.
2 COCO2	See description for COCO1.
1 COCO1	Conversion Complete Flag See detailed description in COCO0, but for use only in hardware trigger mode. **NOTE:** • In the hardware trigger mode, when trigger comes from TSC, COCO bit can be cleared by writing TSC register. When other ADC_ETC trigger sources work as ADC hardware trigger, COCO bit is cleared automatically when each phase is done.
0 COCO0	Conversion Complete Flag The COCOn flag is a read-only bit that is set each time a conversion is completed when the compare function is disabled (ADC_GC[ACFE]=0) and the hardware average function is disabled (ADC_GC[AVGE]=0). When the compare function is enabled (ADC_GC[ACFE]=1), the COCOn flag is set upon completion of a conversion only if the compare result is true. When the hardware average function is enabled (ADC_GC[AVGE]=1), the COCOn flag is set upon completion of the selected number of conversions (determined by the ADC_CFG[AVGS] field). The COCO0 flag will also set at the completion of a Calibration and Test sequence. A COCOn bit is cleared when the respective ADC_HCn is written or when the respective ADC_Rn is read. **NOTE:** • In the hardware trigger mode, when trigger comes from TSC, COCO bit can be cleared by writing TSC register. When other ADC_ETC trigger sources work as ADC hardware trigger, COCO bit is cleared automatically when each phase is done. • In the software trigger mode, COCO0 bit is cleared when ADC_HC0 is written or when ADC_R0 is read.

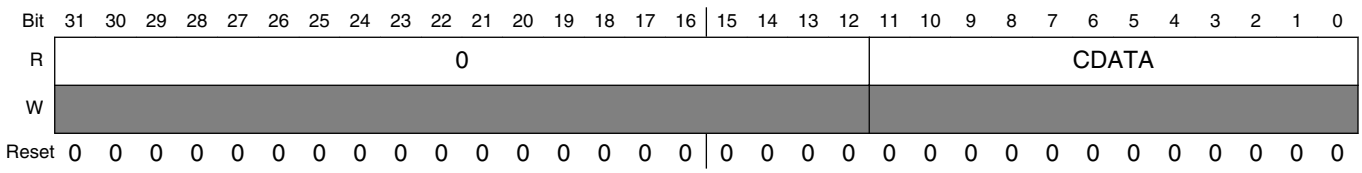
66.8.4 Data result register for HW triggers (ADCx_R0)

Contains the result of an ADC conversion of the channel selected by the respective hardware trigger and channel control register (ADC_HC0:ADC_HCn). For every ADC_HC0:ADC_HCn status and channel control register, there is a respective ADC_R0:ADC_Rn data result register. Unused bits in the ADC_Rn register are cleared in unsigned right justified modes. For example when configured for 10-bit single-ended mode, D[31:10] are cleared. The table below describes the behavior of the data result registers in the different modes of operation.

Table 66-12. Data Result Register Description

Conversion Mode	Data Result Register bits																Format
	D31	D30	.....	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	
12b single-ended	0	0	0	0	D	D	D	D	D	D	D	D	D	D	D	D	unsigned right justified
10b single-ended	0	0	0	0	0	0	D	D	D	D	D	D	D	D	D	D	unsigned right justified
8b single-ended	0	0	0	0	0	0	0	0	D	D	D	D	D	D	D	D	unsigned right justified

Address: Base address + 24h offset



ADCx_R0 field descriptions

Field	Description
31–12 Reserved	This read-only field is reserved and always has the value 0.
CDATA	Data (result of an ADC conversion)

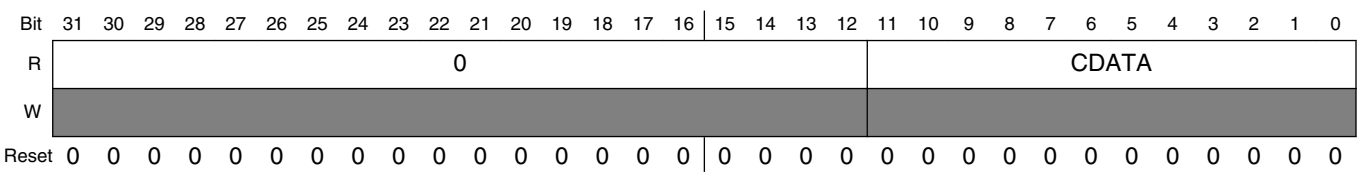
66.8.5 Data result register for HW triggers (ADC*x*_R*n*)

Contains the result of an ADC conversion of the channel selected by the respective Hardware Trigger and channel control register (ADC_HC0:ADC_HCn). For every ADC_HC0:ADC_HCn status and channel control register, there is a respective ADC_R0 to ADC_Rn data result register. Unused bits in the ADC_Rn register are cleared in unsigned right justified modes. For example when configured for 10-bit single-ended mode, D[31:10] are cleared. The table below describes the behavior of the data result registers in the different modes of operation.

Table 66-13. Data Result Register Description

Conversion Mode	Data Result Register bits																Format
	D31	D30	.....	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0	
12b single-ended	0	0	0	0	D	D	D	D	D	D	D	D	D	D	D	D	unsigned right justified
10b single-ended	0	0	0	0	0	0	D	D	D	D	D	D	D	D	D	D	unsigned right justified
8b single-ended	0	0	0	0	0	0	0	0	D	D	D	D	D	D	D	D	unsigned right justified

Address: Base address + 28h offset + (4d × i), where i=0d to 6d



ADC*x*_R*n* field descriptions

Field	Description
31–12 Reserved	This read-only field is reserved and always has the value 0.
CDATA	Data (result of an ADC conversion)

66.8.6 Configuration register (ADCx_CFG)

Selects the mode of operation, clock source, clock divide, configure for low power, long sample time, high speed configuration and selects the sample time duration.

Address: Base address + 44h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															OVWREN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	AVGS		ADTRG	REFSEL		ADHSC	ADSTS		ADLPC	ADIV		ADLSMP	MODE		ADICLK	
W																
Reset	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0

ADCx_CFG field descriptions

Field	Description
31–17 Reserved	This read-only field is reserved and always has the value 0.
16 OVWREN	Data Overwrite Enable Controls the overwriting of the next converted Data onto the existing (previous) unread data into the Data result register. 1 Enable the overwriting. 0 Disable the overwriting. Existing Data in Data result register will not be overwritten by subsequent converted data.
15–14 AVGS	Hardware Average select Determines how many ADC conversions will be averaged to create the ADC average result. This functionality is activated when ADC_GC[AVGE] = 1. 00 4 samples averaged 01 8 samples averaged 10 16 samples averaged 11 32 samples averaged
13 ADTRG	Conversion Trigger Select Selects the type of trigger used for initiating a conversion. Two types of trigger are selectable: software trigger and hardware trigger. When software trigger is selected, a conversion is initiated following a write to ADC_HC0. When hardware trigger is selected, a conversion is initiated following the assertion of a pulse on Alternate Hardware trigger input along with the assertion of the enable of respective the hardware Triggers input.

Table continues on the next page...

ADCx_CFG field descriptions (continued)

Field	Description
	0 Software trigger selected 1 Hardware trigger selected
12–11 REFSEL	Voltage Reference Selection Selects the voltage reference source used for conversions. 00 Selects VREFH/VREFL as reference voltage. 01 Reserved 10 Reserved 11 Reserved
10 ADHSC	High Speed Configuration This bit configures the ADC for high speed operation. The internal ADC clock is higher than normal. 0 Normal conversion selected. 1 High speed conversion selected.
9–8 ADSTS	Defines the **total** sample time duration **in number of full cycles**. This has two modes, short and long. When long sample time is selected (ADLSMP=1) this works for long sample time otherwise this works for short sample. This allows higher impedance inputs to be accurately sampled or to maximize conversion speed for lower impedance inputs. Longer sample times can also be used to lower overall power consumption when continuous conversions are enabled if high conversion rates are not required. **NOTE:** The actual sampling of capacitors happens 1/2 cycle less than the above values to prevent leakage of charge during switching from sample to conversion phase for better accuracy. 00 Sample period (ADC clocks) = 3 if ADLSMP=0b Sample period (ADC clocks) = 13 if ADLSMP=1b 01 Sample period (ADC clocks) = 5 if ADLSMP=0b Sample period (ADC clocks) = 17 if ADLSMP=1b 10 Sample period (ADC clocks) = 7 if ADLSMP=0b Sample period (ADC clocks) = 21 if ADLSMP=1b 11 Sample period (ADC clocks) = 9 if ADLSMP=0b Sample period (ADC clocks) = 25 if ADLSMP=1b
7 ADLPC	Low-Power Configuration Puts the ADC hard block into low power mode and reduces the comparator enable period by controlling its timing in the SAR controller block towards the analog hard block. The signal indicating low power mode to the Analog block is asserted when this bit is set. 0 ADC hard block not in low power mode. 1 ADC hard block in low power mode.
6–5 ADIV	Clock Divide Select Selects the divide ratio used by the ADC to generate the internal clock ADCK. 00 Input clock 01 Input clock / 2 10 Input clock / 4 11 Input clock / 8

Table continues on the next page...

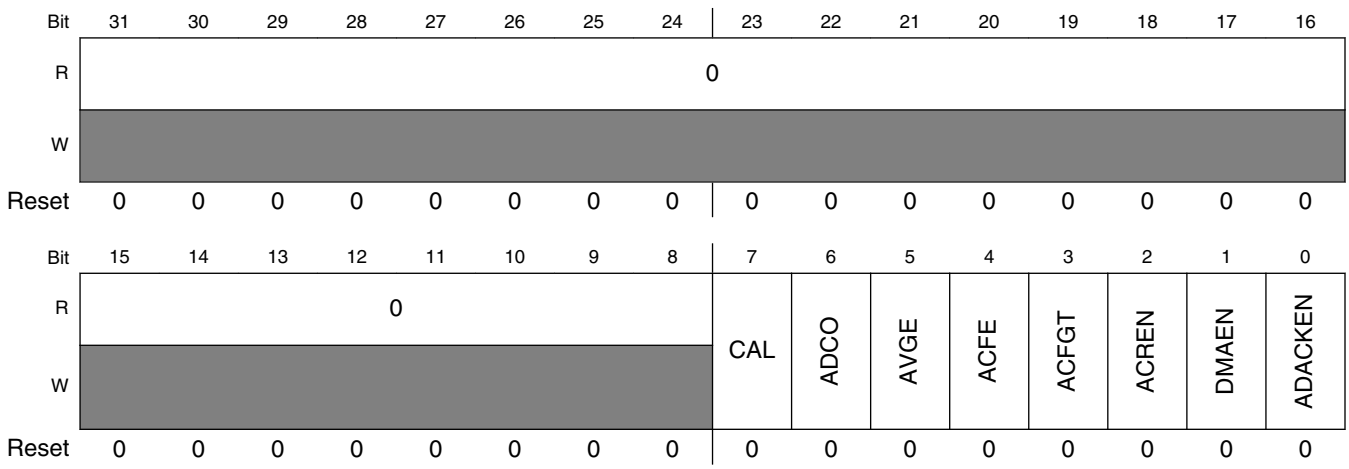
ADCx_CFG field descriptions (continued)

Field	Description
4 ADLSMP	Long Sample Time Configuration Selects between different sample times based on the ADC_CFG[ADSTS] field. This bit adjusts the sample period to allow higher impedance inputs to be accurately sampled or to maximize conversion speed for lower impedance inputs. If high conversion rates are not required, longer sample times can also be used to lower overall power consumption when continuous conversions are enabled. When ADLSMP=1, the Long Sample Time mode is selected and the time is defined by ADSTS[1:0] of the ADC_CFG register. 0 Short sample mode. 1 Long sample mode.
3–2 MODE	Conversion Mode Selection Used to set the ADC resolution mode. 00 8-bit conversion 01 10-bit conversion 10 12-bit conversion 11 Reserved
ADICLK	Input Clock Select Selects the input clock source to generate the internal clock ADCK. 00 IPG clock 01 IPG clock divided by 2 10 Reserved 11 Asynchronous clock (ADACK)

66.8.7 General control register (ADCx_GC)

Controls the calibration, continuous convert, hardware averaging functions, conversion active, hardware/software trigger select, compare function and voltage reference select of the ADC module.

Address: Base address + 48h offset



ADCx_GC field descriptions

Field	Description
31–8 Reserved	This read-only field is reserved and always has the value 0.
7 CAL	Calibration CAL begins the calibration sequence when set. This bit stays set while the calibration is in progress and is cleared when the calibration sequence is complete. The ADC_GS[CALF] bit must be checked to determine the result of the calibration sequence. After it has started, the calibration routine cannot be interrupted by writes to the ADC registers or the results will be invalid and the ADC_GS[CALF] bit will set. Setting the CAL bit will abort any current conversion.
6 ADCO	Continuous Conversion Enable Enables continuous conversions. 0 One conversion or one set of conversions if the hardware average function is enabled (AVGE=1) after initiating a conversion. 1 Continuous conversions or sets of conversions if the hardware average function is enabled (AVGE=1) after initiating a conversion.
5 AVGE	Hardware average enable Enables the hardware average function of the ADC. 0 Hardware average function disabled 1 Hardware average function enabled

Table continues on the next page...

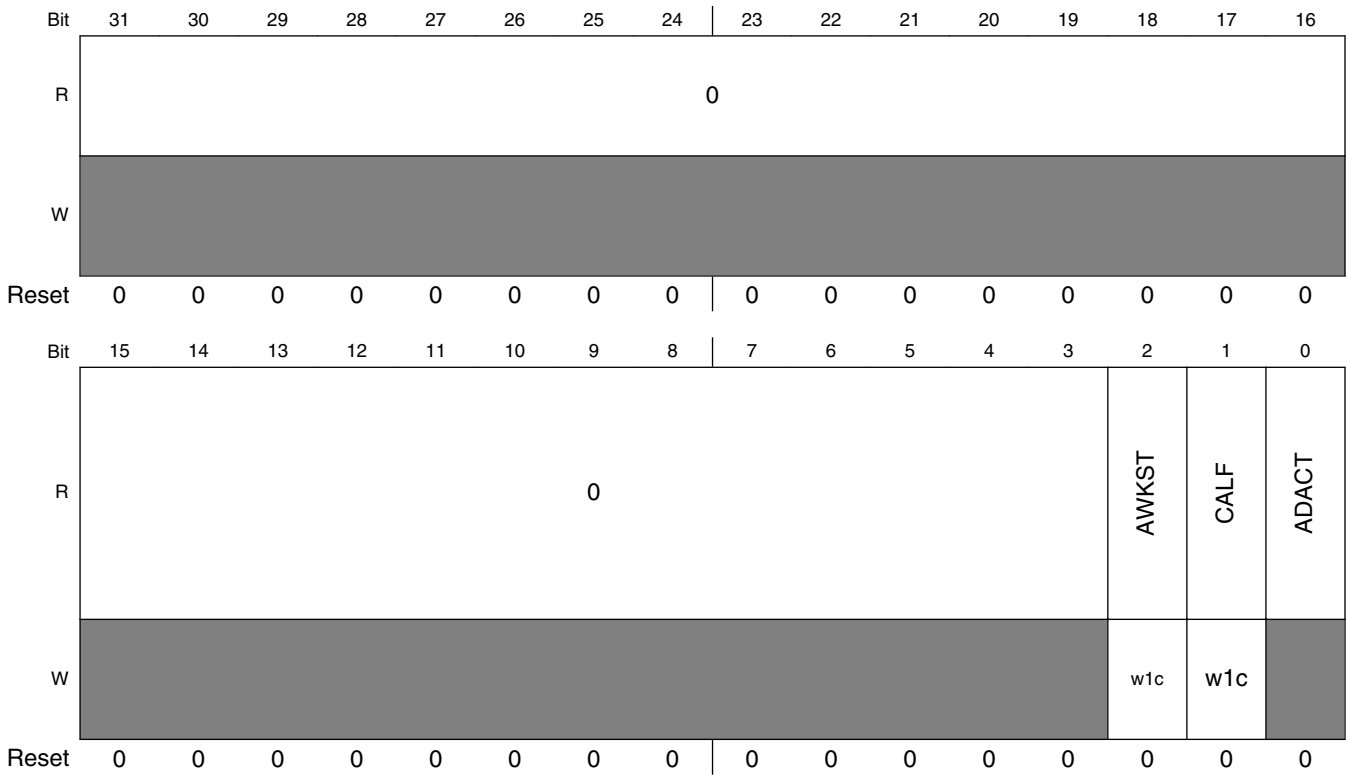
ADCx_GC field descriptions (continued)

Field	Description
4 ACFE	Compare Function Enable Enables the compare function. 0 Compare function disabled 1 Compare function enabled
3 ACFGT	Compare Function Greater Than Enable Configures the compare function to check the conversion result relative to the compare value register (ADC_CV) based upon the value of ACREN (bit 2 in ADC_GC register). The ACFE bit must be set for ACFGT to have any effect. 0 Configures "Less Than Threshold, Outside Range Not Inclusive and Inside Range Not Inclusive" functionality based on the values placed in the ADC_CV register. 1 Configures "Greater Than Or Equal To Threshold, Outside Range Inclusive and Inside Range Inclusive" functionality based on the values placed in the ADC_CV registers.
2 ACREN	Compare Function Range Enable Configures the compare function to check the conversion result of the input being monitored is either between or outside the range formed by the compare values in register (ADC_CV) determined by the value of ACFGT. The ACFE bit must be set for ACFGT to have any effect. 0 Range function disabled. Only the compare value 1 of ADC_CV register (CV1) is compared. 1 Range function enabled. Both compare values of ADC_CV registers (CV1 and CV2) are compared.
1 DMAEN	DMA Enable Enables the DMA logic. 0 DMA disabled (default) 1 DMA enabled
0 ADACKEN	Asynchronous clock output enable Enables the ADC's asynchronous clock source and the clock source output regardless of the conversion and input clock select (ADC_CFG[ADICLK]) settings of the ADC. Based on MCU configuration, the asynchronous clock may be used by other modules (see module introduction section). Setting this bit allows the clock to be used even while the ADC is idle or operating from a different clock source. Also, latency of initiating a single or first-continuous conversion with the asynchronous clock selected is reduced since the ADACK clock is already operational. 0 Asynchronous clock output disabled; Asynchronous clock only enabled if selected by ADICLK and a conversion is active. 1 Asynchronous clock and clock output enabled regardless of the state of the ADC

66.8.8 General status register (ADCx_GS)

Controls the calibration, continuous convert, hardware averaging functions, conversion active, hardware/software trigger select, compare function and voltage reference select of the ADC module.

Address: Base address + 4Ch offset



ADCx_GS field descriptions

Field	Description
31–3 Reserved	This read-only field is reserved and always has the value 0.
2 AWKST	Asynchronous wakeup interrupt status Holds the status of asynchronous interrupt status that occurred during stop mode. This bit is set when ipg_stop is deasserted and ipg_clk has started. It is cleared by writing '1' to it. Clearing this bit also deasserts the Asynchronous interrupt to CPU. 1 Asynchronous wake up interrupt occurred in stop mode. 0 No asynchronous interrupt.
1 CALF	Calibration Failed Flag

Table continues on the next page...

ADCx_GS field descriptions (continued)

Field	Description
	Displays the result of the calibration sequence. The calibration sequence will fail if Hardware Trigger is selected (i.e. ADC_CFG[ADTRG] = 1), or any ADC register is written, or any stop mode is entered before the calibration sequence completes. The CALF bit is cleared by writing a 1 to it. 0 Calibration completed normally. 1 Calibration failed. ADC accuracy specifications are not guaranteed.
0 ADACT	Conversion Active Indicates that a conversion or hardware averaging is in progress. ADACT is set when a conversion is initiated and cleared when a conversion is completed or aborted. 0 Conversion not in progress. 1 Conversion in progress.

66.8.9 Compare value register (ADCx_CV)

Contains compare values used to compare with the conversion result when the compare function is enabled (ADC_GC[ACFE]=1). The compare values are right justified. Therefore, the compare function only uses the compare value register bits that are related to the ADC mode of operation. (e.g. in 8 bit mode, CV1 = ADC_CV[7:0] and CV2 = ADC_CV[23:16], similarly in 10 bit mode, CV1 = ADC_CV[9:0] and CV2 = ADC_CV[25:16] etc.) The compare value 2 in this register is utilized only when the compare range function is enabled (ADC_GC[ACREN]=1).

Address: Base address + 50h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0				CV2												0				CV1											
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADCx_CV field descriptions

Field	Description
31–28 Reserved	This read-only field is reserved and always has the value 0.
27–16 CV2	Compare Value 2 Contains a compare value used to compare with the conversion result when the compare function and compare range function are enabled (ADC_GC[ACFE]=1, ADC_GC[ACREN]=1).
15–12 Reserved	This read-only field is reserved and always has the value 0.
CV1	Compare Value 1 Contains a compare value used to compare with the conversion result when the compare function is enabled (ADC_GC[ACFE]=1).

66.8.10 Offset correction value register (ADCx_OFS)

Contains the user-defined offset error correction value. This register is 13 bits wide. The value in the most significant bit (13th bit) is the operation bit. If this bit is ‘0’ then the value in the other 12 bits is added with the converted result value to generate final result to be loaded into ADC_Rn; if this bit is ‘1’ then this field is subtracted from converted value to generate final result (ADC_Rn).

Address: Base address + 54h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0			SIGN	OFS											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADCx_OFS field descriptions

Field	Description
31–13 Reserved	This read-only field is reserved and always has the value 0.
12 SIGN	Sign bit 0 The offset value is added with the raw result 1 The offset value is subtracted from the raw converted value
OFS	Offset value User configurable offset value.

66.8.11 Calibration value register (ADCx_CAL)

Contains calibration information that is generated by the calibration function. This register contains a calibration value of four bits(CAL[3:0]); this is automatically set after the self calibration sequence is done (ADC_SC[CAL] bit is cleared). If this register is written to by the user after calibration, the linearity error specifications may not be met.

Address: Base address + 58h offset

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0																CAL_CODE															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

ADCx_CAL field descriptions

Field	Description
31–4 Reserved	This read-only field is reserved and always has the value 0.
CAL_CODE	Calibration Result Value This value is automatically loaded and updated at the end of calibration.

Chapter 67
ADC External Trigger Control (ADC_ETC)

67.1 Chip-specific ADC_ETC information

Table 67-1. Reference links to related information

Topic	Related module(s)	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
XBAR resource assignments	XBAR	XBAR Resource Assignments
Touch Screen Control	TSC	TSC

67.2 Overview

The ADC External Trigger Control (ADC_ETC) module enables multiple trigger sources to share the ADC modules in a TIME-Division-Multiplexing (TDM) way. The external triggers can be from the Cross BAR (XBAR) and TSC (Touch Screen Controller) in SOC . The ADC_ETC has one TSC external trigger and 8 external triggers input from XBAR. The TSC external trigger is shared by TSC0 and TSC1. The TRIG0-TRIG3 input from XBAR together with TSC0 belong to channel0, while TRIG4-TRIG7 together with TSC1 belong to channel1. Channel0 controls ADC1 and Channel1 controls ADC2. When CTRL[TSC_BYPASS] is set, TSC external trigger will control ADC2 directly. The ADC_ETC also supports SyncMode. When TRIGa_CTRL[SYNC_MODE] is set, the

ADC_ETC trigger source will control ADC1 and ADC2 synchronously. The SyncMode can not be used when TSC_BYPASS is active. The ADC_ETC can support interrupt mode and DMA mode (controlled independently).

NOTE

The trigger in this chapter refers the trigger of ADC_ETC, except when referring to the ADC trigger explicitly.

67.2.1 Block diagram

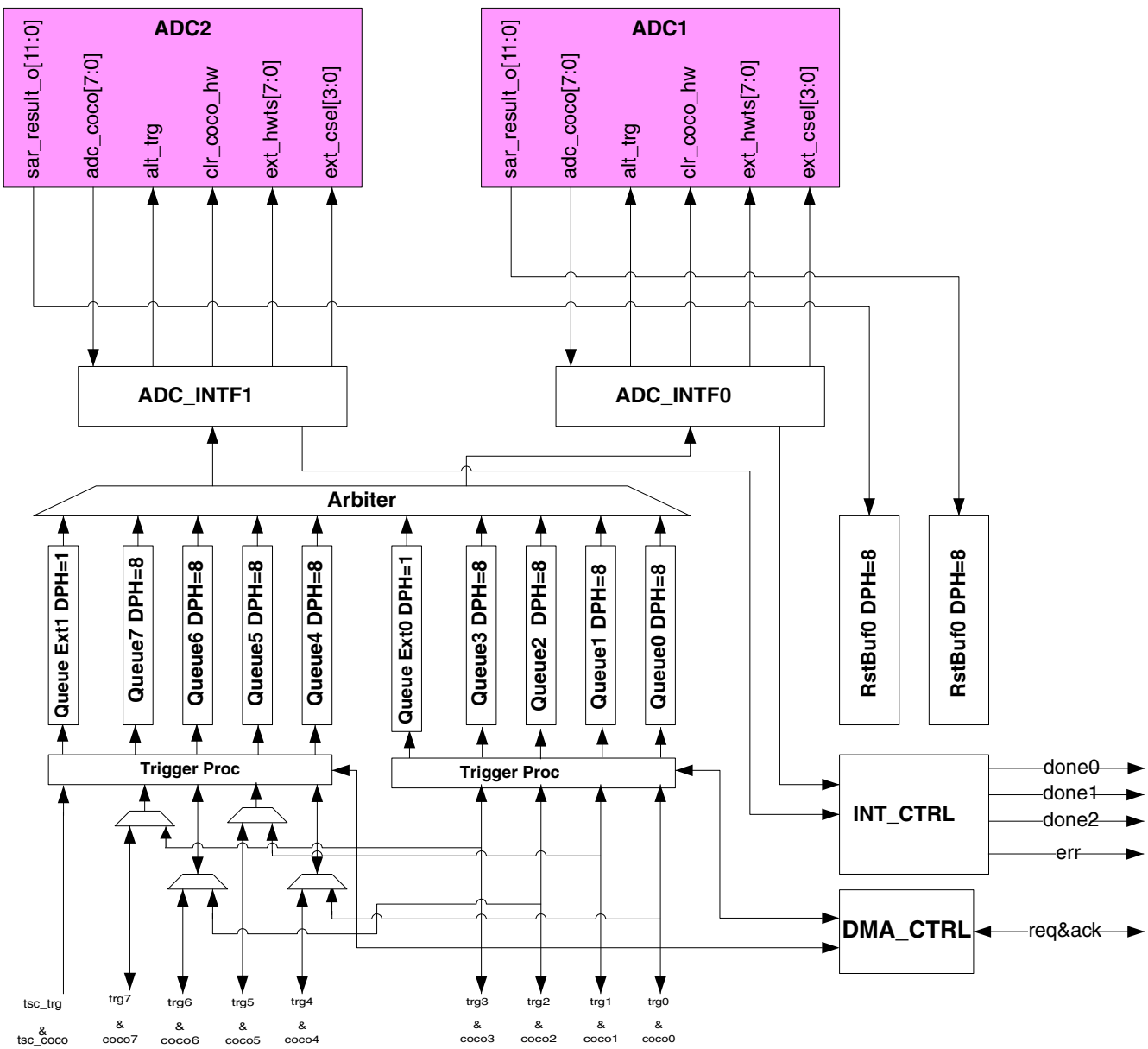


Figure 67-1. ADC_ETC block diagram

67.2.2 Features

The ADC_ETC includes the following features:

- External Trigger Interface
  - ADC control interface is connected with 2 ADCs, and support up to 8 Hardware External Trigger (ext_hwts[7:0] in ADC_ETC block diagram) for each ADC
- Capable of triggering dual ADCs in SyncMode or AsyncMode:
  - In SyncMode ADC1 and ADC2 are controlled by the same trigger source of ADC_ETC.
  - In AsyncMode ADC1 and ADC2 are controlled by separate trigger source of ADC_ETC.
- Support up to 8 external trigger inputs, input from XBAR (4 for each ADC, TRIG0-TRIG3 for ADC1, and TRIG4-TRIG7 for ADC2).
  - One external trigger of ADC_ETC can launch up to 8 sequential triggers to ADC, which are called a trigger chain, and each trigger is called a segment of the trigger chain.
  - Flexible ADC trigger interval and initial delay control.
  - Each trigger of ADC_ETC can be configured as HW or SW trigger mode.
- ADC result holding and status reporting
- ADC_ETC trigger auto hold and arbitration
  - Each trigger of ADC_ETC can be configured with a fixed priority. External trigger with the highest priority is severed first.
  - Hold one trigger event upon when it failed to win the arbitration or ADC is busy.
- Supports ADC trigger interface cascading
- Supports interrupt mode. When interrupt mode is selected and one ADC conversion is done, an interrupt request signal will be generated on one of the Done0~Done2 interrupt outputs.
- Supports DMA mode. When DMA is enabled and one ADC conversion is done, a DMA request will be sent. 2 trigger modes of DMA can be selected by configuring CTRL[DMA_MODE_SEL].

67.3 Functional description

The ADC_ETC block works with XBAR and TSC blocks to share with the ADC modules. The following sections describe functional details of the ADC_ETC module.

67.3.1 Operations

This section describes the ADC_ETC module's operations.

The trigger process is used to handle the multiple trigger sources based on the settings in registers. CTRL[TRIG_ENABLE] can be used to mask the input hardware trigger sources. TRIGa_CTRL[TRIG_MODE] selects the hardware trigger source or software trigger source for TRIGa of ADC_ETC. The selected trigger source is double synchronized to a synchronous positive pulse and the pulse will be sent to the arbiter after initial delay defined by TRIGa_COUNTER[INIT_DELAY].

If TRIGa_CTRL[SYNC_MODE] is set, the trigger process will control TRIGa and TRIG(a+4) synchronously. In this mode, the initial delay of TRIGa and TRIG(a+4) is controlled by the settings of TRIGa, while other settings are independent.

The trigger process diagram is indicated as below:

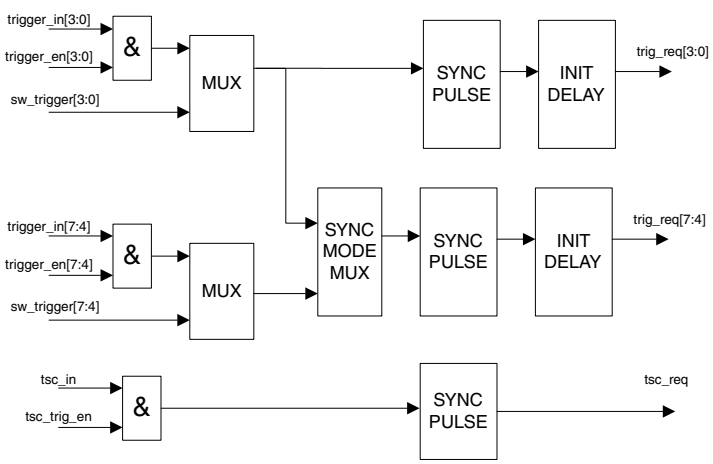


Figure 67-2. Trigger process diagram

As showing in ADC_ETC block diagram, the trig_req[7:0] output from trigger process will go to the arbiter with a queue module.

The arbiter trigger will approve the request from XBAR and TSC which has the highest priority defined in TRIGa_CTRL[TRIG_PRIORITY]. There are total of eight priority levels available for each trigger request, where 7 is the highest priority level and 0 is the lowest priority level. If multiple trigger requests have the same priority level, then the round-robin mechanism will be applied, and the trigger with larger number owns the higher priority. The trigger request is granted only when the previous trigger has finished and the ADC module is IDLE. When in SYNC mode, the trigger request is granted only when both ADC modules are IDLE state.

When there is a trigger request that is granted, the ADC_ETC sends several triggers of ADC as a chain of trigger, with ext_hwts[7:0] as ADC trigger selection signal, and with tcmd[3:0] defining the TCMD to ADC. When the ADC conversion done, the ADC sends back adc_coco to the ADC_ETC, which has the same width as ext_hwts and indicates which trigger the current sar_result_o is for, then the ADC_ETC stores the ADC converted results into the internal registers .

At the same time, interrupt asserts according to the software configuration. After that, the next trigger is sent after a interval delay configured by TRIGa_COUNTER[SAMPLE_INTERVAL], until all triggers in the chain are processed. When complete the the last trigger in chain, ADC_ETC sends the coco signal to mark the end of one of ADC_ETC external trigger, and a DMA request will be generated if DMA_CTRL[TIRGa_ENABLE] is set to 1.

Each ADC trigger in the trigger chains, initiated by trigger requests of ADC_ETC, has separate configurations by software. These configurations include Interrupt selection (IE), Back to Back ADC trigger (B2B), HWTS and CSEL to ADC. The ADC_ETC have interrupt outputs of Done0, Done1, Done2 and Error Interrupt. The Done0/1/2 interrupts are controlled by software for each ADC trigger. IEx selects which port the interrupt occurs on. (IEx=2'b00 means no interrupt, while 2'b01~2'b11 mean selecting Done0~Done2 correspondingly). Error interrupt occurs when there is an external trigger ignored by ADC_ETC due to the previous trigger not finished yet.

The ETC-ADC interface timing is indicated in below diagrams:

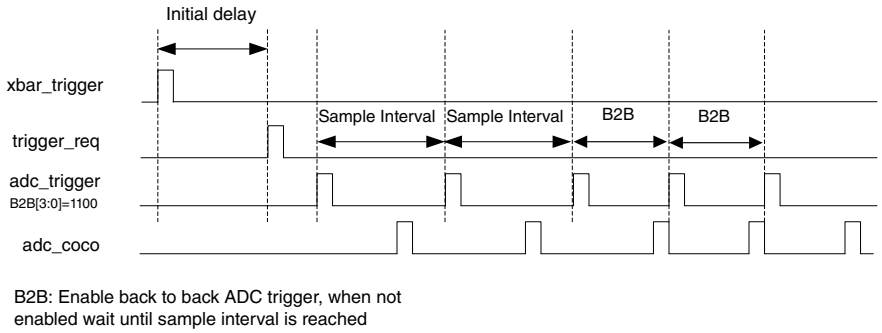


Figure 67-3. ETC-ADC interface timing

After one ADC conversion, the next trigger in the chain starts as soon as possible if B2B is enabled. Otherwise the next trigger does not start until waiting until interval delay in reached.

As showing in ETC-ADC interface timing, due to B2B[3:0]=1100, the slots after the first and second adc_trigger are equal to interval delay, but the slots after the third and fourth adc_trigger are shorter.

The length of initial delay and interval delay are configurable, the formulas to calculate initial delay and interval delay are described in the description of the TRIGa_COUNTER.

67.3.2 Clocks

The ADC_ETC module has one global functional clock: ipg_clk. The ipg_clk is used for register read/write operations, also all the functionality inside the ADC_ETC module are synchronized to this clock.

The XBAR and TSC input may be synchronous or asynchronous with the ADC_ETC clock.

67.3.3 Reset

The ADC_ETC has two resets: ipg_hard_async_reset_b and sw_reset_n. The ipg_hard_async_reset_b is a hardware reset, which resets all registers in ADC_ETC block. The sw_reset_n is a software reset, which can reset all registers except itself by software.

NOTE

sw_reset_n is generated by writting logic 1 into CRTL[SOFTRST], while ipg_hard_async_reset_b is generated by SoC hardware.

67.4 Initialization

Before using the ADC_ETC module, the following initialization procedure must be performed:

1. Configure the Global Control bits, include selecting DMA mode, enabling triggers which will be used, and setting pre-division factor.
2. Set DMA enable bits as needed.
3. Configure control bits for each enabled trigger, include enabling or disabling synchronization mode, assigning trigger priority, setting the length of trigger chain, selecting trigger mode and setting the initial and interval delay of this trigger.
4. According to the length of trigger chain, several segments of the chain should be configured. The configurations of each segment contain:
    a. setting the DONE interrupt for this chain, enabling or disabling B2B mode;
    b. selecting which trigger of ADC will be triggered, setting the ADC command.

After the initialization of ADC_ETC, the initialization of the ADC module should also be confirmed before launching a trigger to ADC_ETC.

67.5 Application information

This section describes applications supported by the ADC_ETC module.

67.5.1 ADC Trigger Concept – Dual PMSM motor control

ADC_ETC's flexible configuration is suitable for complex application timing design. The following describes the configuration scheme of ADC_ETC for dual motor control. There is a 180-degrees phase lag between the PWMs of dual motors, and on this basis ADC_ETC uses the triggers from eFlexPWM to implement time division sampling of analog signals of dual motors.

The figure below presents the synchronization between ADC and PWMs.

Once eFlexPWM1 counter reaches submodule 1 VAL4, ADC_ETC will be triggered to control ADCs to sampling analog signal of motor 1. After ADC conversation done, ADC_ETC_DONE0 interrupt is enabled to run motor 1 control algorithm. Once eFlexPWM2 counter reaches submodule 0 VAL4, ADC_ETC will be triggered to control ADCs to sampling analog signal of motor 2. After ADC conversation done, ADC_ETC_DONE1 interrupt is enabled to run motor 2 control algorithm.

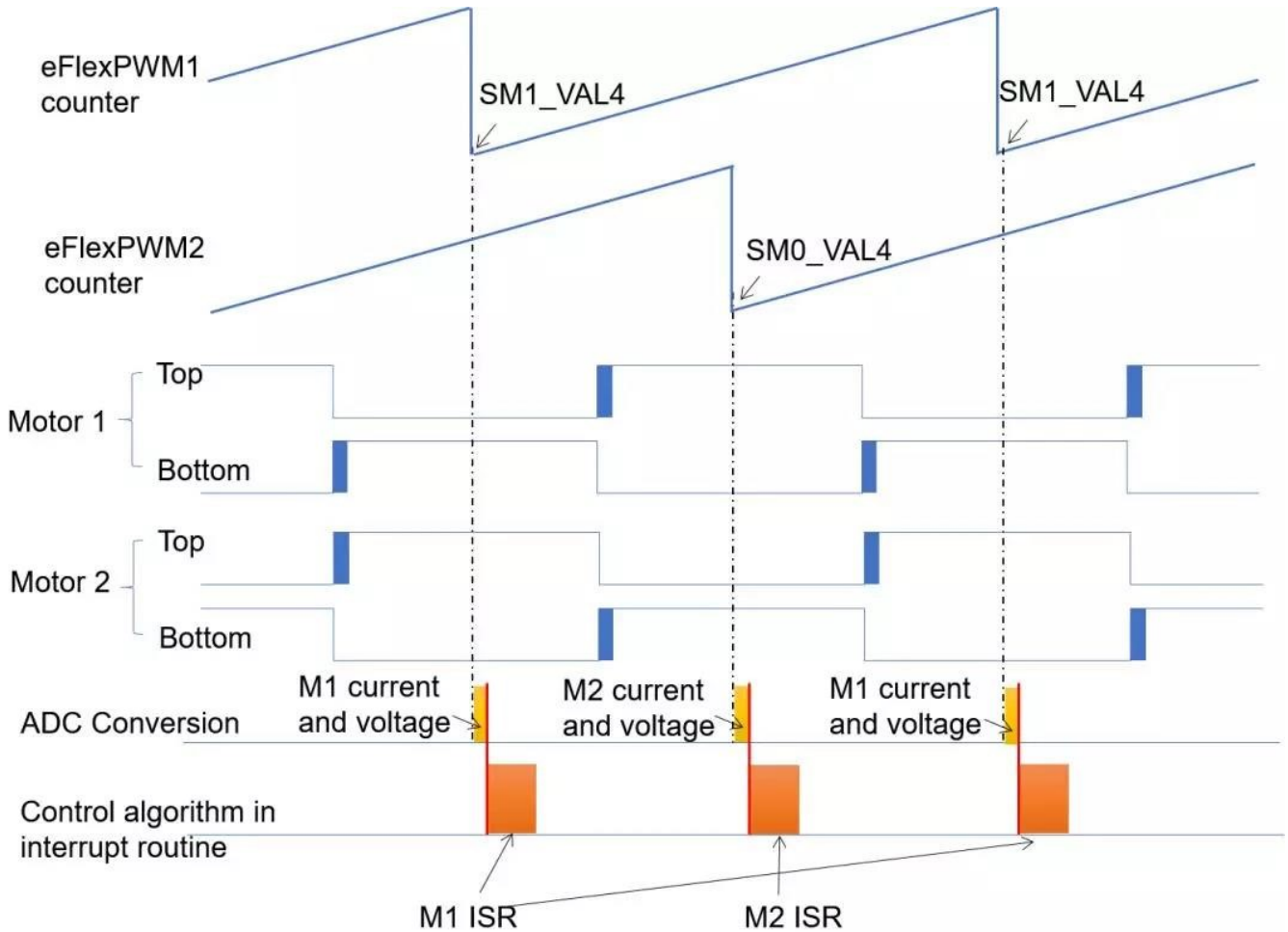


Figure 67-4. Synchronization between ADC and PWMs

The external triggers are generated from the Cross BAR (XBAR) or other sources. The ADC scan is started via ADC_ETC. The following is the key configuration point for this application.

- Enable external XBAR trigger 0 and 1, both ADCs have their own trigger segments.
- The trigger chain length is set to 2. The back-to-back ADC trigger mode is enabled.

- Enable the Sync Mode. In the Sync Mode, ADC1 and ADC2 are controlled by the same trigger source.
- Both trigger chain has their own finished interrupt. When Chain 4 triggered by TRIG0 is finished, enable DONE0 interrupt. When Chain 5 triggered by TRIG1 is finished, enable DONE1 interrupt.

The figure below shows configuration of ADC_ETC to control dual ADCs in this dual motor application.

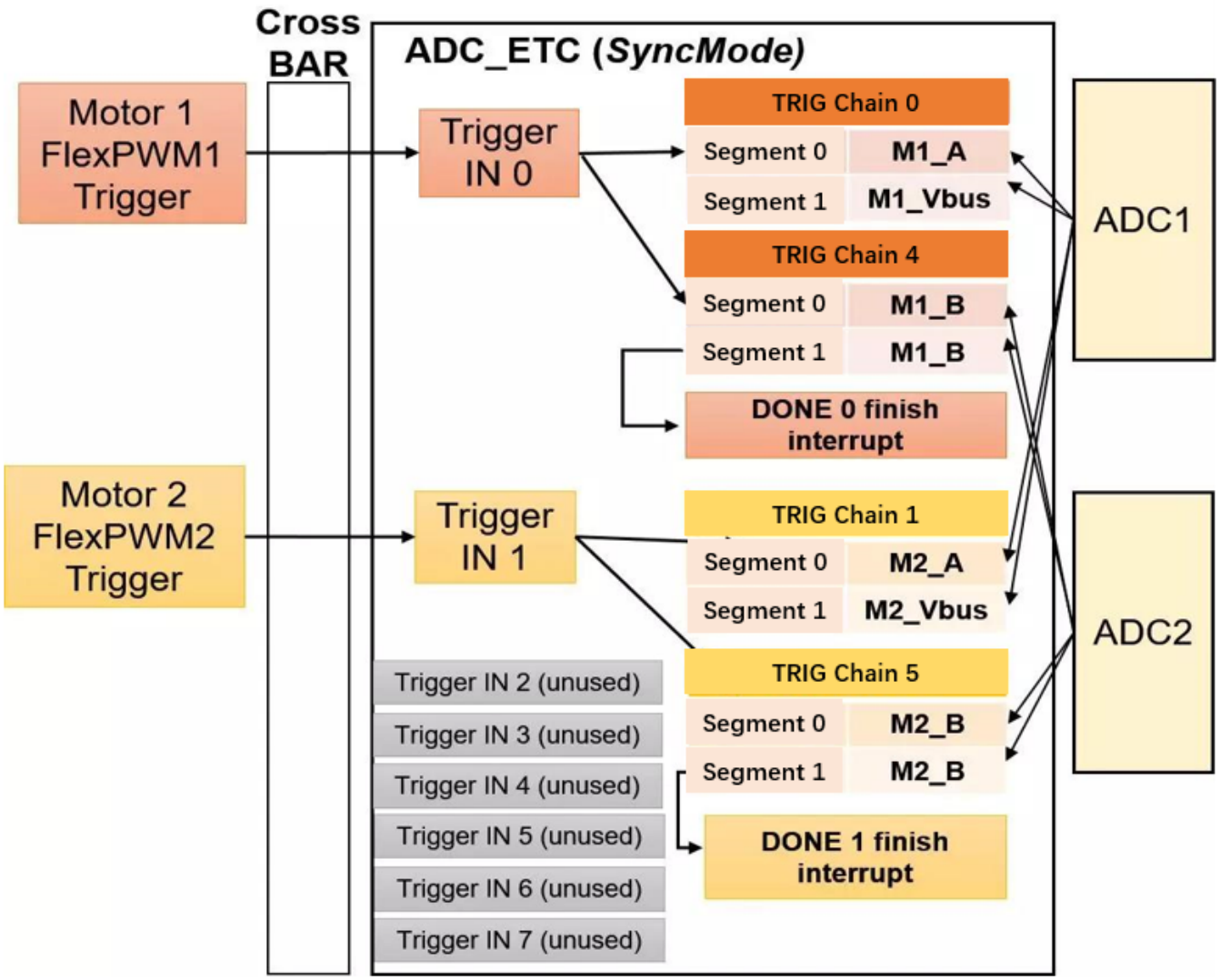


Figure 67-5. The configuration of ADC_ETC for dual motor control

67.6 Memory Map and register definition

This section includes the ADC_ETC module memory map and detailed descriptions of all registers.

67.6.1 ADC_ETC register descriptions

The descriptions below will list the memory map and give detailed explanations for all registers and the bit fields of the registers.

67.6.1.1 ADC_ETC memory map

ADC_ETC base address: 403B_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	ADC_ETC Global Control Register (CTRL)	32	RW	C000_0000h
4h	ETC DONE0 and DONE1 IRQ State Register (DONE0_1_IRQ)	32	W1C	0000_0000h
8h	ETC DONE_2 and DONE_ERR IRQ State Register (DONE2_3_ERR_IRQ)	32	RW	0000_0000h
Ch	ETC DMA control Register (DMA_CTRL)	32	RW	0000_0000h
10h	ETC_TRIG Control Register (TRIG0_CTRL)	32	RW	0000_0000h
14h	ETC_TRIG Counter Register (TRIG0_COUNTER)	32	RW	0000_0000h
18h	ETC_TRIG Chain 0/1 Register (TRIG0_CHAIN_1_0)	32	RW	0000_0000h
1Ch	ETC_TRIG Chain 2/3 Register (TRIG0_CHAIN_3_2)	32	RW	0000_0000h
20h	ETC_TRIG Chain 4/5 Register (TRIG0_CHAIN_5_4)	32	RW	0000_0000h
24h	ETC_TRIG Chain 6/7 Register (TRIG0_CHAIN_7_6)	32	RW	0000_0000h
28h	ETC_TRIG Result Data 1/0 Register (TRIG0_RESULT_1_0)	32	RO	0000_0000h
2Ch	ETC_TRIG Result Data 3/2 Register (TRIG0_RESULT_3_2)	32	RO	0000_0000h
30h	ETC_TRIG Result Data 5/4 Register (TRIG0_RESULT_5_4)	32	RO	0000_0000h
34h	ETC_TRIG Result Data 7/6 Register (TRIG0_RESULT_7_6)	32	RO	0000_0000h
38h	ETC_TRIG Control Register (TRIG1_CTRL)	32	RW	0000_0000h
3Ch	ETC_TRIG Counter Register (TRIG1_COUNTER)	32	RW	0000_0000h
40h	ETC_TRIG Chain 0/1 Register (TRIG1_CHAIN_1_0)	32	RW	0000_0000h
44h	ETC_TRIG Chain 2/3 Register (TRIG1_CHAIN_3_2)	32	RW	0000_0000h
48h	ETC_TRIG Chain 4/5 Register (TRIG1_CHAIN_5_4)	32	RW	0000_0000h
4Ch	ETC_TRIG Chain 6/7 Register (TRIG1_CHAIN_7_6)	32	RW	0000_0000h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
50h	ETC_TRIG Result Data 1/0 Register (TRIG1_RESULT_1_0)	32	RO	0000_0000h
54h	ETC_TRIG Result Data 3/2 Register (TRIG1_RESULT_3_2)	32	RO	0000_0000h
58h	ETC_TRIG Result Data 5/4 Register (TRIG1_RESULT_5_4)	32	RO	0000_0000h
5Ch	ETC_TRIG Result Data 7/6 Register (TRIG1_RESULT_7_6)	32	RO	0000_0000h
60h	ETC_TRIG Control Register (TRIG2_CTRL)	32	RW	0000_0000h
64h	ETC_TRIG Counter Register (TRIG2_COUNTER)	32	RW	0000_0000h
68h	ETC_TRIG Chain 0/1 Register (TRIG2_CHAIN_1_0)	32	RW	0000_0000h
6Ch	ETC_TRIG Chain 2/3 Register (TRIG2_CHAIN_3_2)	32	RW	0000_0000h
70h	ETC_TRIG Chain 4/5 Register (TRIG2_CHAIN_5_4)	32	RW	0000_0000h
74h	ETC_TRIG Chain 6/7 Register (TRIG2_CHAIN_7_6)	32	RW	0000_0000h
78h	ETC_TRIG Result Data 1/0 Register (TRIG2_RESULT_1_0)	32	RO	0000_0000h
7Ch	ETC_TRIG Result Data 3/2 Register (TRIG2_RESULT_3_2)	32	RO	0000_0000h
80h	ETC_TRIG Result Data 5/4 Register (TRIG2_RESULT_5_4)	32	RO	0000_0000h
84h	ETC_TRIG Result Data 7/6 Register (TRIG2_RESULT_7_6)	32	RO	0000_0000h
88h	ETC_TRIG Control Register (TRIG3_CTRL)	32	RW	0000_0000h
8Ch	ETC_TRIG Counter Register (TRIG3_COUNTER)	32	RW	0000_0000h
90h	ETC_TRIG Chain 0/1 Register (TRIG3_CHAIN_1_0)	32	RW	0000_0000h
94h	ETC_TRIG Chain 2/3 Register (TRIG3_CHAIN_3_2)	32	RW	0000_0000h
98h	ETC_TRIG Chain 4/5 Register (TRIG3_CHAIN_5_4)	32	RW	0000_0000h
9Ch	ETC_TRIG Chain 6/7 Register (TRIG3_CHAIN_7_6)	32	RW	0000_0000h
A0h	ETC_TRIG Result Data 1/0 Register (TRIG3_RESULT_1_0)	32	RO	0000_0000h
A4h	ETC_TRIG Result Data 3/2 Register (TRIG3_RESULT_3_2)	32	RO	0000_0000h
A8h	ETC_TRIG Result Data 5/4 Register (TRIG3_RESULT_5_4)	32	RO	0000_0000h
ACh	ETC_TRIG Result Data 7/6 Register (TRIG3_RESULT_7_6)	32	RO	0000_0000h
B0h	ETC_TRIG Control Register (TRIG4_CTRL)	32	RW	0000_0000h
B4h	ETC_TRIG Counter Register (TRIG4_COUNTER)	32	RW	0000_0000h
B8h	ETC_TRIG Chain 0/1 Register (TRIG4_CHAIN_1_0)	32	RW	0000_0000h
BCh	ETC_TRIG Chain 2/3 Register (TRIG4_CHAIN_3_2)	32	RW	0000_0000h
C0h	ETC_TRIG Chain 4/5 Register (TRIG4_CHAIN_5_4)	32	RW	0000_0000h
C4h	ETC_TRIG Chain 6/7 Register (TRIG4_CHAIN_7_6)	32	RW	0000_0000h
C8h	ETC_TRIG Result Data 1/0 Register (TRIG4_RESULT_1_0)	32	RO	0000_0000h
CCh	ETC_TRIG Result Data 3/2 Register (TRIG4_RESULT_3_2)	32	RO	0000_0000h
D0h	ETC_TRIG Result Data 5/4 Register (TRIG4_RESULT_5_4)	32	RO	0000_0000h
D4h	ETC_TRIG Result Data 7/6 Register (TRIG4_RESULT_7_6)	32	RO	0000_0000h
D8h	ETC_TRIG Control Register (TRIG5_CTRL)	32	RW	0000_0000h
DCh	ETC_TRIG Counter Register (TRIG5_COUNTER)	32	RW	0000_0000h
E0h	ETC_TRIG Chain 0/1 Register (TRIG5_CHAIN_1_0)	32	RW	0000_0000h
E4h	ETC_TRIG Chain 2/3 Register (TRIG5_CHAIN_3_2)	32	RW	0000_0000h
E8h	ETC_TRIG Chain 4/5 Register (TRIG5_CHAIN_5_4)	32	RW	0000_0000h
ECh	ETC_TRIG Chain 6/7 Register (TRIG5_CHAIN_7_6)	32	RW	0000_0000h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
F0h	ETC_TRIG Result Data 1/0 Register (TRIG5_RESULT_1_0)	32	RO	0000_0000h
F4h	ETC_TRIG Result Data 3/2 Register (TRIG5_RESULT_3_2)	32	RO	0000_0000h
F8h	ETC_TRIG Result Data 5/4 Register (TRIG5_RESULT_5_4)	32	RO	0000_0000h
FCh	ETC_TRIG Result Data 7/6 Register (TRIG5_RESULT_7_6)	32	RO	0000_0000h
100h	ETC_TRIG Control Register (TRIG6_CTRL)	32	RW	0000_0000h
104h	ETC_TRIG Counter Register (TRIG6_COUNTER)	32	RW	0000_0000h
108h	ETC_TRIG Chain 0/1 Register (TRIG6_CHAIN_1_0)	32	RW	0000_0000h
10Ch	ETC_TRIG Chain 2/3 Register (TRIG6_CHAIN_3_2)	32	RW	0000_0000h
110h	ETC_TRIG Chain 4/5 Register (TRIG6_CHAIN_5_4)	32	RW	0000_0000h
114h	ETC_TRIG Chain 6/7 Register (TRIG6_CHAIN_7_6)	32	RW	0000_0000h
118h	ETC_TRIG Result Data 1/0 Register (TRIG6_RESULT_1_0)	32	RO	0000_0000h
11Ch	ETC_TRIG Result Data 3/2 Register (TRIG6_RESULT_3_2)	32	RO	0000_0000h
120h	ETC_TRIG Result Data 5/4 Register (TRIG6_RESULT_5_4)	32	RO	0000_0000h
124h	ETC_TRIG Result Data 7/6 Register (TRIG6_RESULT_7_6)	32	RO	0000_0000h
128h	ETC_TRIG Control Register (TRIG7_CTRL)	32	RW	0000_0000h
12Ch	ETC_TRIG Counter Register (TRIG7_COUNTER)	32	RW	0000_0000h
130h	ETC_TRIG Chain 0/1 Register (TRIG7_CHAIN_1_0)	32	RW	0000_0000h
134h	ETC_TRIG Chain 2/3 Register (TRIG7_CHAIN_3_2)	32	RW	0000_0000h
138h	ETC_TRIG Chain 4/5 Register (TRIG7_CHAIN_5_4)	32	RW	0000_0000h
13Ch	ETC_TRIG Chain 6/7 Register (TRIG7_CHAIN_7_6)	32	RW	0000_0000h
140h	ETC_TRIG Result Data 1/0 Register (TRIG7_RESULT_1_0)	32	RO	0000_0000h
144h	ETC_TRIG Result Data 3/2 Register (TRIG7_RESULT_3_2)	32	RO	0000_0000h
148h	ETC_TRIG Result Data 5/4 Register (TRIG7_RESULT_5_4)	32	RO	0000_0000h
14Ch	ETC_TRIG Result Data 7/6 Register (TRIG7_RESULT_7_6)	32	RO	0000_0000h

67.6.1.2 ADC_ETC Global Control Register (CTRL)

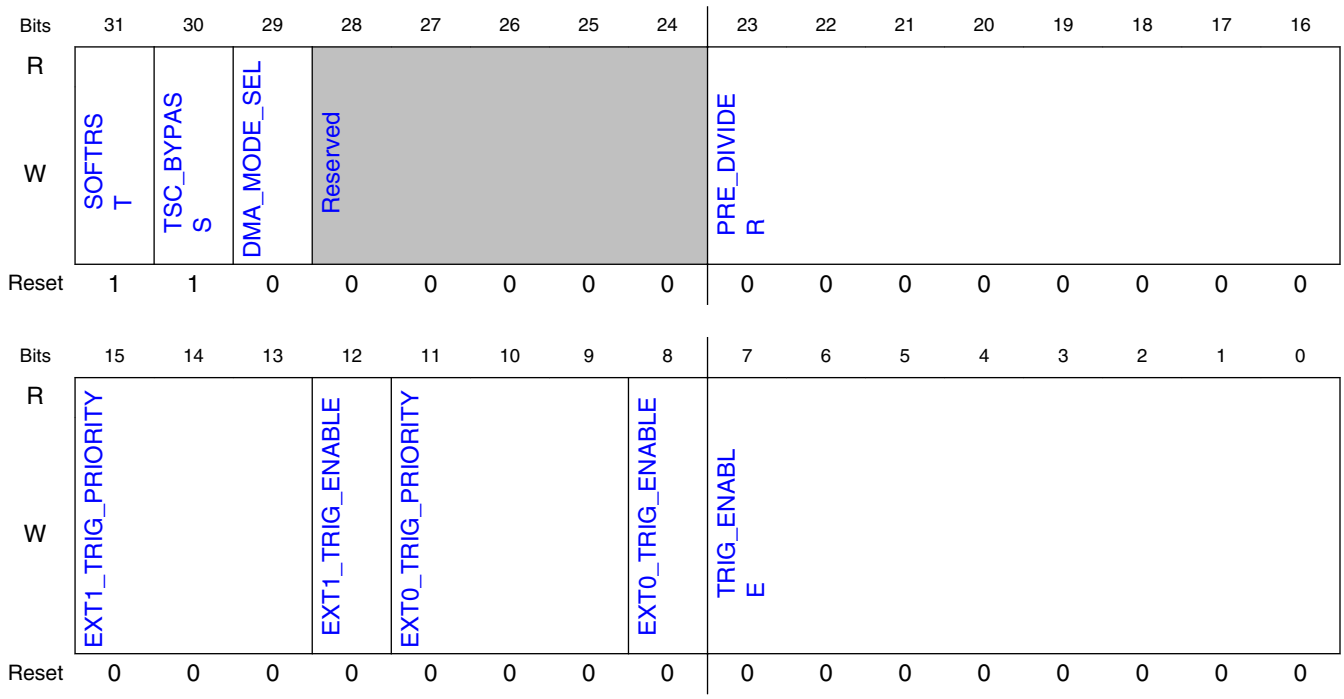
67.6.1.2.1 Offset

Register	Offset
CTRL	0h

67.6.1.2.2 Function

This register controls various global functions of the ADC_ETC, such as enabling the external XBAR triggers, setting the pre-division factor for trigger delay and interval, and selecting the trigger type of the DMA_REQ. In addition, this register contains SOFTRST to reset all registers inside ADC_ETC.

67.6.1.2.3 Diagram



67.6.1.2.4 Fields

Field	Function
31 SOFTRST	Software synchronous reset, active high. 0b - ADC_ETC works normally. 1b - All registers inside ADC_ETC will be reset to the default value.
30 TSC_BYPASS	TSC Bypass **NOTE:** To use ADC2, this bit should be cleared. 0b - TSC not bypassed. 1b - TSC is bypassed to ADC2, that means TSC will control ADC2 directly.
29 DMA_MODE_SEL	Select the trigger type of the DMA_REQ. 0b - Trig DMA_REQ with latched signal, REQ will be cleared when ACK and source request cleared. 1b - Trig DMA_REQ with pulsed signal, REQ will be cleared by ACK only.
28-24	Reserved

Table continues on the next page...

Field	Function
—	
23-16 PRE_DIVIDER	Pre-divider for trig delay and interval. The meaning of this field will be explained in the description of TRIGa_COUNTER.
15-13 EXT1_TRIG_PRIORITY	External TSC1 trigger priority, 7 is highest priority, while 0 is lowest.
12 EXT1_TRIG_ENABLE	TSC1 TRIG enable register. 0b - disable external TSC1 trigger. 1b - enable external TSC1 trigger.
11-9 EXT0_TRIG_PRIORITY	External TSC0 trigger priority, 7 is highest priority, while 0 is lowest.
8 EXT0_TRIG_ENABLE	TSC0 TRIG enable register. 0b - disable external TSC0 trigger. 1b - enable external TSC0 trigger.
7-0 TRIG_ENABLE	TRIG enable register. 00000000b - disable all 8 external XBAR triggers. 00000001b - enable external XBAR trigger0. 00000010b - enable external XBAR trigger1. 00000011b - enable external XBAR trigger0 and trigger1. 11111111b - enable all 8 external XBAR triggers.

67.6.1.3 ETC DONE0 and DONE1 IRQ State Register (DONE0_1_IRQ)

67.6.1.3.1 Offset

Register	Offset
DONE0_1_IRQ	4h

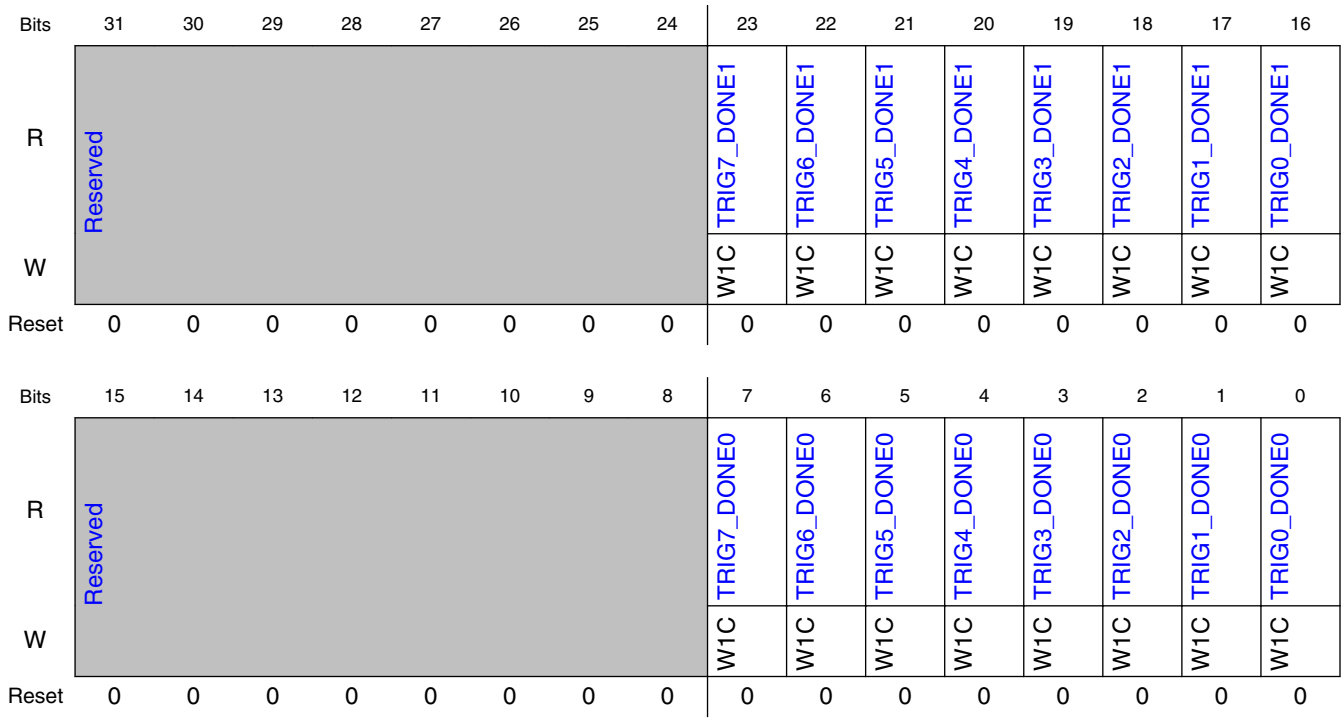
67.6.1.3.2 Function

This register has the state for ETC DONE0 and DONE1 interrupts.

NOTE

The IRQs can be cleared by writing 1 to the corresponding bits.

67.6.1.3.3 Diagram



67.6.1.3.4 Fields

Field	Function
31-24 —	Reserved
23 TRIG7_DONE1	TRIG7 done1 interrupt detection. 0b - No TRIG7_DONE1 interrupt detected 1b - TRIG7_DONE1 interrupt detected
22 TRIG6_DONE1	TRIG6 done1 interrupt detection. 0b - No TRIG6_DONE1 interrupt detected 1b - TRIG6_DONE1 interrupt detected
21 TRIG5_DONE1	TRIG5 done1 interrupt detection. 0b - No TRIG5_DONE1 interrupt detected 1b - TRIG5_DONE1 interrupt detected
20 TRIG4_DONE1	TRIG4 done1 interrupt detection. 0b - No TRIG4_DONE1 interrupt detected 1b - TRIG4_DONE1 interrupt detected
19 TRIG3_DONE1	TRIG3 done1 interrupt detection. 0b - No TRIG3_DONE1 interrupt detected 1b - TRIG3_DONE1 interrupt detected
18	TRIG2 done1 interrupt detection.

Table continues on the next page...

Field	Function
TRIG2_DONE1	0b - No TRIG2_DONE1 interrupt detected 1b - TRIG2_DONE1 interrupt detected
17 TRIG1_DONE1	TRIG1 done1 interrupt detection. 0b - No TRIG1_DONE1 interrupt detected 1b - TRIG1_DONE1 interrupt detected
16 TRIG0_DONE1	TRIG0 done1 interrupt detection. 0b - No TRIG0_DONE1 interrupt detected 1b - TRIG0_DONE1 interrupt detected
15-8 —	Reserved
7 TRIG7_DONE0	TRIG7 done0 interrupt detection. 0b - No TRIG7_DONE0 interrupt detected 1b - TRIG7_DONE0 interrupt detected
6 TRIG6_DONE0	TRIG6 done0 interrupt detection. 0b - No TRIG6_DONE0 interrupt detected 1b - TRIG6_DONE0 interrupt detected
5 TRIG5_DONE0	TRIG5 done0 interrupt detection. 0b - No TRIG5_DONE0 interrupt detected 1b - TRIG5_DONE0 interrupt detected
4 TRIG4_DONE0	TRIG4 done0 interrupt detection. 0b - No TRIG4_DONE0 interrupt detected 1b - TRIG4_DONE0 interrupt detected
3 TRIG3_DONE0	TRIG3 done0 interrupt detection. 0b - No TRIG3_DONE0 interrupt detected 1b - TRIG3_DONE0 interrupt detected
2 TRIG2_DONE0	TRIG2 done0 interrupt detection. 0b - No TRIG2_DONE0 interrupt detected 1b - TRIG2_DONE0 interrupt detected
1 TRIG1_DONE0	TRIG1 done0 interrupt detection. 0b - No TRIG1_DONE0 interrupt detected 1b - TRIG1_DONE0 interrupt detected
0 TRIG0_DONE0	TRIG0 done0 interrupt detection. 0b - No TRIG0_DONE0 interrupt detected 1b - TRIG0_DONE0 interrupt detected

67.6.1.4 ETC DONE_2 and DONE_ERR IRQ State Register (DONE2_3_ERR_IRQ)

67.6.1.4.1 Offset

Register	Offset
DONE2_3_ERR_IRQ	8h

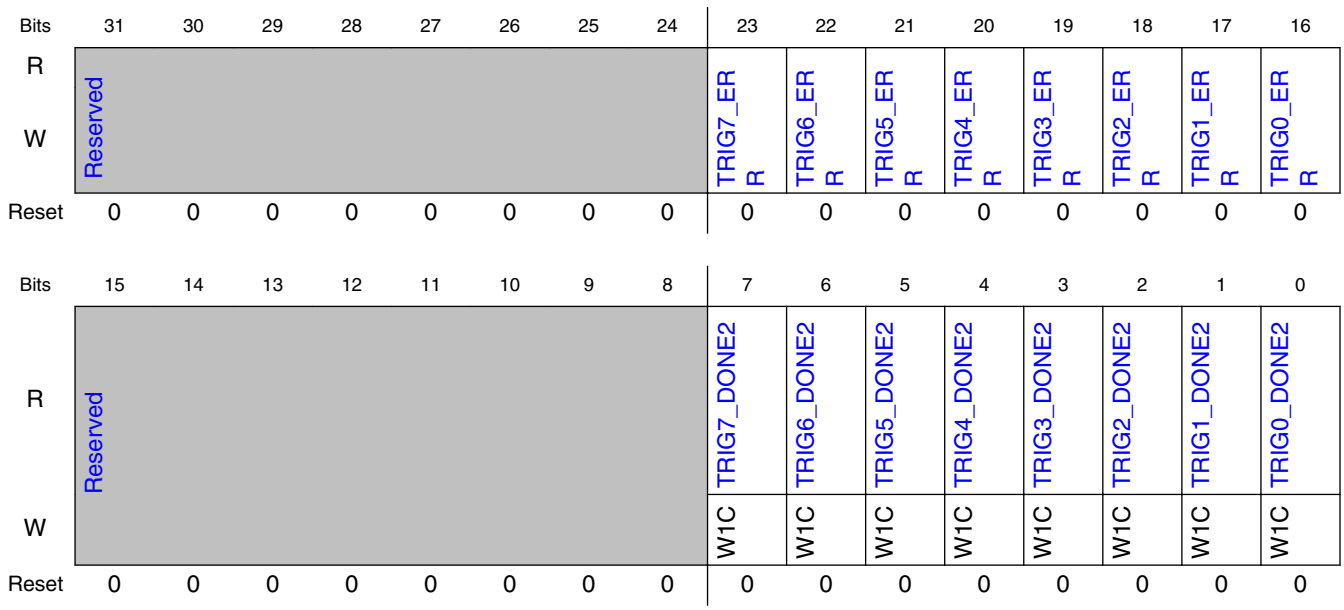
67.6.1.4.2 Function

This register has the state for ETC DONE2 and DONE_ERR interrupts.

NOTE

The IRQs can be cleared by writing 1 to the corresponding bits.

67.6.1.4.3 Diagram



67.6.1.4.4 Fields

Field	Function
31-24 —	Reserved
23 TRIG7_ERR	TRIG7 error interrupt detection. 0b - No TRIG7_ERR interrupt detected 1b - TRIG7_ERR interrupt detected
22 TRIG6_ERR	TRIG6 error interrupt detection. 0b - No TRIG6_ERR interrupt detected

Table continues on the next page...

Field	Function
	1b - TRIG6_ERR interrupt detected
21 TRIG5_ERR	TRIG5 error interrupt detection. 0b - No TRIG5_ERR interrupt detected 1b - TRIG5_ERR interrupt detected
20 TRIG4_ERR	TRIG4 error interrupt detection. 0b - No TRIG4_ERR interrupt detected 1b - TRIG4_ERR interrupt detected
19 TRIG3_ERR	TRIG3 error interrupt detection. 0b - No TRIG3_ERR interrupt detected 1b - TRIG3_ERR interrupt detected
18 TRIG2_ERR	TRIG2 error interrupt detection. 0b - No TRIG2_ERR interrupt detected 1b - TRIG2_ERR interrupt detected
17 TRIG1_ERR	TRIG1 error interrupt detection. 0b - No TRIG1_ERR interrupt detected 1b - TRIG1_ERR interrupt detected
16 TRIG0_ERR	TRIG0 error interrupt detection. 0b - No TRIG0_ERR interrupt detected 1b - TRIG0_ERR interrupt detected
15-8 —	Reserved
7 TRIG7_DONE2	TRIG7 done2 interrupt detection. 0b - No TRIG7_DONE2 interrupt detected 1b - TRIG7_DONE2 interrupt detected
6 TRIG6_DONE2	TRIG6 done2 interrupt detection. 0b - No TRIG6_DONE2 interrupt detected 1b - TRIG6_DONE2 interrupt detected
5 TRIG5_DONE2	TRIG5 done2 interrupt detection. 0b - No TRIG5_DONE2 interrupt detected 1b - TRIG5_DONE2 interrupt detected
4 TRIG4_DONE2	TRIG4 done2 interrupt detection. 0b - No TRIG4_DONE2 interrupt detected 1b - TRIG4_DONE2 interrupt detected
3 TRIG3_DONE2	TRIG3 done2 interrupt detection. 0b - No TRIG3_DONE2 interrupt detected 1b - TRIG3_DONE2 interrupt detected
2 TRIG2_DONE2	TRIG2 done2 interrupt detection. 0b - No TRIG2_DONE2 interrupt detected 1b - TRIG2_DONE2 interrupt detected
1 TRIG1_DONE2	TRIG1 done2 interrupt detection. 0b - No TRIG1_DONE2 interrupt detected 1b - TRIG1_DONE2 interrupt detected
0 TRIG0_DONE2	TRIG0 done2 interrupt detection. 0b - No TRIG0_DONE2 interrupt detected

Field	Function
	1b - TRIG0_DONE2 interrupt detected

67.6.1.5 ETC DMA control Register (DMA_CTRL)

67.6.1.5.1 Offset

Register	Offset
DMA_CTRL	Ch

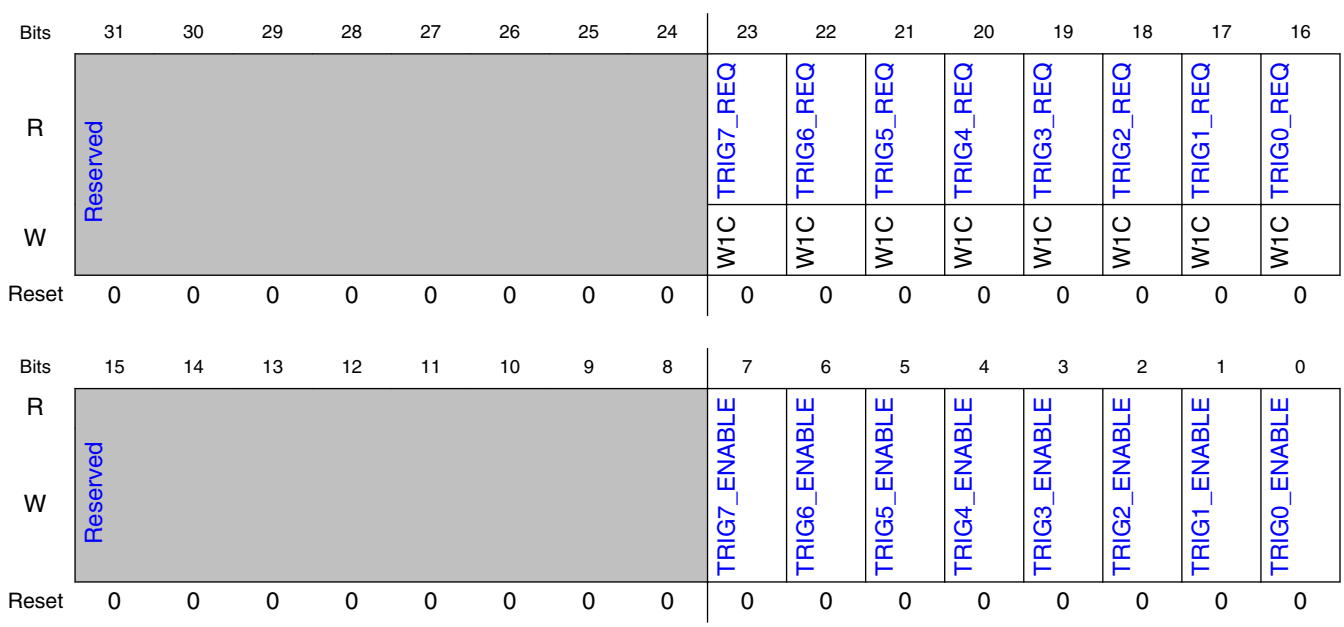
67.6.1.5.2 Function

This register has the control and status bit for the DMA of ADC_ETC. Each trigger has separate enabling and status bit in this register. The control bit enables the DMA request for each trigger, and the status bit indicates whether the corresponding trigger request was detected.

NOTE

TRIGa_REQ can be cleard by writing 1 to the corresponding bit.

67.6.1.5.3 Diagram



67.6.1.5.4 Fields

Field	Function
31-24 —	Reserved
23 TRIG7_REQ	Flag bit for DMA request. TRIG7_REQ being active indicates trigger7 done and a DMA transmission is needed. If TRIG7_ENABLE is also set, a DMA request will be sent. 0b - TRIG7_REQ not detected. 1b - TRIG7_REQ detected.
22 TRIG6_REQ	Flag bit for DMA request. TRIG6_REQ being active indicates trigger6 done and a DMA transmission is needed. If TRIG6_ENABLE is also set, a DMA request will be sent. 0b - TRIG6_REQ not detected. 1b - TRIG6_REQ detected.
21 TRIG5_REQ	Flag bit for DMA request. TRIG5_REQ being active indicates trigger5 done and a DMA transmission is needed. If TRIG5_ENABLE is also set, a DMA request will be sent. 0b - TRIG5_REQ not detected. 1b - TRIG5_REQ detected.
20 TRIG4_REQ	Flag bit for DMA request. TRIG4_REQ being active indicates trigger4 done and a DMA transmission is needed. If TRIG4_ENABLE is also set, a DMA request will be sent. 0b - TRIG4_REQ not detected. 1b - TRIG4_REQ detected.
19 TRIG3_REQ	Flag bit for DMA request. TRIG3_REQ being active indicates trigger3 done and a DMA transmission is needed. If TRIG3_ENABLE is also set, a DMA request will be sent. 0b - TRIG3_REQ not detected. 1b - TRIG3_REQ detected.
18 TRIG2_REQ	Flag bit for DMA request. TRIG2_REQ being active indicates trigger2 done and a DMA transmission is needed. If TRIG2_ENABLE is also set, a DMA request will be sent. 0b - TRIG2_REQ not detected. 1b - TRIG2_REQ detected.
17 TRIG1_REQ	Flag bit for DMA request. TRIG1_REQ being active indicates trigger1 done and a DMA transmission is needed. If TRIG1_ENABLE is also set, a DMA request will be sent. 0b - TRIG1_REQ not detected. 1b - TRIG1_REQ detected.
16 TRIG0_REQ	Flag bit for DMA request. TRIG0_REQ being active indicates trigger0 done and a DMA transmission is needed. If TRIG0_ENABLE is also set, a DMA request will be sent. 0b - TRIG0_REQ not detected. 1b - TRIG0_REQ detected.
15-8 —	Reserved
7 TRIG7_ENABLE	Enable DMA request when TRIG7 done. 0b - TRIG7 DMA request disabled. 1b - TRIG7 DMA request enabled.
6 TRIG6_ENABLE	Enable DMA request when TRIG6 done. 0b - TRIG6 DMA request disabled. 1b - TRIG6 DMA request enabled.

Table continues on the next page...

Field	Function
5 TRIG5_ENABLE	Enable DMA request when TRIG5 done. 0b - TRIG5 DMA request disabled. 1b - TRIG5 DMA request enabled.
4 TRIG4_ENABLE	Enable DMA request when TRIG4 done. 0b - TRIG4 DMA request disabled. 1b - TRIG4 DMA request enabled.
3 TRIG3_ENABLE	Enable DMA request when TRIG3 done. 0b - TRIG3 DMA request disabled. 1b - TRIG3 DMA request enabled.
2 TRIG2_ENABLE	Enable DMA request when TRIG2 done. 0b - TRIG2 DMA request disabled. 1b - TRIG2 DMA request enabled.
1 TRIG1_ENABLE	Enable DMA request when TRIG1 done. 0b - TRIG1 DMA request disabled. 1b - TRIG1 DMA request enabled.
0 TRIG0_ENABLE	Enable DMA request when TRIG0 done. 0b - TRIG0 DMA request disabled. 1b - TRIG0 DMA request enabled.

67.6.1.6 ETC_TRIG Control Register (TRIG0_CTRL - TRIG7_CTRL)

67.6.1.6.1 Offset

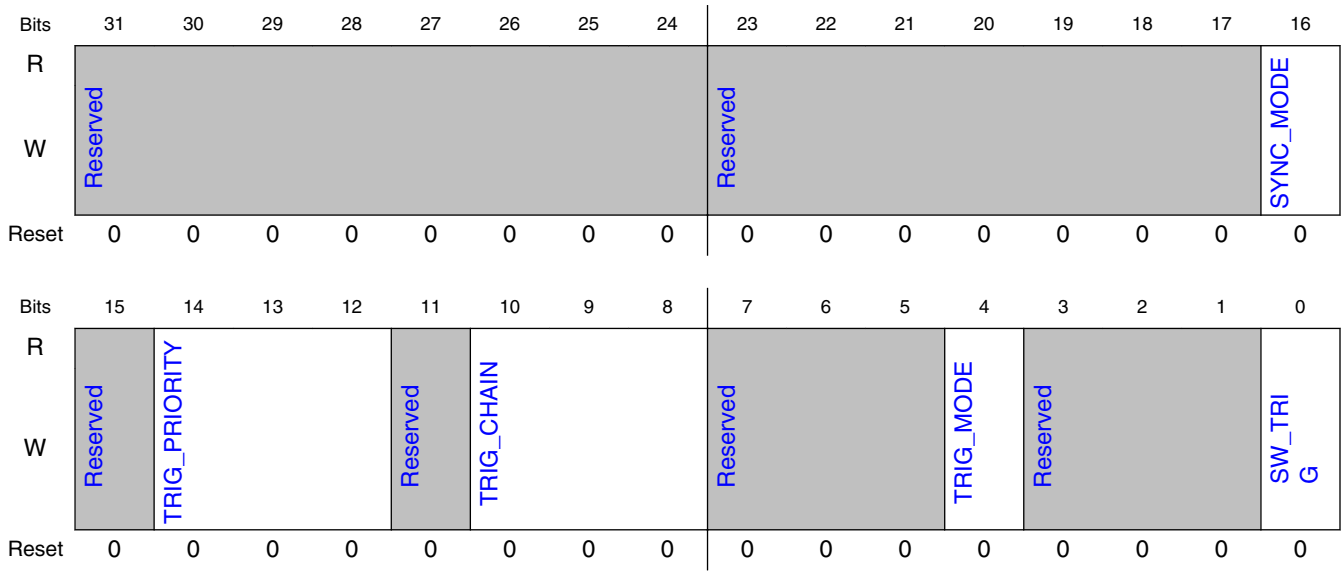
For a = 0 to 7:

Register	Offset
TRIGa_CTRL	10h + (a × 28h)

67.6.1.6.2 Function

This register contains the control bits for trigger, such as selecting synchronization mode, setting trigger priority and setting the length of trigger chain. It also contains a bit for selecting trigger mode between hardware trigger and software trigger, and a self-clearing bit for launching a software trigger.

67.6.1.6.3 Diagram



67.6.1.6.4 Fields

Field	Function
31-24 —	Reserved
23-17 —	Reserved
16 SYNC_MODE	Trigger synchronization mode selection. **NOTE:** Setting SYNC_MODE = 1 for TRIGa or TRIG(a+4) represents the same meaning. 0b - Synchronization mode disabled, TRIGa and TRIG(a+4) are triggered independently. 1b - Synchronization mode enabled, TRIGa and TRIG(a+4) are triggered by TRIGa source synchronously.
15 —	Reserved
14-12 TRIG_PRIORITY	External trigger priority, 7 is highest priority, while 0 is lowest. **NOTE:** When more than one triggers use the same level setting of priority, then round-robin mechanism implemented and the trigger with larger number owns the higher priority.
11 —	Reserved
10-8 TRIG_CHAIN	The number of segments inside the trigger chain of TRIGa. 000b - Trigger chain length is 1 001b - Trigger chain length is 2 010b - Trigger chain length is 3 011b - Trigger chain length is 4 100b - Trigger chain length is 5 101b - Trigger chain length is 6

Table continues on the next page...

Field	Function
	110b - Trigger chain length is 7 111b - Trigger chain length is 8
7-5 —	Reserved
4 TRIG_MODE	Trigger mode selection. 0b - Hardware trigger. The softerware trigger will be ignored. 1b - Software trigger. The hardware trigger will be ignored.
3-1 —	Reserved
0 SW_TRIG	Software trigger. **NOTE:** This field is self-clearing. 0b - No software trigger event generated. 1b - Software trigger event generated.

67.6.1.7 ETC_TRIG Counter Register (TRIG0_COUNTER - TRIG7_COUNTER)

67.6.1.7.1 Offset

For a = 0 to 7:

Register	Offset
TRIGa_COUNTER	14h + (a × 28h)

67.6.1.7.2 Function

This register controls the initial delay ahead the first ADC trigger and interval delay between ADC triggers launched by the same ADC_ETC trigger.

67.6.1.7.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	SAMPLE_INTERVAL															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	INIT_DELAY															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

67.6.1.7.4 Fields

Field	Function
31-16 SAMPLE_INTERVAL	TRIGGER sampling interval counter. When B2B is unset: Interval_delay = (SAMPLE_INTERVAL+1)*(PRE_DIVIDER+1)*ipg_clk
15-0 INIT_DELAY	TRIGGER initial delay counter. Initial_delay = (INIT_DELAY+1)*(PRE_DIVIDER+1)*ipg_clk

67.6.1.8 ETC_TRIG Chain 0/1 Register (TRIG0_CHAIN_1_0 - TRIG7_CHAIN_1_0)

67.6.1.8.1 Offset

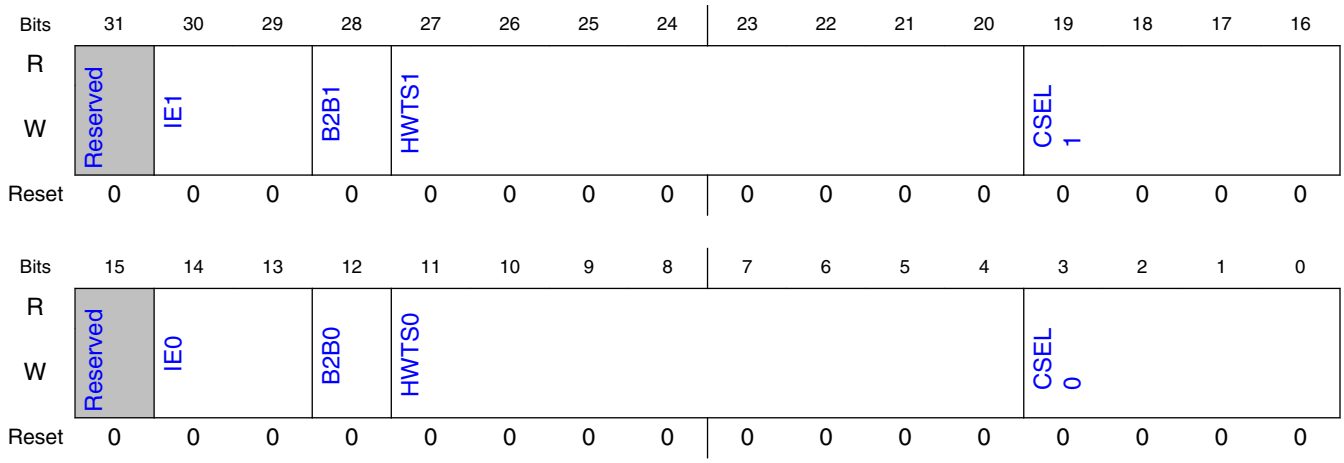
For a = 0 to 7:

Register	Offset
TRIGa_CHAIN_1_0	18h + (a × 28h)

67.6.1.8.2 Function

This register controls ETC_TRIG segment 0/1 configuration, such as enabling and selecting interrupt, setting back-to-back mode,selecting ADC trigger and trigger command.

67.6.1.8.3 Diagram



67.6.1.8.4 Fields

Field	Function
31 —	Reserved
30-29 IE1	Segment 1 done interrupt selection 00b - No interrupt when finished 01b - Generate interrupt on Done0 when Segment 1 finish. 10b - Generate interrupt on Done1 when Segment 1 finish. 11b - Generate interrupt on Done2 when Segment 1 finish.
28 B2B1	Segment 1 B2B 0b - Disable B2B. Wait until delay value defined by TRIG1_COUNTER[SAMPLE_INTERVAL] is reached 1b - Enable B2B. When Segment 0 finished (ADC COCO) then automatically trigger next ADC conversion, no need to wait until interval delay reached.
27-20 HWTS1	Segment 1 HWTS ADC hardware trigger selection. This field will control the ext_hwts of ADC (see the ADC_ETC block diagram). **NOTE:** Please keep only one bit set at the same time. 00000000b - no trigger selected 00000001b - ADC TRIG0 selected 00000010b - ADC TRIG1 selected 00000100b - ADC TRIG2 selected 00001000b - ADC TRIG3 selected 00010000b - ADC TRIG4 selected 00100000b - ADC TRIG5 selected 01000000b - ADC TRIG6 selected 10000000b - ADC TRIG7 selected
19-16 CSEL1	ADC channel selection 0000b - ADC Channel 0 selected 0001b - ADC Channel 1 selected. 0010b - ADC Channel 2 selected.

Table continues on the next page...

Field	Function
	0011b - ADC Channel 3 selected. 0100b - ADC Channel 4 selected. 0101b - ADC Channel 5 selected. 0110b - ADC Channel 6 selected. 0111b - ADC Channel 7 selected. 1000b - ADC Channel 8 selected. 1001b - ADC Channel 9 selected. 1010b - ADC Channel 10 selected. 1011b - ADC Channel 11 selected. 1100b - ADC Channel 12 selected. 1101b - ADC Channel 13 selected. 1110b - ADC Channel 14 selected. 1111b - ADC Channel 15 selected.
15 —	Reserved
14-13 IE0	Segment 0 done interrupt selection 00b - No interrupt when finished 01b - Generate interrupt on Done0 when segment 0 finish. 10b - Generate interrupt on Done1 when segment 0 finish. 11b - Generate interrupt on Done2 when segment 0 finish.
12 B2B0	Segment 0 B2B 0b - Disable B2B. Wait until delay value defined by TRIG0_COUNTER[SAMPLE_INTERVAL] is reached 1b - Enable B2B. When Segment 0 finished (ADC COCO) then automatically trigger next ADC conversion, no need to wait until interval delay reached.
11-4 HWTS0	Segment 0 HWTS ADC hardware trigger selection. This field will control the ext_hwts of ADC (see the ADC_ETC block diagram). **NOTE:** Please keep only one bit set at the same time. 00000000b - no trigger selected 00000001b - ADC TRIG0 selected 00000010b - ADC TRIG1 selected 00000100b - ADC TRIG2 selected 00001000b - ADC TRIG3 selected 00010000b - ADC TRIG4 selected 00100000b - ADC TRIG5 selected 01000000b - ADC TRIG6 selected 10000000b - ADC TRIG7 selected
3-0 CSEL0	ADC channel selection 0000b - ADC Channel 0 selected 0001b - ADC Channel 1 selected. 0010b - ADC Channel 2 selected. 0011b - ADC Channel 3 selected. 0100b - ADC Channel 4 selected. 0101b - ADC Channel 5 selected. 0110b - ADC Channel 6 selected. 0111b - ADC Channel 7 selected. 1000b - ADC Channel 8 selected. 1001b - ADC Channel 9 selected. 1010b - ADC Channel 10 selected. 1011b - ADC Channel 11 selected. 1100b - ADC Channel 12 selected.

Field	Function
	1101b - ADC Channel 13 selected. 1110b - ADC Channel 14 selected. 1111b - ADC Channel 15 selected.

67.6.1.9 ETC_TRIG Chain 2/3 Register (TRIG0_CHAIN_3_2 - TRIG7_CHAIN_3_2)

67.6.1.9.1 Offset

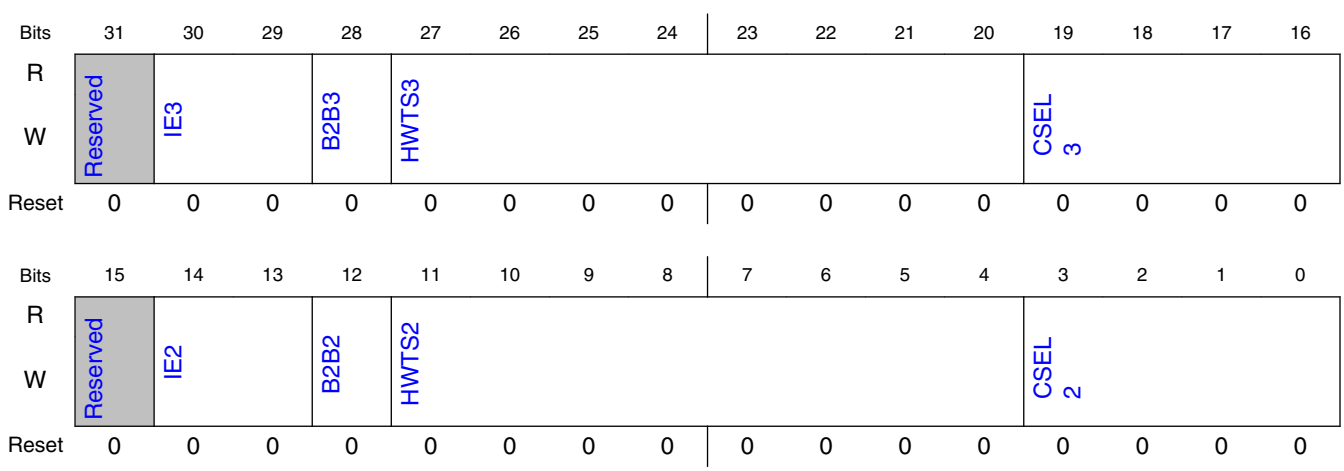
For a = 0 to 7:

Register	Offset
TRIGa_CHAIN_3_2	1Ch + (a × 28h)

67.6.1.9.2 Function

This register controls ETC_TRIG segment 2/3 configuration, such as enabling and selecting interrupt, setting back-to-back mode,selecting ADC trigger and trigger command.

67.6.1.9.3 Diagram



67.6.1.9.4 Fields

Field	Function
31 —	Reserved
30-29 IE3	Segment 3 done interrupt selection 00b - No interrupt when finished 01b - Generate interrupt on Done0 when segment 3 finish. 10b - Generate interrupt on Done1 when segment 3 finish. 11b - Generate interrupt on Done2 when segment 3 finish.
28 B2B3	Segment 3 B2B 0b - Disable B2B. Wait until delay value defined by TRIG3_COUNTER[SAMPLE_INTERVAL] is reached 1b - Enable B2B. When Segment 0 finished (ADC COCO) then automatically trigger next ADC conversion, no need to wait until interval delay reached.
27-20 HWTS3	Segment 3 HWTS ADC hardware trigger selection. This field will control the ext_hwts of ADC (see the ADC_ETC block diagram). **NOTE:** Please keep only one bit set at the same time. 00000000b - no trigger selected 00000001b - ADC TRIG0 selected 00000010b - ADC TRIG1 selected 00000100b - ADC TRIG2 selected 00001000b - ADC TRIG3 selected 00010000b - ADC TRIG4 selected 00100000b - ADC TRIG5 selected 01000000b - ADC TRIG6 selected 10000000b - ADC TRIG7 selected
19-16 CSEL3	ADC channel selection 0000b - ADC Channel 0 selected 0001b - ADC Channel 1 selected. 0010b - ADC Channel 2 selected. 0011b - ADC Channel 3 selected. 0100b - ADC Channel 4 selected. 0101b - ADC Channel 5 selected. 0110b - ADC Channel 6 selected. 0111b - ADC Channel 7 selected. 1000b - ADC Channel 8 selected. 1001b - ADC Channel 9 selected. 1010b - ADC Channel 10 selected. 1011b - ADC Channel 11 selected. 1100b - ADC Channel 12 selected. 1101b - ADC Channel 13 selected. 1110b - ADC Channel 14 selected. 1111b - ADC Channel 15 selected.
15 —	Reserved
14-13 IE2	Segment 2 done interrupt selection 00b - No interrupt when finished 01b - Generate interrupt on Done0 when segment 2 finish. 10b - Generate interrupt on Done1 when segment 2 finish.

Table continues on the next page...

Field	Function
	11b - Generate interrupt on Done2 when segment 2 finish.
12 B2B2	Segment 2 B2B 0b - Disable B2B. Wait until delay value defined by TRIG2_COUNTER[SAMPLE_INTERVAL] is reached 1b - Enable B2B. When Segment 0 finished (ADC COCO) then automatically trigger next ADC conversion, no need to wait until interval delay reached.
11-4 HWTS2	Segment 2 HWTS ADC hardware trigger selection. This field will control the ext_hwts of ADC (see the ADC_ETC block diagram). **NOTE:** Please keep only one bit set at the same time. 00000000b - no trigger selected 00000001b - ADC TRIG0 selected 00000010b - ADC TRIG1 selected 00000100b - ADC TRIG2 selected 00001000b - ADC TRIG3 selected 00010000b - ADC TRIG4 selected 00100000b - ADC TRIG5 selected 01000000b - ADC TRIG6 selected 10000000b - ADC TRIG7 selected
3-0 CSEL2	ADC channel selection 0000b - ADC Channel 0 selected 0001b - ADC Channel 1 selected. 0010b - ADC Channel 2 selected. 0011b - ADC Channel 3 selected. 0100b - ADC Channel 4 selected. 0101b - ADC Channel 5 selected. 0110b - ADC Channel 6 selected. 0111b - ADC Channel 7 selected. 1000b - ADC Channel 8 selected. 1001b - ADC Channel 9 selected. 1010b - ADC Channel 10 selected. 1011b - ADC Channel 11 selected. 1100b - ADC Channel 12 selected. 1101b - ADC Channel 13 selected. 1110b - ADC Channel 14 selected. 1111b - ADC Channel 15 selected.

67.6.1.10 ETC_TRIG Chain 4/5 Register (TRIG0_CHAIN_5_4 - TRIG7_CHAIN_5_4)

67.6.1.10.1 Offset

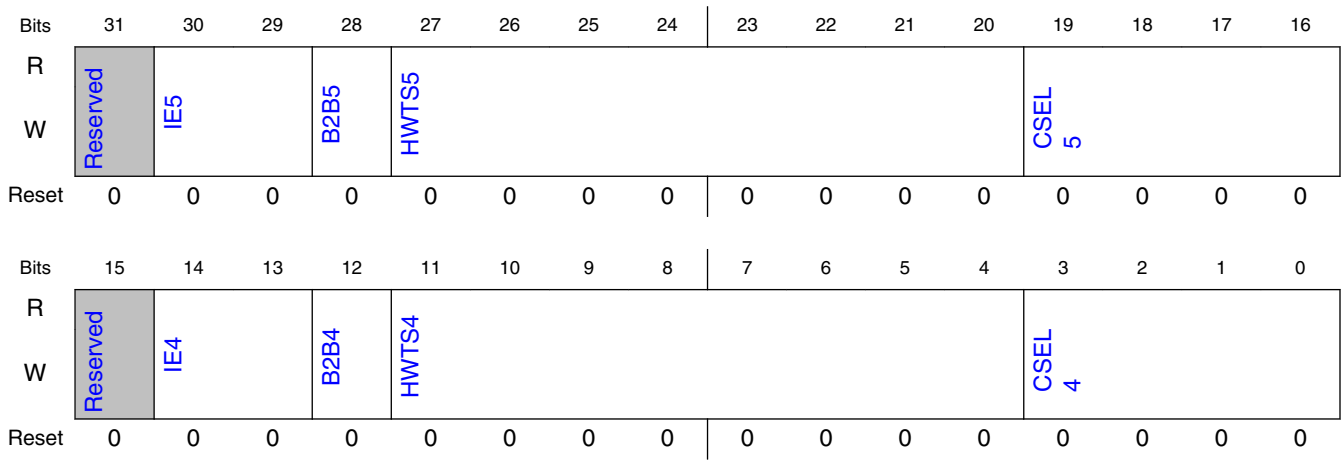
For a = 0 to 7:

Register	Offset
TRIGa_CHAIN_5_4	20h + (a × 28h)

67.6.1.10.2 Function

This register controls ETC_TRIG segment 4/5 configuration, such as enabling and selecting interrupt, setting back-to-back mode,selecting ADC trigger and trigger command.

67.6.1.10.3 Diagram



67.6.1.10.4 Fields

Field	Function
31 —	Reserved
30-29 IE5	Segment 5 done interrupt selection 00b - No interrupt when finished 01b - Generate interrupt on Done0 when segment 5 finish. 10b - Generate interrupt on Done1 when segment 5 finish. 11b - Generate interrupt on Done2 when segment 5 finish.
28 B2B5	Segment 5 B2B 0b - Disable B2B. Wait until delay value defined by TRIG5_COUNTER[SAMPLE_INTERVAL] is reached 1b - Enable B2B. When Segment 0 finished (ADC COCO) then automatically trigger next ADC conversion, no need to wait until interval delay reached.
27-20 HWTS5	Segment 5 HWTS ADC hardware trigger selection. This field will control the ext_hwts of ADC (see the ADC_ETC block diagram). **NOTE:** Please keep only one bit set at the same time. 00000000b - no trigger selected 00000001b - ADC TRIG0 selected 00000010b - ADC TRIG1 selected

Table continues on the next page...

Field	Function
	00000100b - ADC TRIG2 selected 00001000b - ADC TRIG3 selected 00010000b - ADC TRIG4 selected 00100000b - ADC TRIG5 selected 01000000b - ADC TRIG6 selected 10000000b - ADC TRIG7 selected
19-16 CSEL5	ADC channel selection 0000b - ADC Channel 0 selected 0001b - ADC Channel 1 selected. 0010b - ADC Channel 2 selected. 0011b - ADC Channel 3 selected. 0100b - ADC Channel 4 selected. 0101b - ADC Channel 5 selected. 0110b - ADC Channel 6 selected. 0111b - ADC Channel 7 selected. 1000b - ADC Channel 8 selected. 1001b - ADC Channel 9 selected. 1010b - ADC Channel 10 selected. 1011b - ADC Channel 11 selected. 1100b - ADC Channel 12 selected. 1101b - ADC Channel 13 selected. 1110b - ADC Channel 14 selected. 1111b - ADC Channel 15 selected.
15 —	Reserved
14-13 IE4	Segment 4 done interrupt selection 00b - No interrupt when finished 01b - Generate interrupt on Done0 when segment 4 finish. 10b - Generate interrupt on Done1 when segment 4 finish. 11b - Generate interrupt on Done2 when segment 4 finish.
12 B2B4	Segment 4 B2B 0b - Disable B2B. Wait until delay value defined by TRIG4_COUNTER[SAMPLE_INTERVAL] is reached 1b - Enable B2B. When Segment 0 finished (ADC COCO) then automatically trigger next ADC conversion, no need to wait until interval delay reached.
11-4 HWTS4	Segment 4 HWTS ADC hardware trigger selection. This field will control the ext_hwts of ADC (see the ADC_ETC block diagram). **NOTE:** Please keep only one bit set at the same time. 00000000b - no trigger selected 00000001b - ADC TRIG0 selected 00000010b - ADC TRIG1 selected 00000100b - ADC TRIG2 selected 00001000b - ADC TRIG3 selected 00010000b - ADC TRIG4 selected 00100000b - ADC TRIG5 selected 01000000b - ADC TRIG6 selected 10000000b - ADC TRIG7 selected
3-0 CSEL4	ADC channel selection 0000b - ADC Channel 0 selected 0001b - ADC Channel 1 selected.

Field	Function
	0010b - ADC Channel 2 selected. 0011b - ADC Channel 3 selected. 0100b - ADC Channel 4 selected. 0101b - ADC Channel 5 selected. 0110b - ADC Channel 6 selected. 0111b - ADC Channel 7 selected. 1000b - ADC Channel 8 selected. 1001b - ADC Channel 9 selected. 1010b - ADC Channel 10 selected. 1011b - ADC Channel 11 selected. 1100b - ADC Channel 12 selected. 1101b - ADC Channel 13 selected. 1110b - ADC Channel 14 selected. 1111b - ADC Channel 15 selected.

67.6.1.11 ETC_TRIG Chain 6/7 Register (TRIG0_CHAIN_7_6 - TRIG7_CHAIN_7_6)

67.6.1.11.1 Offset

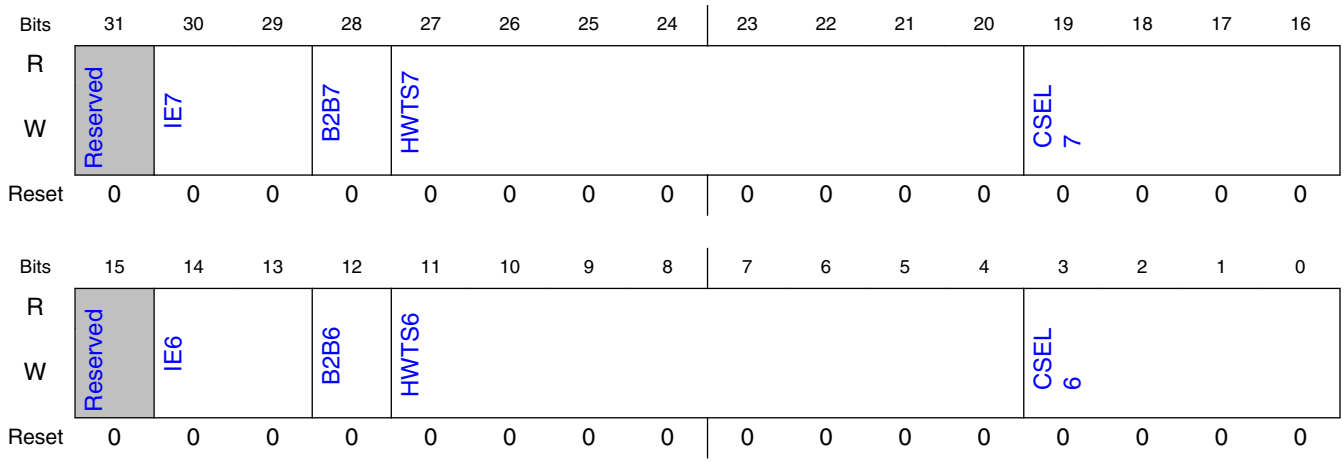
For a = 0 to 7:

Register	Offset
TRIGa_CHAIN_7_6	24h + (a × 28h)

67.6.1.11.2 Function

This register controls ETC_TRIG Chain 6/7 configuration, such as enabling and selecting interrupt, setting back-to-back mode,selecting ADC trigger and trigger command.

67.6.1.11.3 Diagram



67.6.1.11.4 Fields

Field	Function
31 —	Reserved
30-29 IE7	Segment 7 done interrupt selection 00b - No interrupt when finished 01b - Generate interrupt on Done0 when segment 7 finish. 10b - Generate interrupt on Done1 when segment 7 finish. 11b - Generate interrupt on Done2 when segment 7 finish.
28 B2B7	Segment 7 B2B 0b - Disable B2B. Wait until delay value defined by TRIG7_COUNTER[SAMPLE_INTERVAL] is reached 1b - Enable B2B. When Segment 0 finished (ADC COCO) then automatically trigger next ADC conversion, no need to wait until interval delay reached.
27-20 HWTS7	Segment 7 HWTS ADC hardware trigger selection. This field will control the ext_hwts of ADC (see the ADC_ETC block diagram). **NOTE:** Please keep only one bit set at the same time. 00000000b - no trigger selected 00000001b - ADC TRIG0 selected 00000010b - ADC TRIG1 selected 00000100b - ADC TRIG2 selected 00001000b - ADC TRIG3 selected 00010000b - ADC TRIG4 selected 00100000b - ADC TRIG5 selected 01000000b - ADC TRIG6 selected 10000000b - ADC TRIG7 selected
19-16 CSEL7	ADC channel selection 0000b - ADC Channel 0 selected. 0001b - ADC Channel 1 selected. 0010b - ADC Channel 2 selected.

Table continues on the next page...

Field	Function
	0011b - ADC Channel 3 selected. 0100b - ADC Channel 4 selected. 0101b - ADC Channel 5 selected. 0110b - ADC Channel 6 selected. 0111b - ADC Channel 7 selected. 1000b - ADC Channel 8 selected. 1001b - ADC Channel 9 selected. 1010b - ADC Channel 10 selected. 1011b - ADC Channel 11 selected. 1100b - ADC Channel 12 selected. 1101b - ADC Channel 13 selected. 1110b - ADC Channel 14 selected. 1111b - ADC Channel 15 selected.
15 —	Reserved
14-13 IE6	Segment 6 done interrupt selection 00b - No interrupt when finished 01b - Generate interrupt on Done0 when segment 6 finish. 10b - Generate interrupt on Done1 when segment 6 finish. 11b - Generate interrupt on Done2 when segment 6 finish.
12 B2B6	Segment 6 B2B 0b - Disable B2B. Wait until delay value defined by TRIG6_COUNTER[SAMPLE_INTERVAL] is reached 1b - Enable B2B. When Segment 0 finished (ADC COCO) then automatically trigger next ADC conversion, no need to wait until interval delay reached.
11-4 HWTS6	Segment 6 HWTS ADC hardware trigger selection. This field will control the ext_hwts of ADC (see the ADC_ETC block diagram). **NOTE:** Please keep only one bit set at the same time. 00000000b - no trigger selected 00000001b - ADC TRIG0 selected 00000010b - ADC TRIG1 selected 00000100b - ADC TRIG2 selected 00001000b - ADC TRIG3 selected 00010000b - ADC TRIG4 selected 00100000b - ADC TRIG5 selected 01000000b - ADC TRIG6 selected 10000000b - ADC TRIG7 selected
3-0 CSEL6	ADC channel selection 0000b - ADC Channel 0 selected 0001b - ADC Channel 1 selected. 0010b - ADC Channel 2 selected. 0011b - ADC Channel 3 selected. 0100b - ADC Channel 4 selected. 0101b - ADC Channel 5 selected. 0110b - ADC Channel 6 selected. 0111b - ADC Channel 7 selected. 1000b - ADC Channel 8 selected. 1001b - ADC Channel 9 selected. 1010b - ADC Channel 10 selected. 1011b - ADC Channel 11 selected. 1100b - ADC Channel 12 selected.

Field	Function
	1101b - ADC Channel 13 selected. 1110b - ADC Channel 14 selected. 1111b - ADC Channel 15 selected.

67.6.1.12 ETC_TRIG Result Data 1/0 Register (TRIG0_RESULT_1_0 - TRIG7_RESULT_1_0)

67.6.1.12.1 Offset

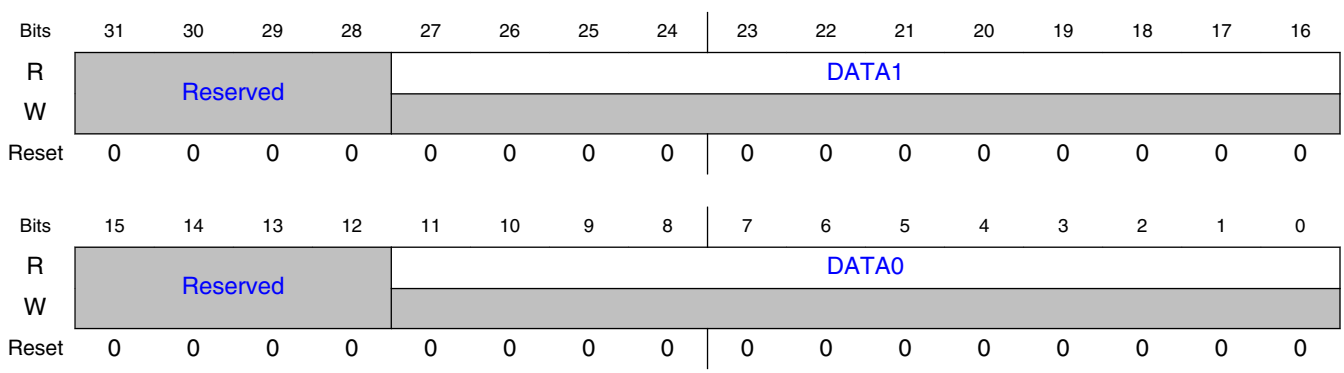
For a = 0 to 7:

Register	Offset
TRIGa_RESULT_1_0	28h + (a × 28h)

67.6.1.12.2 Function

This register stores the results of the ADC conversions corresponding to segment 0 and segment 1 of TRIGa.

67.6.1.12.3 Diagram



67.6.1.12.4 Fields

Field	Function
31-28 —	Reserved

Table continues on the next page...

Field	Function
27-16 DATA1	Result DATA1 **NOTE:** The sign bit from ADC result FIFO is ignored by ETC_TRIG result, so only 12-bit unsigned results is supported by ADC_ETC module.
15-12 —	Reserved
11-0 DATA0	Result DATA0 **NOTE:** The sign bit from ADC result FIFO is ignored by ETC_TRIG result, so only 12-bit unsigned results is supported by ADC_ETC module.

67.6.1.13 ETC_TRIG Result Data 3/2 Register (TRIG0_RESULT_3_2 - TRIG7_RESULT_3_2)

67.6.1.13.1 Offset

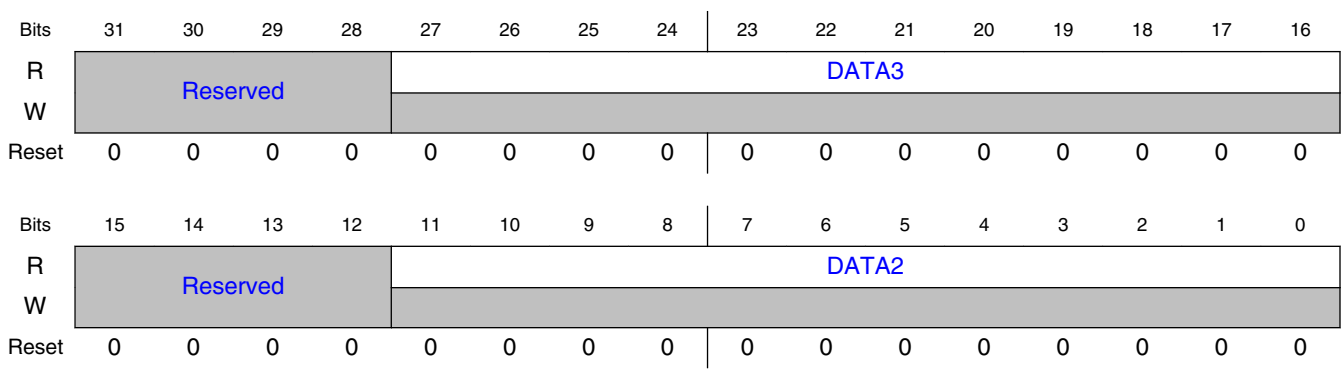
For a = 0 to 7:

Register	Offset
TRIGa_RESULT_3_2	2Ch + (a × 28h)

67.6.1.13.2 Function

This register stores the results of the ADC conversions corresponding to segment 2 and segment 3 of TRIGa.

67.6.1.13.3 Diagram



67.6.1.13.4 Fields

Field	Function
31-28 —	Reserved
27-16 DATA3	Result DATA3 **NOTE:** The sign bit from ADC result FIFO is ignored by ETC_TRIG result, so only 12-bit unsigned results is supported by ADC_ETC module.
15-12 —	Reserved
11-0 DATA2	Result DATA2 **NOTE:** The sign bit from ADC result FIFO is ignored by ETC_TRIG result, so only 12-bit unsigned results is supported by ADC_ETC module.

67.6.1.14 ETC_TRIG Result Data 5/4 Register (TRIG0_RESULT_5_4 - TRIG7_RESULT_5_4)

67.6.1.14.1 Offset

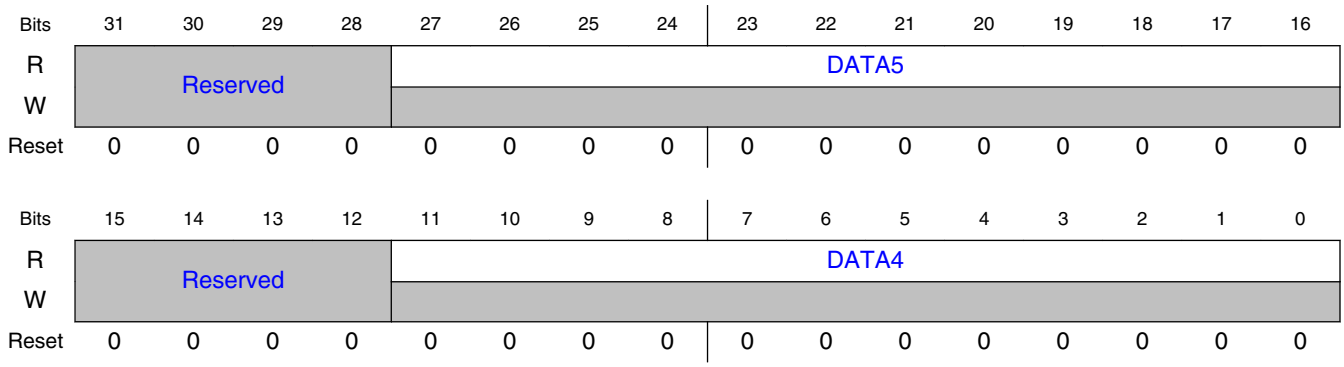
For a = 0 to 7:

Register	Offset
TRIGa_RESULT_5_4	30h + (a × 28h)

67.6.1.14.2 Function

This register stores the results of the ADC conversions corresponding to segment 4 and segment 5 of TRIGa.

67.6.1.14.3 Diagram



67.6.1.14.4 Fields

Field	Function
31-28 —	Reserved
27-16 DATA5	Result DATA5 **NOTE:** The sign bit from ADC result FIFO is ignored by ETC_TRIG result, so only 12-bit unsigned results is supported by ADC_ETC module.
15-12 —	Reserved
11-0 DATA4	Result DATA4 **NOTE:** The sign bit from ADC result FIFO is ignored by ETC_TRIG result, so only 12-bit unsigned results is supported by ADC_ETC module.

67.6.1.15 ETC_TRIG Result Data 7/6 Register (TRIG0_RESULT_7_6 - TRIG7_RESULT_7_6)

67.6.1.15.1 Offset

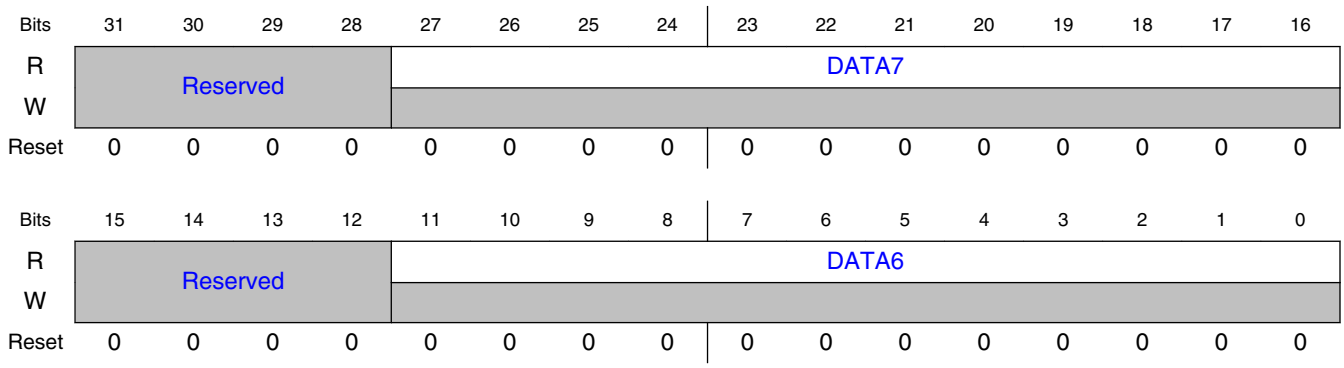
For a = 0 to 7:

Register	Offset
TRIGa_RESULT_7_6	34h + (a × 28h)

67.6.1.15.2 Function

This register stores the results of the ADC conversions corresponding to segment 6 and segment 7 of TRIGa.

67.6.1.15.3 Diagram



67.6.1.15.4 Fields

Field	Function
31-28 —	Reserved
27-16 DATA7	Result DATA7 **NOTE:** The sign bit from ADC result FIFO is ignored by ETC_TRIG result, so only 12-bit unsigned results is supported by ADC_ETC module.
15-12 —	Reserved
11-0 DATA6	Result DATA6 **NOTE:** The sign bit from ADC result FIFO is ignored by ETC_TRIG result, so only 12-bit unsigned results is supported by ADC_ETC module.

Chapter 68
Touch Screen Controller (TSC)

68.1 Chip-specific TSC information

Table 68-1. Reference links to related information

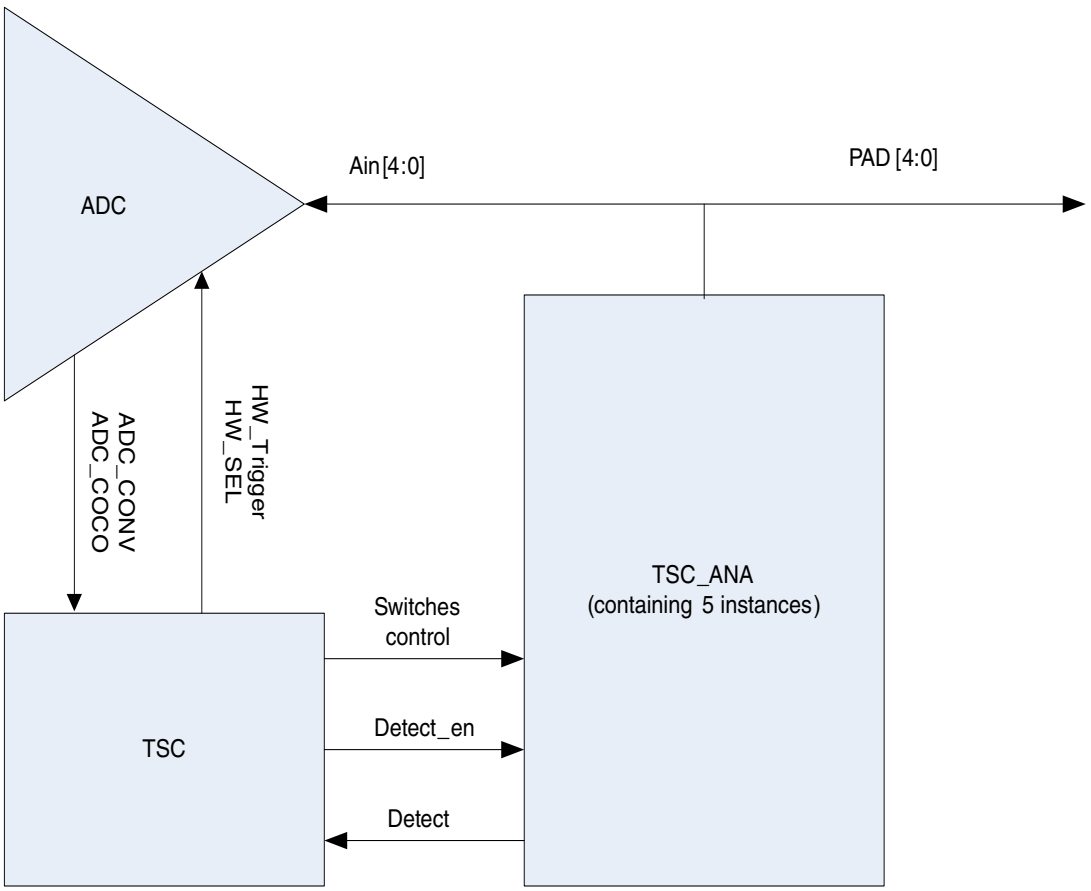
Topic	Related module(s)	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management Clock Control Module (CCM)
Power management	PMU	Power Management Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX
Interrupts, DMA Events and XBAR Assignments	-	Interrupts, DMA Events and XBAR Assignments

68.2 Overview

This block describes the Touch Screen Controller (TSC), which is used for ADC and touch screen analogue block.

TSC is responsible for providing control of ADC and touch screen analogue block to form a touch screen system, which achieves function of touch detection and touch location detection. The controller utilizes ADC hardware trigger function and control switches in touch screen analogue block. The controller only supports 4-wire of 5-wire screen touch modes.

Figure 68-1. TSC block diagram



68.2.1 Features

The features of TSC controller are following.

- Configure registers: 32-bit, fully support sky-blue bus interface
- 4-wire or 5-wire mode of touch screen
- Low power wake up functions
- ADC average function and custom 8-bit, 10-bit, and 12-bit conversion result
- Custom pre-charge and de-glitch threshold time setting
- Total control five analogue groups of switches
- Fully asynchronous interface to ADC and analogue switches
- Easy software operation
- Software takes control of operation flow
- Strong debug functions—enable software recognize the IP as a trnasparent box and operation ouput directly
- Software reset function

68.3 Functional Description

This block works with ADC and TSC analogue to form a touch screen system. The system is responsible to detect touch on screen and measure the coordinate of the touch point. The touch screen controller, which is at the heart of the system, given control to TSC analogue and ADC. TSC analogue provides positive or negative voltage to the screen if the touch screen controller provides relevant signals. The ADC convert coordinates value if requested by touch screen controller.

68.3.1 Operating modes

The TSC can be worked in the following modes.

68.3.1.1 Idle

The TSC always stays at idle status if finish a coordinate measurement. The TSC only starts to detect a touch or starts to measure coordinate value if set the start_sen bit. The TSC returns to idle status after finish current task if disable bit is set by software.

68.3.1.2 Pre-charge

Pre-charge is a preparation for detection stage. Before detection stage, the upper layer of screen is required to charge to positive high. The time required for the pre-charge can be set in the pre-charge_timer.

68.3.1.3 Detection

Detection is a stage to sense touch detect on the screen. This stage is initialed a delay (set by pre-charge_timer) in order to wait enough time for the lower screen layer to achieve even-potential status. After the TSC send out detect enable signal to detect signal from analogue, after receiving the signal, TSC starts to measure coordinates or waiting for the instruction from software according to the value of auto_measure. If auto_measure is set then the TSC automatically measures coordinates after detect a touch. Otherwise TSC

waits for software order after detects a touch (and generates an interrupt). The software can choose to drop the measure (and return back to idle) by setting drop_measure, or start the measure by setting start_measure.

68.3.1.4 Measurement

During the measure stage, the TSC hardware gives control to TSC analogue and ADC. No software operation is needed. If requesting average function, do average configuration for ADC.

68.3.1.5 Data valid check

After measure the coordinate value, TSC do a touch detects again. If no touch has been detected, then previous measured coordinates' value is invalid. Otherwise, the measured coordinates' value is valid.

68.3.1.6 Interrupt

Each interrupt provides three software interface bits: interrupt enable, interrupt signal enable, and interrupt signal.

68.3.1.7 Reset

The TSC has two resets: ipg_reset_b and sw_rst. The ipg_reset_b reset is a hardware reset, which resets all registers in TSC block. While the sw_rst is a software reset, which resets every register except ips directly access ones.

68.3.1.8 Debug mode

The TSC provides fully software control signals. After debug_en has been set, then all TSC outputs will be controlled by software. Software can also observe all TSC inputs through debug interface. Furthermore, the debug registers also provides current state machine states. Software can always check the current hardware state.

68.3.2 Configuration

68.3.2.1 TSC configurations

The following tables provide 4-wire and 5-wire screen touch modes.

Table 68-2. 4-wire screen touch mode

	wiper	ynlr	ypll	xnur	xpul
X measurement	OFF	OFF	OFF	LOW	HIGH
Y measurement	OFF	LOW	HIGH	OFF	OFF
Pre-charge	OFF	OFF	OFF	HIGH (200K, PULL UP)	HIGH (200K, PULL UP)
Intermediate	OFF	OFF	OFF	OFF	HIGH (200K)
Touch screen dection	OFF	LOW	LOW	OFF	HIGH (200K)

Table 68-3. 5-wire screen touch mode

	wiper	ynlr	ypll	xnur	xpul
X measurement	OFF	LOW	HIGH	LOW	HIGH
Y measurement	OFF	LOW	LOW	HIGH	HIGH
Pre-charge	HIGH (200K, PULL UP)	OFF	OFF	OFF	OFF
Intermediate[1]	HIGH (200K)	OFF	OFF	OFF	OFF
Touch screen dection	HIGH (200K)	LOW	LOW	OFF	OFF

1. All other intermediate states switche to OFF.

68.3.2.2 TSC-ADC-TSC analogue configuration

The touch screen controller needs to co-work with ADC and TSC analogue.

The ADC is responsible for analogue value conversion. Following tables show the channels that the ADC used.

Table 68-4. 4-wire screen touch mode

X measurement	Channel 1	Connects to TSC ana YNLR

Table continues on the next page...

Table 68-4. 4-wire screen touch mode (continued)

Y measurement	Channel 3	Connects to TSC ana XNUR

Table 68-5. 5-wire screen touch mode

X measurement	Channel 0	Connects to TSC ana WIPER
Y measurement	Channel 0	Connects to TSC ana WIPER

68.3.2.3 TSC, TSC analogue and ADC connection

The following table describes the connections among the TSC, TSC analogue, and ADC.

Table 68-6. TSC, TSC analogue and ADC connection

TSC controller ports	TSC analogue IP ports	TSC analogue system integration instance Name	ADC Ain ports
wiper_pull_up	pu_en	1	Ain [0]
wiper-200k_pull_up	pull_up-200k_en		
wiper_pull_down	pd_en		
ynlr_pull_up	pu_en	2	Ain [1]
ynlr_200k_pull_up	pull_up_200k_en		
ynlr_pull_down	pd_en		
ypll_pull_up	pu_en	3	Ain [2]
ypll_200k_pull_up	pull_up_200k_en		
ypll_pull_down	pd_en		
xnur_pull_up	pu_en	4	Ain [3]
xnur_200k_pull_up	pull_up_200k_en		
xnur_pull_down	pd_en		
xpul_pull_up	pu_en	5	Ain [4]
xpul_200k_pull_up	pull_up_200k_en		
xpul_pull_down	pd_en		

68.3.2.4 TSC and GPIO

From TSC point of view, it does not care what kinds of PAD used. It can use digital pad or analogue pad.

Table 68-7. The ports used in TSC and GPIO

TSC function ports	TSC analogue instance	ADC Ain pin	GPIO ports
wiper	1	Ain [0]	GPIO_AD_B1_11 pin 21
ynlr	2	Ain [1]	GPIO_AD_B1_12 pin 38 (T4.1)
ypll	3	Ain [2]	GPIO_AD_B1_13 pin 39 (T4.1)
xnur	4	Ain [3]	GPIO_AD_B1_14 pin 26
xpul	5	Ain [4]	GPIO_AD_B1_15 pin 27

NOTE

For the PAD used for TSC, make sure the PAD behavior , such as a wire only connects to the Pin and TSC analogue switches.

For the PAD used for GPIO, for example, has a keeper to prevent the GPIO pad working from a wire only connects the Pin and TSC. In such condition, disable keeper when TSC working.

68.3.2.5 ADC-TSC co-working

The TSC is designed to work with the ADC. Two IP must work together to achieve the correct functions. The coordinate work includes two steps:

1. TSC starts ADC

   TSC sends two signals: trigger and hardware trigger select signal: HWTS[4:0].

   ADC regards triggers as clock to capture the select signal: HWTS. The details of information about trigger and HWTS is show in the ADC hardware trigger chapter. The TSC hardware makes sure that the HWTS signals ready one clock cycles before sending trigger signals. Make sure that the trigger delay is less than HWTS delay + one clock cycle.

   The TSC send HWTS as in the following table:

	TSC 4-wire mode	TSC 5-wire mode
x-coordinate measure	HWTS = 5'b01000	HWTS = 5'b10000
y-coordinate measure	HWTS = 5'b00010	HWTS = 5'b10000

TSC HWTS[4:0] connects to the ADC HWTS[4:0]. The integration change will lead change in driver implementation. On ADC side, HWTS = 1 << x indicates the x logic channel is selected to start hardware ADC conversion. Configure ADC_HCx to configure x logic channel and conversion will store in ADC_Rx.

Each ADC hardware trigger logic channel can be configured to one of many physical channels. The details of the ADC configuration, refer to the ADC hardware trigger chapter. Software driver configures the physical ADC. A sample configuration is shown in the following table.

	TSC 4-wire mode	TSC 5-wire mode
x-coordinate measure	ADC measures channel 1 or 2, so configure ADC_HC3[ADCH] to 1 or 2.	ADC measure channel 0, so configure ADC_HC[4] to 0.
y-coordinate measure	ADC measures channel3 or 4, so configure ADC_HC1[ADCH] to 3 or 4.	ADC measures channel 0, so configure ADC_HC[4] to 0.

Before TSC starts work, software driver configure ADC_HCx. TSC hardware automatically sends trigger and HWTS at the correct time.

2. ADC sends back converted value and complete signals to TSC.
When ADC finishes conversion, the ADC would send a complete signal to TSC. A set of handshake signals between ADC and TSC makes sure that the TSC could receive complete signals, store correct ADC converted value, and clear ADC coco signals at correct order. All parts of this step are automatically done by hardware.

NOTE

For different ADC conversion modes, such 8-bit, 10-bit, 12-bit mode, average mode, please configure in ADC registers.

68.4 TSC Memory Map/Register Definition

The TSC Memory Map and Register Definition is described in the following section.

TSC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400E_0000	Basic Setting (TSC_BASIC_SETTING)	32	R/W	0000_0000h	68.4.1/3495
400E_0010	Pre-charge Time (TSC_PRE_CHARGE_TIME)	32	R/W	0000_0000h	68.4.2/3496
400E_0020	Flow Control (TSC_FLOW_CONTROL)	32	R/W	0000_0000h	68.4.3/3496

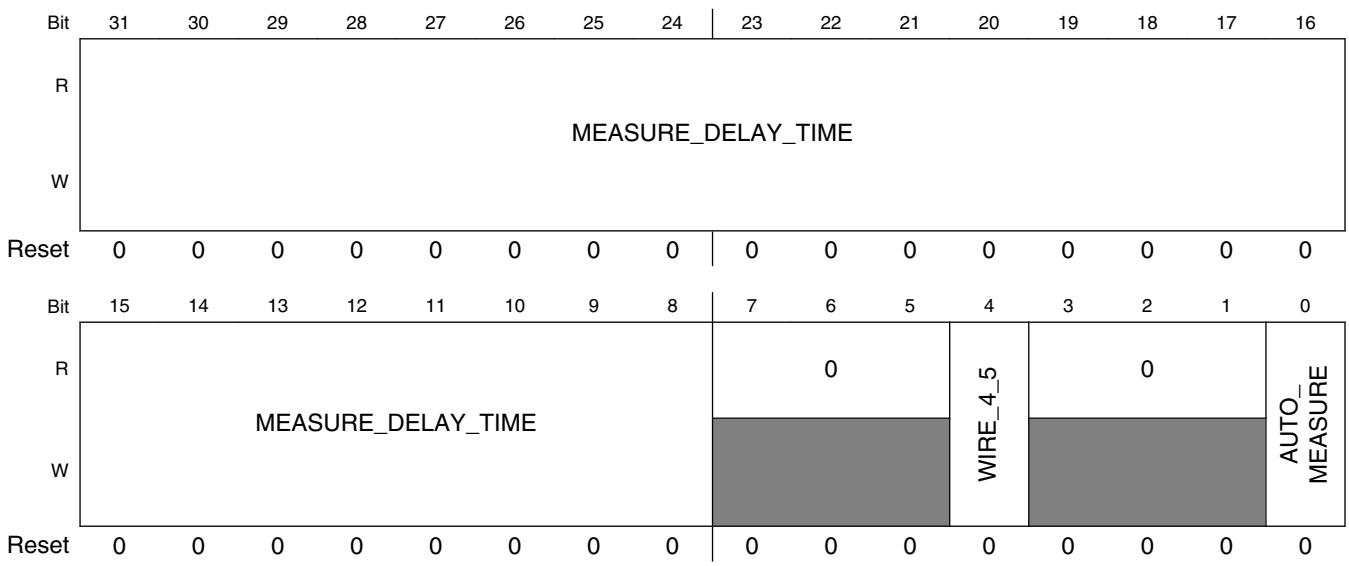
Table continues on the next page...

TSC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400E_0030	Measure Value (TSC_MEASEURE_VALUE)	32	R	0000_0000h	68.4.4/3497
400E_0040	Interrupt Enable (TSC_INT_EN)	32	R/W	0000_0000h	68.4.5/3498
400E_0050	Interrupt Signal Enable (TSC_INT_SIG_EN)	32	R/W	0000_0000h	68.4.6/3499
400E_0060	Intterrupt Status (TSC_INT_STATUS)	32	R/W	0000_0000h	68.4.7/3500
400E_0070	Debug Mode Register (TSC_DEBUG_MODE)	32	R/W	0000_0000h	68.4.8/3502
400E_0080	Debug Mode Register 2 (TSC_DEBUG_MODE2)	32	R/W	0000_0000h	68.4.9/3504

68.4.1 Basic Setting (TSC_BASIC_SETTING)

Address: 400E_0000h base + 0h offset = 400E_0000h



TSC_BASIC_SETTING field descriptions

Field	Description
31–8 MEASURE_ DELAY_TIME	Measure Delay Time Before X-axis or Y-axis measurement, the screen need some time before even potential distribution ready.
7–5 Reserved	This read-only field is reserved and always has the value 0.
4 WIRE_4_5	4/5 Wire detection 4-Wire Detection Mode or 5-Wire Detection Mode 0 4-Wire Detection Mode 1 5-Wire Detection Mode
3–1 Reserved	This read-only field is reserved and always has the value 0.

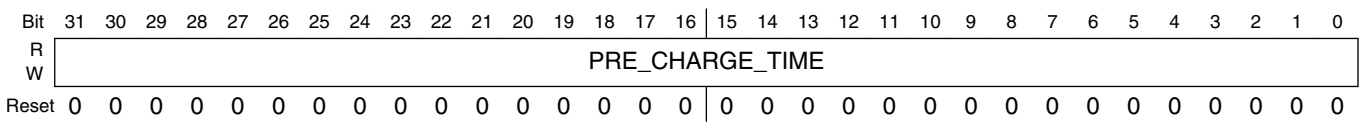
Table continues on the next page...

TSC_BASIC_SETTING field descriptions (continued)

Field	Description
0 AUTO_ MEASURE	Auto Measure This field indicates after detect touch, whether automatic start measurement. SW must make sure that the detect interrupt should be disable. This bit is not HW self-clear. 0 Disable Auto Measure 1 Auto Measure

68.4.2 Pre-charge Time (TSC_PRE_CHARGE_TIME)

Address: 400E_0000h base + 10h offset = 400E_0010h

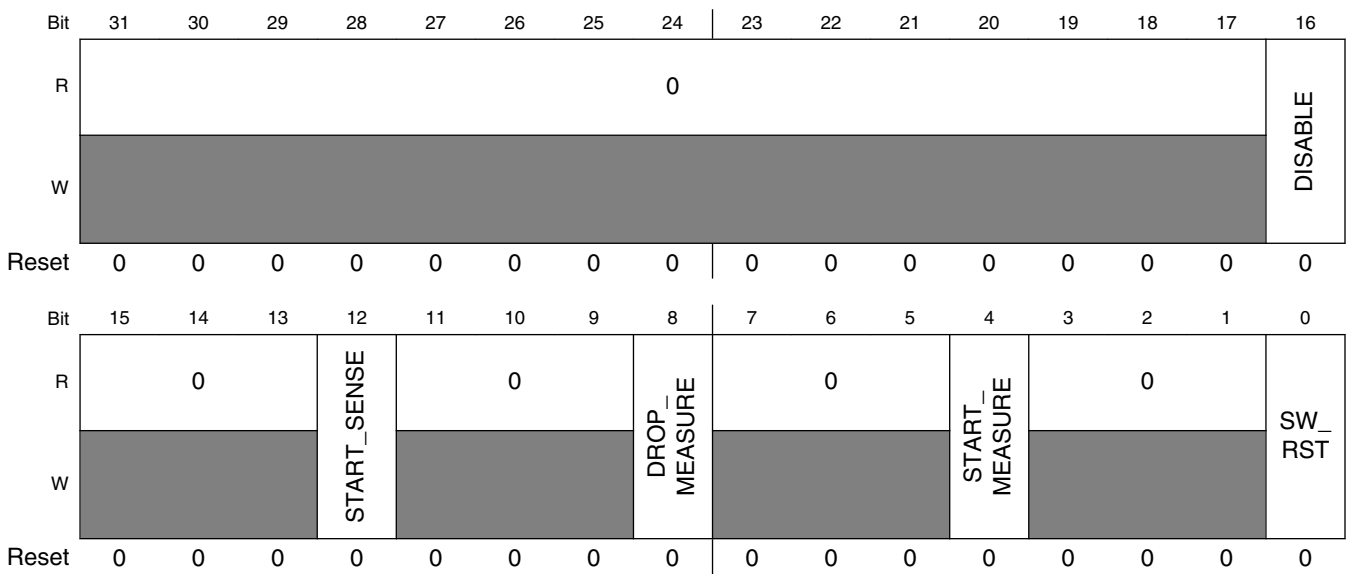


TSC_PRE_CHARGE_TIME field descriptions

Field	Description
PRE_CHARGE_ TIME	Before detection, the top screen needs some time before being pulled up to a high voltage.

68.4.3 Flow Control (TSC_FLOW_CONTROL)

Address: 400E_0000h base + 20h offset = 400E_0020h

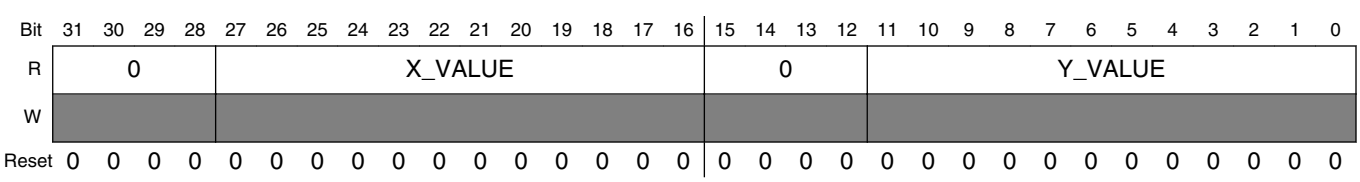


TSC_FLOW_CONTROL field descriptions

Field	Description
31–17 Reserved	This read-only field is reserved and always has the value 0.
16 DISABLE	This bit is for SW disable registers. After set this bit, the hardware returns to idle status after finish the current state operation. SW must check the SW_IDLE bit to confirm that the controller has return to idle status. It's a HW self-clean bit. 0 Leave HW state machine control 1 SW set to idle status
15–13 Reserved	This read-only field is reserved and always has the value 0.
12 START_SENSE	Start Sense It's a HW self-clean bit. 0 Stay at idle status 1 Start sense detection and (if auto_measure set to 1) measure after detect a touch
11–9 Reserved	This read-only field is reserved and always has the value 0.
8 DROP_ MEASURE	Drop Measure This field indicates whether start measure X/Y coordinate value after detect a touch It's a HW self-clean bit. 0 Do not drop measure for now 1 Drop the measure and controller return to idle status
7–5 Reserved	This read-only field is reserved and always has the value 0.
4 START_ MEASURE	Start Measure This field indicates whether start measure X/Y coordinate value after detect a touch It's a self-clean bit. 0 Do not start measure for now 1 Start measure the X/Y coordinate value
3–1 Reserved	This read-only field is reserved and always has the value 0.
0 SW_RST	Soft Reset This is a synchronization reset, which reset all HW registers. **NOTE:** All SW accessible registers would keep as it original setting Software reset would be self-clear.

68.4.4 Measure Value (TSC_MEASEURE_VALUE)

Address: 400E_0000h base + 30h offset = 400E_0030h

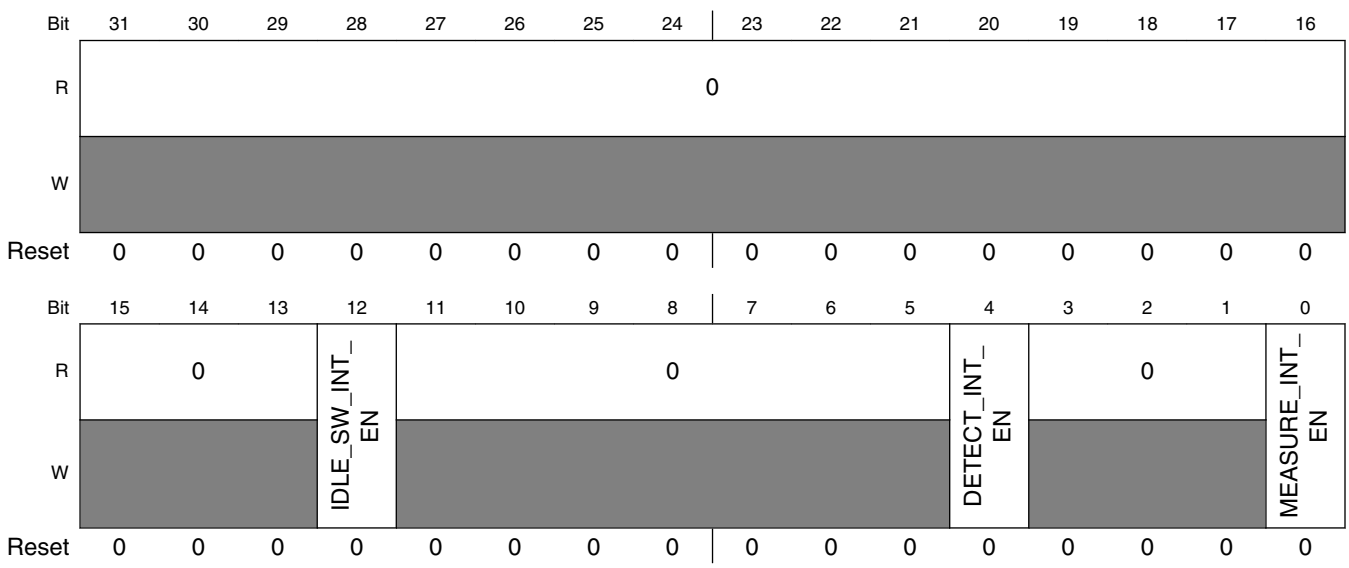


TSC_MEASEURE_VALUE field descriptions

Field	Description
31–28 Reserved	This read-only field is reserved and always has the value 0.
27–16 X_VALUE	X Value Coordinate value **NOTE:** It is an ADC conversion value, SW need to convert it to fit screen size.
15–12 Reserved	This read-only field is reserved and always has the value 0.
Y_VALUE	Y Value Y coordinate value, note, it is an ADC conversion value, SW need to convert it to fit screen size

68.4.5 Interrupt Enable (TSC_INT_EN)

Address: 400E_0000h base + 40h offset = 400E_0040h



TSC_INT_EN field descriptions

Field	Description
31–13 Reserved	This read-only field is reserved and always has the value 0.
12 IDLE_SW_INT_EN	Idle Software Interrupt Enable This field indicates whether enable the software return to idle status interrupt. This interrupt is generated if the controller has return to idle status. This bit is only valid if disable bit in flow control register set to 1'b1. 0 Disable idle software interrupt 1 Enable idle software interrupt

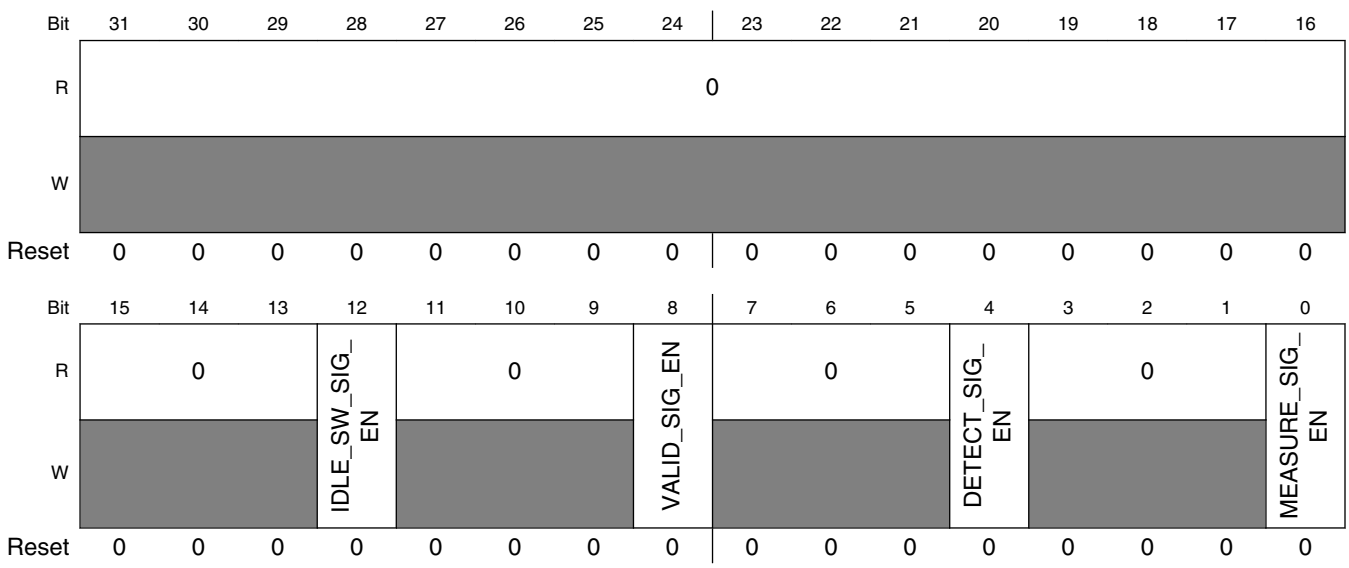
Table continues on the next page...

TSC_INT_EN field descriptions (continued)

Field	Description
11–5 Reserved	Reserved This read-only field is reserved and always has the value 0.
4 DETECT_INT_EN	Detect Interrupt Enable This field indicates whether enable the detect interrupt. This interrupt is generated if there is a touch detect after pre-charge ready. 0 Disable detect interrupt 1 Enable detect interrupt
3–1 Reserved	This read-only field is reserved and always has the value 0.
0 MEASURE_INT_EN	Measure Interrupt Enable This field indicates whether enable the measure interrupt. This interrupt is generated after the touch detection which follows measurement. 0 Disable measure interrupt 1 Enable measure interrupt

68.4.6 Interrupt Signal Enable (TSC_INT_SIG_EN)

Address: 400E_0000h base + 50h offset = 400E_0050h



TSC_INT_SIG_EN field descriptions

Field	Description
31–13 Reserved	This read-only field is reserved and always has the value 0.

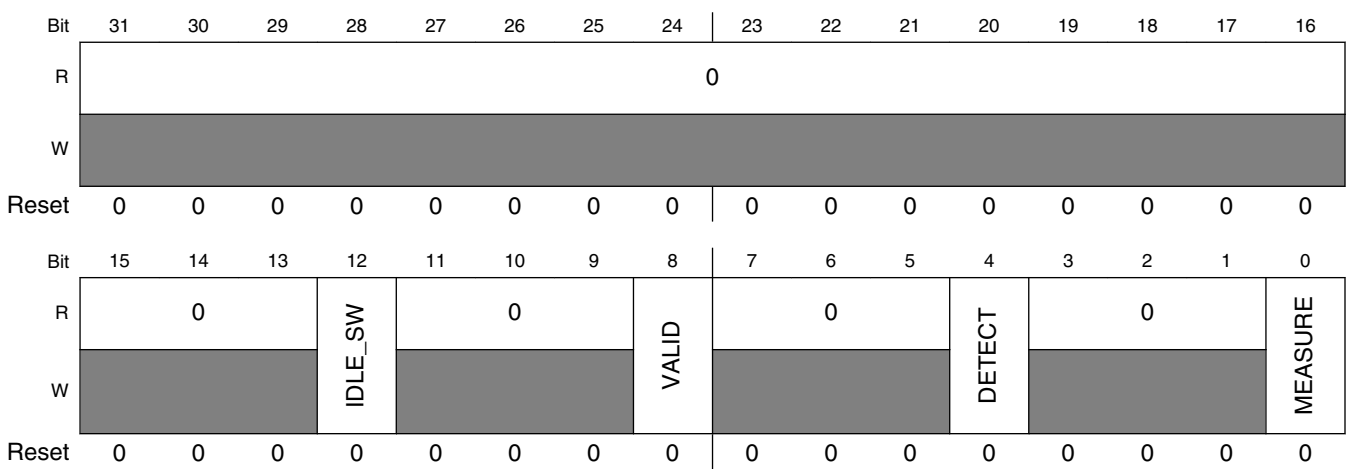
Table continues on the next page...

TSC_INT_SIG_EN field descriptions (continued)

Field	Description
12 IDLE_SW_SIG_EN	Idle Software Signal Enable This field indicates whether enable the software return to idle status. This signal is generated if the controller has return to idle status. This bit is only valid if disable bit in flow control register set to 1'b1. 0 Disable idle software signal 1 Enable idle software signal
11–9 Reserved	This read-only field is reserved and always has the value 0.
8 VALID_SIG_EN	Valid Signal Enable This field indicates whether enable the valid signal. This field generated at the same time with MEASURE_SIG_EN. 0 Disable valid signal 1 Enable valid signal
7–5 Reserved	This read-only field is reserved and always has the value 0.
4 DETECT_SIG_EN	Detect Signal Enable This field indicates whether enable the detect signal. 0 Disable detect signal 1 Enable detect signal
3–1 Reserved	This read-only field is reserved and always has the value 0.
0 MEASURE_SIG_EN	Measure Signal Enable This field indicates whether enable the measure signal.

68.4.7 Intterrupt Status (TSC_INT_STATUS)

Address: 400E_0000h base + 60h offset = 400E_0060h



TSC_INT_STATUS field descriptions

Field	Description
31–13 Reserved	This read-only field is reserved and always has the value 0.
12 IDLE_SW	Idle Software This field indicates whether the state machine return to idle status. This signal is generated if the controller has return to idle status. This bit is only valid if disable bit in flow control register set to 1'b1. 0 Haven't return to idle status 1 Already return to idle status
11–9 Reserved	This read-only field is reserved and always has the value 0.
8 VALID	Valid Signal This field indicats whether the measure value is valid. This field generated at the same time with MEASURE_SIG. 0 There is no touch detected after measurement, indicates that the measured value is not valid 1 There is touch detection after measurement, indicates that the measure is valid
7–5 Reserved	This read-only field is reserved and always has the value 0.
4 DETECT	Detect Signal This field indicates whether there is a detect signal. 0 Does not exist a detect signal 1 Exist detect signal
3–1 Reserved	This read-only field is reserved and always has the value 0.
0 MEASURE	Measure Signal This field indicates whether there is a measure signal. 0 Does not exist a measure signal 1 Exist a measure signal

68.4.8 Debug Mode Register (TSC_DEBUG_MODE)

Address: 400E_0000h base + 70h offset = 400E_0070h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0			DEBUG_EN	0	ADC_COCO_CLEAR_DISABLE	ADC_COCO_CLEAR	TRIGGER	0			EXT_HWTS				
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0			ADC_COCO	ADC_CONV_VALUE											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TSC_DEBUG_MODE field descriptions

Field	Description
31–29 Reserved	This read-only field is reserved and always has the value 0.
28 DEBUG_EN	Debug Enable The RO registers in debug_mode and debug_mode2 always reflect the latest value and is not controlled by this bit. All RW SW bits are controlled by this bit. 0 Enable debug mode 1 Disable debug mode
27 Reserved	This read-only field is reserved and always has the value 0.
26 ADC_COCO_ CLEAR_ DISABLE	ADC COCO Clear Disable This bit could prevent TSC HW generates an ADC COCO clear signal. **NOTE:** This bit does not effect ADC_COCO_CLEAR bit. 0 Allow TSC hardware generates ADC COCO clear 1 Prevent TSC from generate ADC COCO clear signal — —
25 ADC_COCO_ CLEAR	ADC Coco Clear Original ADC coco only clear is system read conv result from IPS bus. However, TSC read conv result directly from ADC to lower SW load, so that ADC requires TSC generates a clear signal. 0 No ADC COCO clear 1 Set ADC COCO clear
24 TRIGGER	Trigger Hardware trigger signal to ADC 0 No hardware trigger signal 1 Hardware trigger signal, the signal must last at least 1 ips clock period
23–21 Reserved	This read-only field is reserved and always has the value 0.
20–16 EXT_HWTS	Hardware Trigger Select Signal Hardware trigger select signal, select which channel to start conversion.
15–13 Reserved	This read-only field is reserved and always has the value 0.
12 ADC_COCO	ADC COCO Signal This signal is generated by ADC.
ADC_CONV_ VALUE	ADC Conversion Value This signal is generated by ADC.

68.4.9 Debug Mode Register 2 (TSC_DEBUG_MODE2)

Address: 400E_0000h base + 80h offset = 400E_0080h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0	DE_GLITCH		DETECT_ENABLE_FIVE_WIRE	0			DETECT_ENABLE_FOUR_WIRE	INTERMEDIATE	STATE_MACHINE			0		DETECT_FIVE_WIRE	DETECT_FOUR_WIRE
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0	WIPER_200K_PULL_UP	WIPER_PULL_UP	WIPER_PULL_DOWN	YNLR_200K_PULL_UP	YNLR_PULL_UP	YNLR_PULL_DOWN	YPLL_200K_PULL_UP	YPLL_PULL_UP	YPLL_PULL_DOWN	XNUR_200K_PULL_UP	XNUR_PULL_UP	XNUR_PULL_DOWN	XPUL_200K_PULL_UP	XPUL_PULL_UP	XPUL_PULL_DOWN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

TSC_DEBUG_MODE2 field descriptions

Field	Description
31 Reserved	This read-only field is reserved and always has the value 0.
30–29 DE_GLITCH	This field indicates glitch threshold. The threshold is defined by number of clock cycles. A detect signal is only valid if it is exist longer than threshold. Any signal transfer through detect and length is smaller than threshold is regards as a glitch. 00 • Normal function: 0x1fff ipg clock cycles; • Low power mode: 0x9 low power clock cycles 01 • Normal function: 0xfff ipg clock cycles; • Low power mode: :0x7 low power clock cycles 10 • Normal function: 0x7ff ipg clock cycles; • Low power mode:0x5 low power clock cycles 11 • Normal function: 0x3 ipg clock cycles; • Low power mode:0x3 low power clock cycles
28 DETECT_ ENABLE_FIVE_ WIRE	Detect Enable Five Wire 0 Do not read five wire detect value, read default value from analogue 1 Read five wire detect status from analogue
27–25 Reserved	This read-only field is reserved and always has the value 0.
24 DETECT_ ENABLE_FOUR_ WIRE	Detect Enable Four Wire 0 Do not read four wire detect value, read default value from analogue 1 Read four wire detect status from analogue
23 INTERMEDIATE	Intermediate State It's an intermediate state, between two state machine states, in order to provide better timing for analogue. It lasts for only one cycle. Intermediate state exists after pre-charge, detect, x- measure, y-measure, and 2nd-pre-charge, 2nd-detect. Note, the intermediate after pre-charge and 2nd-pre-charge is different. 0 Not in intermedia 1 Intermedia
22–20 STATE_ MACHINE	State Machine 000 Idle 001 Pre-charge 010 Detect 011 X-measure 100 Y-measure 101 Pre-charge 110 Detect
19–18 Reserved	This read-only field is reserved and always has the value 0.
17 DETECT_FIVE_ WIRE	Detect Five Wire This field in only valid when the touch controller is under detect mode. That is not in idle. **NOTE:** It is in measure mode, not pre-charge mode. This is an asynchronous signal.

Table continues on the next page...

TSC_DEBUG_MODE2 field descriptions (continued)

Field	Description
	0 No detect signal 1 Yes, there is a detect on the touch screen.
16 DETECT_FOUR_WIRE	Detect Four Wire This field in only valid when the touch controller is under detect mode. That is not in idle. **NOTE:** It is in measure mode, not pre-charge mode. This is an asynchronous signal. 0 No detect signal 1 Yes, there is a detect on the touch screen.
15 Reserved	This read-only field is reserved and always has the value 0.
14 WIPER_200K_PULL_UP	Wiper Wire 200K Pull Up Switch This is a control signal of this wiper wire 200k pull up switch. 0 Close the switch 1 Open up the switch
13 WIPER_PULL_UP	Wiper Wire Pull Up Switch This is a control signal of this switch. 0 Close the switch 1 Open up the switch
12 WIPER_PULL_DOWN	Wiper Wire Pull Down Switch This is a control signal of this wiper wire pull down switch. 0 Close the switch 1 Open up the switch
11 YNLR_200K_PULL_UP	YNLR Wire 200K Pull Up Switch This is a control signal of this YNLR wire 200K pull up switch. 0 Close the switch 1 Open up the switch
10 YNLR_PULL_UP	YNLR Wire Pull Up Switch This is a control signal of this YNLR wire pull up switch. 0 Close the switch 1 Open up the switch
9 YNLR_PULL_DOWN	YNLR Wire Pull Down Switch This is a control signal of this YNLR wire pull down switch. 0 Close the switch 1 Open up the switch
8 YPLL_200K_PULL_UP	YPLL Wire 200K Pull Up Switch This is a control signal of this YPLL wire 200k pull up switch.

Table continues on the next page...

TSC_DEBUG_MODE2 field descriptions (continued)

Field	Description
	0 Close the switch 1 Open up the switch
7 YPLL_PULL_UP	YPLL Wire Pull Up Switch This is a control signal of this YPLL wire pull up switch. 0 Close the switch 1 Open the switch
6 YPLL_PULL_DOWN	YPLL Wire Pull Down Switch This is a control signal of this YPLL wire pull down switch. 0 Close the switch 1 Open up the switch
5 XNUR_200K_PULL_UP	XNUR Wire 200K Pull Up Switch This is a control signal of this XNUR wire 200K pull up switch. 0 Close the switch 1 Open up the switch
4 XNUR_PULL_UP	XNUR Wire Pull Up Switch This is a control signal of this XNUR Wire Pull Up Switch. 0 Close the switch 1 Open up the switch
3 XNUR_PULL_DOWN	XNUR Wire Pull Down Switch This is a control signal of this XNUR Wire Pull Down Switch. 0 Close the switch 1 Open up the switch
2 XPUL_200K_PULL_UP	XPUL Wire 200K Pull Up Switch This is a control signal of this XPUL Wire 200K Pull Up Switch. 0 Close the switch 1 Open up the switch
1 XPUL_PULL_UP	XPUL Wire Pull Up Switch This is a control signal of this XPUL Wire Pull Up Switch. 0 Close the switch 1 Open up the switch
0 XPUL_PULL_DOWN	XPUL Wire Pull Down Switch This is a control signal of this XPUL wire pull down switch. 0 Close the switch 1 Open up the switch

Appendix A
Revision History

The following table provides a revision history for this document.

Table A-1. Revision History

Rev. No.	Date	Substantial Changes
0	08/2018	Initial public release.
1	12/2018	Major revision upgrade: updates in several chapters, see "Change summary" in this specific revision for more details.
2	12/2019	Major revision upgrade: updates in several chapters, see "Substantive changes" in this specific revision for more details.
3	07/2021	Major revision upgrade: updates in several chapters, see "Substantive changes" below for more details.

Appendix B
Substantive changes for this revision

Substantive changes for this revision are as follows:

B.1 About this Document Revision History

No substantive changes

B.2 Introduction Revision History

No substantive changes

B.3 Memory Maps Revision History

No substantive changes

B.4 Interrupts, DMA Events, and XBAR Assignments Revision History

- Minor corrections in the tables “CM7 domain interrupt summary” and “XBAR3 Input Assignments”.

B.5 DMAMUX Revision History

- Block guide structure optimized, and added the section “Clocks”.

B.6 eDMA Revision History

- Block guide optimized: e.g. some changes in the section “Block parts”, added the section “Clocks”, and some edits in the register part.

B.7 System Security Revision History

No substantive changes

B.8 System Debug Revision History

- Some corrections: removed “AMBA Trace Bus (ATB)” and “Coresight Embedded Trace Buffer (ETB)” sections.
- Added new sections “Cortex-M7 DAP (CM7DAP)”, “Cross-Trigger Interface (CTI)”, “Timestamp Generator (TSGEN)” and “JTAG-to-SWD change sequence”.

B.9 System Boot Revision History

- Minor corrections in the tables “FlexSPI Configuration block”, “Serial NOR configuration block” and “Valid DCD address ranges”.
- The table “ROM Clock Setting” and related notes repositioned at the section “Clocks at boot time”.
- Removed the table “IOMUX configuration for SD/MMC”, as the related content is already covered in the table “ROM Bootloader Peripheral PinMux”.
- Editorial updates (e.g. some extra notes added) throughout the chapter.

B.10 External Signals and Pin Multiplexing Revision History

No substantive changes

B.11 IOMUXC Revision History

- Block guide structure optimized, minor updates and editorial changes in the register part.

B.12 GPIO Revision History

- Block guide structure optimized, and some editorial updates.

B.13 Clock and Power Management Revision History

- Updated the table “Low Power Mode Definition”.

B.14 CCM Revision History

- Updated the tables “System Clocks, Gating, and Override” and “System Clock Frequency Values”.
- Minor updates and editorial changes in the register part.

B.15 XTALOSC Revision History

No substantive changes

B.16 PMU Revision History

No substantive changes

B.17 GPC Revision History

No substantive changes

B.18 DCDC Revision History

- Some notes added in the section “Chip-specific DCDC information”.
- Block guide structure optimized.

B.19 TEMPMON Revision History

No substantive changes

B.20 SNVS Revision History

- Thorough updates in the whole chapter, and some sections only show up in the Security Reference Manual.

B.21 SRC Revision History

- Block guide structure optimized.

B.22 Fusemap Revision History

- Minor updates in the fusemap tables.

B.23 OCOTP Revision History

- Block guide structure optimized, and minor updates in the register part.

B.24 External Memory Controllers Revision History

No substantive changes

B.25 SEMC Revision History

- Block guide fully optimized, also with new sections added: e.g. “Initialization” and “Application Information”.

B.26 uSDHC Revision History

- Thorough updates in the whole chapter, e.g. in sections “Overview”, “Signals overview”, “Clock generator”, “Application of uSDHC”, and minor updates in the register part.

B.27 FlexSPI Revision History

- Block guide structure optimized, and minor update in the register part.
- Added new sections, such as “Receiving block diagram” and under the “AHB Memory Map definition” section.
- Updated the table “External Signal List”.

B.28 Arm Cortex M7 Revision History

- Block guide structure optimized, and added the CM7_MCM register part.

B.29 NIC Revision History

No substantive changes

B.30 OCRAM Revision History

- Block guide structure optimized, no substantial content change.

B.31 FlexRAM Revision History

- Minor update in the “FlexRAM Clocks” table.

B.32 AIPSTZ Revision History

- Minor correction (“resource domain” related information is not applicable in this device), no substantial change.

B.33 Display and Camera Overview Revision History

- Updated the “CMOS Sensor Interface (CSI)” overview section.

B.34 CSI Revision History

- Block guide structure optimized, and minor update in the register part.

B.35 eLCDIF Revision History

- Block guide structure optimized.

B.36 PXP Revision History

- Block guide structure optimized.

B.37 Audio Overview Revision History

No substantive changes

B.38 SAI Revision History

- Added the table “SAI MCLK Information”, in the Chip-specific information section.
- Block guide structure optimized.

B.39 MQS Revision History

- Block guide structure optimized, no substantial content change.

B.40 SPDIF Revision History

- Block guide structure optimized, no substantial content change.

B.41 ENET Revision History

- Chapter title corrected as “10/100-Mbps Ethernet MAC (ENET)”.
- Block guide structure optimized.

B.42 USB Revision History

No substantive changes

B.43 USB PHY Revision History

No substantive changes

B.44 FLEXCAN Revision History

- Minor update in the register part.

B.45 CANFD/FlexCAN3 Revision History

- Block guide structure optimized.

B.46 KPP Revision History

- Block guide structure optimized.

B.47 LPI2C Revision History

- Block guide structure optimized.

B.48 LPSPI Revision History

- Block guide structure optimized and some contents updated.

B.49 LPUART Revision History

- Block guide fully optimized.

B.50 FlexIO Revision History

- Block guide structure optimized and some contents updated, including the register part.
- Added a new section “Keypad Interface”.

B.51 Timers Overview Revision History

- Updated the sections “Watchdog Timer (WDOG1,2)” and “Watchdog Timer (WDOG3)”.

B.52 GPT Revision History

- Updated the section “External Signals”.

B.53 PIT Revision History

- Removed the section “PIT External Signals”.

B.54 TMR Revision History

- Block guide structure optimized, no substantial content change.

B.55 eFlexPWM Revision History

- Block guide structure optimized and some contents updated, including the register part.

B.56 QDC Revision History

- Module name clarified as Quadrature Decoder (QDC).

- Block guide structure optimized, and minor update in the register part.
- Add a new section “Speed Measurement Method”.

B.57 WDOG1-2 Revision History

No substantive changes

B.58 RTWDOG/WDOG3 Revision History

- Update the section “Chip-specific RTWDOG information”.
- Block guide structure optimized and some contents updated, including the register part.
- Added a new section “Disable Watchdog after Reset”.

B.59 EWM Revision History

- Block guide structure optimized, no substantial content change.

B.60 On Chip Cross Triggers Overview Revision History

No substantive changes

B.61 XBAR A Revision History

- Block guide structure optimized, no substantial content change.

B.62 XBAR B Revision History

- Block guide structure optimized, no substantial content change.

B.63 AOI Revision History

- Block guide structure optimized, no substantial content change.

B.64 Analog Overview Revision History

No substantive changes

B.65 ACMP Revision History

No substantive changes

B.66 ADC Revision History

- Updated the COCOn bitfields in “Status register for HW triggers (ADCx_HS)”.

B.67 ADC_ETC Revision History

- Block guide fully optimized, also with new sections added: e.g. “Initialization” and “Application Information”, and thorough editorial updates in the register part.

B.68 TSC Revision History

No substantive changes

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors. In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Security — Customer understands that all NXP products may be subject to unidentified or documented vulnerabilities. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP. NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

Document Number IMXRT1060RM
Revision 3, 07/2021

arm

NXP

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Translations — A non-English (translated) version of a document is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

NXP, the NXP logo, NXP SECURE CONNECTIONS FOR A SMARTER WORLD, Freescale,the Freescale logo,Kinetis, are trademarks of NXP B.V. All other product or service names are the property of their respective owners. AMBA, Arm, Arm7, Arm7TDMI, Arm9, Arm11, Artisan, big.LITTLE, Cordio, CoreLink, CoreSight, Cortex, DesignStart, DynamIQ, Jazelle, Keil, Mali, Mbed, Mbed Enabled, NEON, POP, RealView, SecurCore, Socrates, Thumb, TrustZone, ULINK, ULINK2, ULINK-ME, ULINK-PLUS, ULINKpro, µVision, Versatile are trademarks or registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. The related technology may be protected by any or all of patents, copyrights, designs and trade secrets. All rights reserved.Oracle and Java are registered trademarks of Oracle and/or its affiliates.The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© 2021 NXP B.V.

Document Number IMXRT1060RM
Revision 3, 07/2021