i.MX RT1010 Processor Reference Manual

Document Number: IMXRT1010RM
Rev. 0, 09/2019

Contents

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

Chapter 1
About this Document

Chapter 2
Introduction

Chapter 3
Memory Maps

Chapter 4
Interrupts, DMA Events, and XBAR Assignments

Chapter 5
Direct Memory Access Multiplexer (DMAMUX)

Chapter 6
Enhanced Direct Memory Access (eDMA)

Chapter 7
System Security

Chapter 8
System Debug

Chapter 9
System Boot

Chapter 10
External Signals and Pin Multiplexing

Chapter 11
IOMUX Controller (IOMUXC)

Chapter 12
General Purpose Input/Output (GPIO)

Chapter 13
Clock and Power Management

Chapter 14
Clock Controller Module (CCM)

Chapter 15
Crystal Oscillator (XTALOSC)

Chapter 16
Power Management Unit (PMU)

Chapter 17
General Power Controller (GPC)

Chapter 18
DCDC Converter (DCDC)

Chapter 19
Temperature Monitor (TEMPMON)

Chapter 20
Secure Non-Volatile Storage (SNVS)

Chapter 21
System Reset Controller (SRC)

Chapter 22
Fusemap

Chapter 23
On-Chip OTP Controller (OCOTP_CTRL)

Chapter 24
External Memory Controllers

Chapter 25
FlexSPI Controller

Chapter 26
ARM Cortex M7 Platform

Chapter 27
Network Interconnect Bus System (NIC-301)

Chapter 28
On-Chip RAM Memory Controller (OCRAM)

Chapter 34
Sony/Philips Digital Interface (SPDIF)

Chapter 35
Universal Serial Bus Controller (USB)

Chapter 36
Universal Serial Bus 2.0 Integrated PHY (USB-PHY)

Chapter 37
Keypad Port (KPP)

Chapter 38
Low Power Inter-Integrated Circuit (LPI2C)

Chapter 39
Low Power Serial Peripheral Interface (LPSPI)

Chapter 40
Low Power Universal Asynchronous Receiver/Transmitter (LPUART)

Chapter 41
Flexible I/O (FlexIO)

Chapter 42
Timers Overview

Chapter 43
General Purpose Timer (GPT)

Chapter 44
Periodic Interrupt Timer (PIT)

Chapter 45
Enhanced Flex Pulse Width Modulator (eFlexPWM)

Chapter 46
Watchdog Timer (WDOG1-2)

Chapter 47
RTWDOG (WDOG3)

Chapter 48
External Watchdog Monitor (EWM)

Chapter 49
On Chip Cross Triggers Overview

Chapter 50
Inter-Peripheral Crossbar Switch A (XBARA)

Chapter 51
And-Or-Inverter (AOI)

Chapter 52
Analog Overview

Chapter 53
Analog-to-Digital Converter (ADC)

Chapter 54
ADC External Trigger Control (ADC_ETC)

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 RT1010 Data Sheet - Industrial Products

Table continues on the next page...

Table 1-1. Suggested Reading (continued)

Type	Description	Resource
		i.MX RT1010 Data Sheet - Consumer Products
Chip Errata	The Chip mask set Errata provides additional or corrective information for a particular device mask set.	i.MX RT1010 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 RT1010 Application Notes found here
Web Sites	Product summary page on nxp.com with documentation, software, and resources for the device.	i.MX RT1010 Product Summary Page
	Product documentation page on nxp.com with a list of all documentation related to the device.	i.MX RT1010 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 and 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:

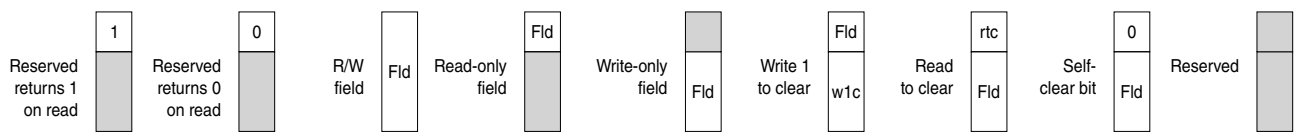


Figure 1-1. Register Field Conventions

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 Signal Conventions

_b, _B

When appended to a signal name, this indicates that a signal is active-low.

$\overline{\text{NEG\_ACTIVE}}$

Overbar also denotes a negative active signal.

UPPERCASE

Package pin names, Block I/O signals.

lowercase

Lowercase is used to indicate internal signals.

1.7 Acronyms and Abbreviations

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

Acronyms and Abbreviated Terms

Term	Meaning
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
BIST	Built-In Self Test
CA/CM	Arm Cortex-A/Cortex-M
CAN	Controller Area Network
CCM	Clock Controller Module
CPU	Central Processing Unit

Table continues on the next page...

Term	Meaning
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
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)
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

Table continues on the next page...

Term	Meaning
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
PCIe	PCI express
PCMCIA	Personal Computer Memory Card International Association
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
ROMCP	ROM Controller with Patch
RTOS	Real-Time Operating System
Rx	Receive
SAI	Synchronous Audio Interface
SCU	Snoop Control Unit
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 Phillips Digital Interface
SPI	Serial Peripheral Interface
SRAM	Static Random-Access Memory
SRC	System Reset Controller
TFT	Thin-Film Transistor
TPIU	Trace Port Interface
TSGEN	Time Stamp Generator

Table continues on the next page...

Term	Meaning
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 RT1010 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.

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.

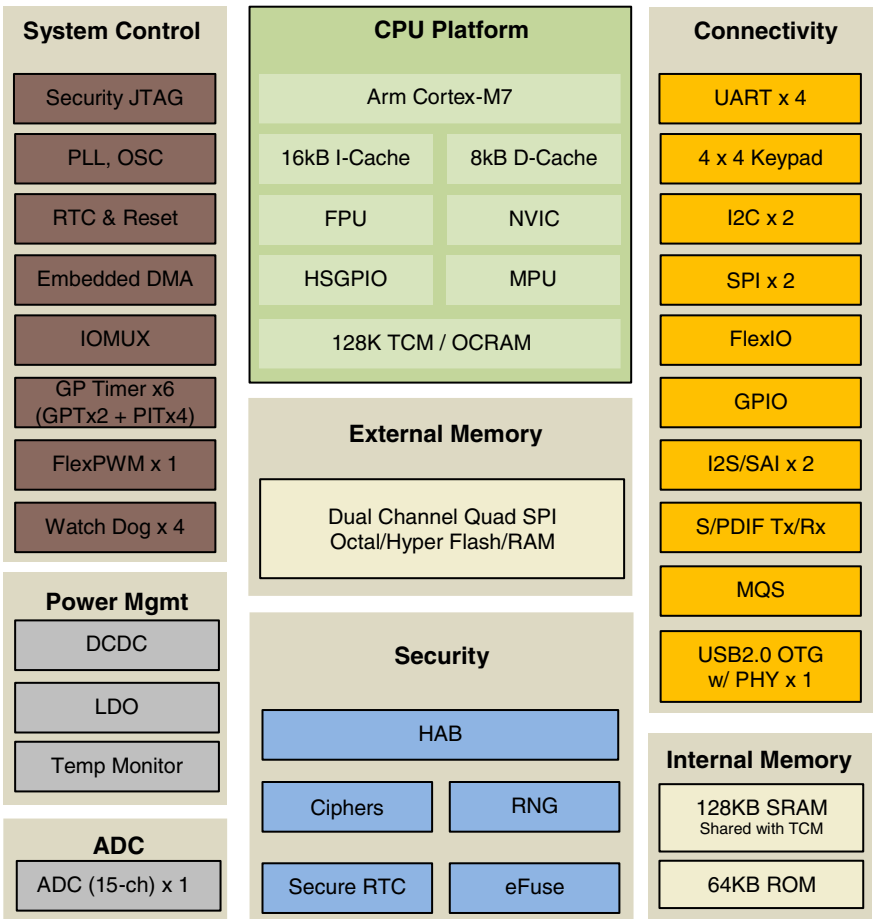


Figure 2-1. Simplified Block Diagram

2.2 Features

The i.MX RT1010 processors are based on Arm®Cortex®-M7 Platform, which have the following features:

Single Arm Cortex-M7 with:

- 16 KB L1 Instruction Cache
- 8 KB L1 Data Cache
- Single-precision FPU (Floating Point Unit)
- Integrated Memory Protection Unit (MPU), up to 16 individual protection regions
- Up to 128 KB I-TCM and D-TCM in total

On Chip Memory:

- Boot ROM (64 KB)
- On-chip RAM, configurable RAM up to 128KB shared with M7 TCM

External Memory Interfaces:

- SPI NOR FLASH
- Single/Dual channel Quad SPI FLASH with XIP support

Audio:
- S/PDIF Input and Output
- 2x SAI (synchronous audio interface) modules supporting I2S, AC97, TDM, and codec/DSP interfaces
- MQS interface for medium quality audio via GPIO pads

Connectivity:
- USB 2.0 OTG controller with integrated PHY interface
- 4x Universal asynchronous receiver/transmitter (UARTs) modules
- 2x I2C modules
- 2x SPI modules
- 1x FlexIO module

Timers:
- 2x General Programmable Timer (GPT)
- 4-channel Periodical Interrupt Timer (PIT)
- 1x FlexPWM

Analog:
- 1x Analog-Digital-Converters (ADC) (upto 16 channels[1])

Security:
- High Assurance Boot (HAB)
- Data Co-Processor (DCP)
  - AES-128, ECB, and CBC mode
  - SHA-1 and SHA-256
  - CRC-32
- FlexSPI with OTFAD (On-the-fly QSPI Flash decryption), AES-128, CTR mode
- True Random Number Generator (TRNG)
- Secure Non-volatile Storage (SNVS)
  - Secure real-time clock (RTC)
  - Zero Master Key (ZMK)
- Secure JTAG Controller (SJC)

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

Power Management:

1. 15 channels available for use due to pad limitation, for this device.

- 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 , IoT, audio, motor control and home appliances, etc.

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.

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
8000_0000	DFFF_FFFF	1.5GB	Reserved
7FC0_0000	7FFF_FFFF	4MB	FlexSPI RX FIFO
7F80_0000	7FBF_FFFF	4MB	FlexSPI TX FIFO
6000_0000	7F7F_FFFF	504MB	FlexSPI/ FlexSPI ciphertext
4800_0000	5FFF_FFFF	384MB	Reserved
4400_0000	47FF_FFFF	64MB	Reserved
4200_0000	43FF_FFFF	32MB	GPIO2
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	Reserved

Table continues on the next page...

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

Start Address	End Address	Size	Description
4040_0000	40FF_FFFF	12MB	Reserved
4030_0000	403F_FFFF	1MB	Reserved
4020_0000	402F_FFFF	1MB	Reserved
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
2022_0000	203F_FFFF	1920KB	OCRAM Reserved
2020_0000	2021_FFFF	128KB	OCRAM
2010_0000	201F_FFFF	1MB	Reserved
2002_0000	200F_FFFF	896KB	DTCM Reserved
2000_0000	2001_FFFF	128KB	DTCM
9000_0000	9FFF_FFFF	256MB	Reserved
6800_0000	8FFF_FFFF	639MB	Reserved
6000_0000	67FF_FFFF	128MB	Reserved
0800_0000	5FFF_FFFF	1527MB	Reserved
0040_0000	07FF_FFFF	124MB	Reserved
0028_0000	003F_FFFF	1536KB	Reserved
0021_0000	0027_FFFF	448KB	ROMCP Reserved
0020_0000	0020_FFFF	64KB	ROMCP
0010_0000	001F_FFFF	1MB	ITCM Reserved
0002_0000	000F_FFFF	896KB	ITCM Reserved
0000_0000	0001_FFFF	128KB	ITCM

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)
400F_8000	400F_BFFF		16KB	SRC(SRC)
400F_4000	400F_7FFF		16KB	GPC
400F_0000	400F_3FFF		16KB	DCP
400E_C000	400E_FFFF		16KB	DMA_CH_MUX
400E_8000	400E_BFFF		16KB	EDMA
400E_4000	400E_7FFF		16KB	USB (USB)
400E_0000	400E_3FFF		16KB	USB (PL301)
400D_C000	400D_FFFF		16KB	CSU
400D_8000	400D_BFFF		16KB	ANALOG
400D_4000	400D_7FFF		16KB	SNVS_HP

Table continues on the next page...

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

Start Address	End Address	Region	Size	NIC Port
400D_0000	400D_3FFF		16KB	WDOG2
400C_C000	400C_FFFF		16KB	TRNG
400C_8000	400C_BFFF		16KB	Reserved
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	FlexSPI
4009_C000	4009_FFFF		16KB	Reserved
4009_8000	4009_BFFF		16KB	XBAR1
4009_4000	4009_7FFF		16KB	AOI
4009_0000	4009_3FFF		16KB	Reserved
4008_C000	4008_FFFF		16KB	Reserved
4008_8000	4008_BFFF		16KB	ADC_ETC
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		16KB	KPP
401F_8000	401F_BFFF		16KB	IOMUXC
401F_4000	401F_7FFF		16KB	OCOTP
401F_0000	401F_3FFF		16KB	GPT2
401E_C000	401E_FFFF	AIPS-2	16KB	GPT1
401E_8000	401E_BFFF		16KB	SAI3
401E_4000	401E_7FFF		16KB	Reserved
401E_0000	401E_3FFF		16KB	SAI1
401D_C000	401D_FFFF		16KB	SPDIF

Table continues on the next page...

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

Start Address	End Address	Region	Size	NIC Port
401D_8000	401D_BFFF		16KB	Reserved
401D_4000	401D_7FFF		16KB	Reserved
401D_0000	401D_3FFF		16KB	Reserved
401C_C000	401C_FFFF		16KB	FlexPWM1
401C_8000	401C_BFFF		16KB	Reserved
401C_4000	401C_7FFF		16KB	Reserved
401C_0000	401C_3FFF		16KB	Reserved
401B_C000	401B_FFFF		16KB	Reserved
401B_8000	401B_BFFF		16KB	GPIO1
401B_4000	401B_7FFF		16KB	Reserved
401B_0000	401B_3FFF		16KB	Reserved
401A_C000	401A_FFFF		16KB	FlexIO
401A_8000	401A_BFFF		16KB	LPI^2C2
401A_4000	401A_7FFF		16KB	LPI^2C1
401A_0000	401A_3FFF		16KB	Reserved
4019_C000	4019_FFFF		16KB	Reserved
4019_8000	4019_BFFF		16KB	LPSPI2
4019_4000	4019_7FFF		16KB	LPSPI1
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	ROMCP
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 PPB detailed memory map.

Table 3-4. 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

Table continues on the next page...

Table 3-4. PPB memory map (continued)

Start Address	End Address	Size	Allocation
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 assignments of interrupts from the Arm domain and from the DMA events .

4.3 CM7 interrupts

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

This table describes the Cortex-M7 interrupt sources:

Table 4-2. CM7 domain interrupt summary

IRQ	Module	Logic	Description
0	eDMA	-	eDMA Channel 0 Transfer Complete
1		-	eDMA Channel 1 Transfer Complete
2		-	eDMA Channel 2 Transfer Complete
3		-	eDMA Channel 3 Transfer Complete
4		-	eDMA Channel 4 Transfer Complete
5		-	eDMA Channel 5 Transfer Complete
6		-	eDMA Channel 6 Transfer Complete
7		-	eDMA Channel 7 Transfer Complete
8		-	eDMA Channel 8 Transfer Complete
9		-	eDMA Channel 9 Transfer Complete
10		-	eDMA Channel 10 Transfer Complete
11		-	eDMA Channel 11 Transfer Complete
12		-	eDMA Channel 12 Transfer Complete
13		-	eDMA Channel 13 Transfer Complete
14		-	eDMA Channel 14 Transfer Complete
15		-	eDMA Channel 15 Transfer Complete
16		-	Error Interrupt, Channels 0-15
17	CM7	-	CTI trigger outputs (internal: CTIIRQ[0])
18		-	CTI trigger outputs (internal: CTIIRQ[1])
19		-	CorePlatform exception IRQ
20	LPUART1	OR	UART1 TX interrupt
			UART1 RX interrupt
21	LPUART2	OR	UART2 TX interrupt
			UART2 RX interrupt
22	LPUART3	OR	UART3 TX interrupt
			UART3 RX interrupt
23	LPUART4	OR	UART4 TX interrupt
			UART4 RX interrupt
24	PIT	OR	PIT timer 0 interrupt
			PIT timer 1 interrupt
			PIT timer 2 interrupt
			PIT timer 3 interrupt
25	USB	-	USBO2 USB OTG1
26	FLEXSPI	-	FlexSPI IRQ
27	CM7	-	FlexRAM address out of range Or access hit IRQ
28	LPI2C1	OR	I2C-1 Interrupt master async
			I2C-1 Interrupt slave async
			I2C-1 Interrupt master
			I2C-1 Interrupt slave

Table continues on the next page...

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

IRQ	Module	Logic	Description
29	LPI2C2	OR	I2C-2 Interrupt master async
			I2C-2 Interrupt slave async
			I2C-2 Interrupt master
			I2C-2 Interrupt slave
30	GPT1	-	All interrupts combined. OR of GPT1 Rollover interrupt line, Input Capture 1 & 2 lines, Output Compare 1,2 &3 Interrupt lines
31	GPT2	-	All interrupts combined. OR of GPT2 Rollover interrupt line, Input Capture 1 & 2 lines, Output Compare 1,2 &3 Interrupt lines
32	LPSPI1	-	LPSPI interrupt request line to the core
33	LPSPI2	-	LPSPI interrupt request line to the core
34	FLEXPWM1	OR	capture PWM0 interrupt
			compare PWM0 interrupt
			reload PWM0 interrupt
35		OR	capture PWM1 interrupt
			compare PWM1 interrupt
			reload PWM1 interrupt
36		OR	capture PWM2 interrupt
			compare PWM2 interrupt
			reload PWM2 interrupt
37		OR	capture PWM3 interrupt
			compare PWM3 interrupt
			reload PWM3 interrupt
38		OR	fault interrupt
			reload error interrupt
39	KPP	-	Keypad Interrupt
40	SRC	-	SRC interrupt request
41	GPR_IRQ	-	Used to notify cores on exception condition while boot
42	CCM	-	CCM, Interrupt Request 1
43		-	CCM, Interrupt Request 2
44	EWM	-	EWM IRQ
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

Table continues on the next page...

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

IRQ	Module	Logic	Description
51		-	IRQ of DCP channel 0
52		-	Reserved
53	TRNG	-	TRNG Interrupt
54	Reserved	-	Reserved
55	Reserved	-	Reserved
56	SAI1	OR	SAI RX interrupt
			SAI RX async interrupt
			SAI TX interrupt
			SAI TX async interrupt
57	RTWDOG	OR	Watchdog Timer reset
			Watchdog Timer Async reset
58	SAI3	OR	SAI RX interrupt
		OR	SAI RX async interrupt
59	SAI3	OR	SAI TX interrupt
		OR	SAI TX async 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	XBAR1	OR	XBAR IRQ0
			XBAR IRQ1
			XBAR IRQ2
			XBAR IRQ3
63	Temperature Monitor	OR	TempSensor low
			TempSensor high
64		-	TempSensor panic
65	USB PHY	-	USBPHY (OTG1 UTMI), Interrupt
66	GPC	-	GPC, Interrupt Request
67	ADC1	OR	ADC1 interrupt
			ADC1 async interrupt
68	FLEXIO1	OR	FlexIO interrupt
			FlexIO asynchronous interrupt
69	DCDC	-	DCDC IRQ
70	GPIO1	-	Combined interrupt indication for GPIO1 signal 0 throughout 15
71	GPIO1	-	Combined interrupt indication for GPIO1 signal 16 throughout 31
72	GPIO2	-	Combined interrupt indication for GPIO2 signal 0 throughout 15
73	GPIO5	-	Combined interrupt indication for GPIO5 signal 0 throughout 15
74	WDOG1	-	Watchdog Timer reset

Table continues on the next page...

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

IRQ	Module	Logic	Description
75	ADC_ETC	-	adc_etc IRQ0
76		-	adc_etc IRQ1
77		-	adc_etc IRQ2
78		-	adc_etc IRQ3
79		-	adc_etc Error IRQ

4.4 DMA Mux

This table 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
		OR	FlexIO DMA Request 1
		OR	FlexIO Async DMA Request 1
1	FLEXIO1	OR	FlexIO DMA Request 4
		OR	FlexIO Async DMA Request 4
		OR	FlexIO DMA Request 5
		OR	FlexIO Async DMA Request 5
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 - 9	Reserved	-	Reserved
10	Reserved	-	Reserved
11	Reserved	-	Reserved
12	Reserved	-	Reserved
13	LPSPI1	OR	LPSPI RX FIFO DMA Request

Table continues on the next page...

Table 4-3. DMA MUX Mapping (continued)

Channel	Module	Logic	Description
		OR	LPSPI RX FIFO Async DMA Request
14	LPSPI1	OR	LPSPI TX FIFO DMA Request
		OR	LPSPI TX FIFO Async DMA Request
15 - 16	Reserved	-	Reserved
17	LPI2C1	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	Reserved	-	Reserved
19	SAI1	-	SAI RX FIFO DMA Request
20	SAI1	-	SAI TX FIFO DMA Request
21 - 22	Reserved	-	Reserved
23	ADC_ETC	-	ADC ETC DMA Request
24	ADC1	-	ADC DMA Request
25 - 26	Reserved	-	Reserved
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
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

Table continues on the next page...

Table 4-3. DMA MUX Mapping (continued)

Channel	Module	Logic	Description
38	FLEXPWM1	-	Write DMA request for value regs of sub-module PWM2
39	FLEXPWM1	-	Write DMA request for value regs of sub-module PWM3
40-47	Reserved	-	Reserved
48-55	Reserved	-	Reserved
56-63	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	FLEXIO1	OR	FlexIO DMA Request 6
		OR	FlexIO Async DMA Request 6
		OR	FlexIO DMA Request 7
		OR	FlexIO Async DMA Request 7
66	LPUART2	OR	UART TX FIFO DMA Request
		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 - 73	Reserved	-	Reserved
74	Reserved	-	Reserved
75-76	Reserved	-	Reserved
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 - 80	Reserved	-	Reserved
81	LPI2C2	OR	I2C Master RX FIFO DMA Request
		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	Reserved	-	Reserved
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 - 90	Reserved	-	Reserved
91	Reserved	-	Reserved
92 - 93	Reserved	-	Reserved
94	XBAR1	-	XBAR DMA Request 2
95	XBAR1	-	XBAR DMA Request 3
96 - 103	Reserved	-	Reserved
104-111	Reserved	-	Reserved
112 - 119	Reserved	-	Reserved
120-127	Reserved	-	Reserved

4.5 XBAR Resource Assignments

This table shows the XBAR resource assignments in the chip.

Table 4-4. XBAR1 Input Assignments

Assigned Input	XBAR1 Input	Gate
Reserved	XBAR1_IN00	-
LOGIC HIGH	XBAR1_IN01	-
Reserved	XBAR1_IN02	-
Reserved	XBAR1_IN03	-
FLEXPWM1_PWM1_OUT_TRIG0	XBAR1_IN04	OR
FLEXPWM1_PWM1_OUT_TRIG1		OR
FLEXPWM1_PWM2_OUT_TRIG0	XBAR1_IN05	OR
FLEXPWM1_PWM2_OUT_TRIG1		OR

Table continues on the next page...

Table 4-4. XBAR1 Input Assignments (continued)

Assigned Input	XBAR1 Input	Gate
FLEXPWM1_PWM3_OUT_TRIG0	XBAR1_IN06	OR
FLEXPWM1_PWM3_OUT_TRIG1		OR
FLEXPWM1_PWM4_OUT_TRIG0	XBAR1_IN07	OR
FLEXPWM1_PWM4_OUT_TRIG1		OR
PIT_TRIGGER0	XBAR1_IN08	-
PIT_TRIGGER1	XBAR1_IN09	-
PIT_TRIGGER2	XBAR1_IN10	-
PIT_TRIGGER3	XBAR1_IN11	-
DMA_DONE0	XBAR1_IN12	-
DMA_DONE1	XBAR1_IN13	-
DMA_DONE2	XBAR1_IN14	-
DMA_DONE3	XBAR1_IN15	-
DMA_DONE4	XBAR1_IN16	-
DMA_DONE5	XBAR1_IN17	-
DMA_DONE6	XBAR1_IN18	-
DMA_DONE7	XBAR1_IN19	-
AOI_OUT0	XBAR1_IN20	-
AOI_OUT1	XBAR1_IN21	-
AOI_OUT2	XBAR1_IN22	-
AOI_OUT3	XBAR1_IN23	-
ADC_ETC_COCO0	XBAR1_IN24	-
ADC_ETC_COCO1	XBAR1_IN25	-
ADC_ETC_COCO2	XBAR1_IN26	-
ADC_ETC_COCO3	XBAR1_IN27	-

Table 4-5. XBAR1 Output Assignments

XBAR1 Output	Assigned Output	Gate
XBAR1_OUT00	FLEXIO1_TRIGGER_IN0	-
XBAR1_OUT01	FLEXIO1_TRIGGER_IN1	-
XBAR1_OUT02	Reserved	-
XBAR1_OUT03	Reserved	-
XBAR1_OUT04	FLEXPWM1_PWM0_EXTA	-
XBAR1_OUT05	FLEXPWM1_PWM1_EXTA	-
XBAR1_OUT06	FLEXPWM1_PWM2_EXTA	-
XBAR1_OUT07	FLEXPWM1_PWM3_EXTA	-
XBAR1_OUT08	FLEXPWM1_PWM0_EXT_SYNC	-
XBAR1_OUT09	FLEXPWM1_PWM1_EXT_SYNC	-
XBAR1_OUT10	FLEXPWM1_PWM2_EXT_SYNC	-

Table continues on the next page...

Table 4-5. XBAR1 Output Assignments (continued)

XBAR1 Output	Assigned Output	Gate
XBAR1_OUT11	FLEXPWM1_PWM3_EXT_SYNC	-
XBAR1_OUT12	FLEXPWM1_EXT_CLK	-
XBAR1_OUT13	FLEXPWM1_FAULT0	-
XBAR1_OUT14	FLEXPWM1_FAULT1	-
XBAR1_OUT15	FLEXPWM1_FAULT2	-
XBAR1_OUT16	FLEXPWM1_FAULT3	-
XBAR1_OUT17	FLEXPWM1_EXT_FORCE	-
XBAR1_OUT18	EWM_EWM_IN	-
XBAR1_OUT19	ADC_ETC_TRIG00	-
XBAR1_OUT20	ADC_ETC_TRIG01	-
XBAR1_OUT21	ADC_ETC_TRIG02	-
XBAR1_OUT22	ADC_ETC_TRIG03	-
XBAR1_OUT23	LPI2C1_TRG_INPUT	-
XBAR1_OUT24	LPI2C2_TRG_INPUT	-
XBAR1_OUT25	LPSPI1_TRG_INPUT	-
XBAR1_OUT26	LPSPI2_TRG_INPUT	-
XBAR1_OUT27	LPUART1_TRG_INPUT	-
XBAR1_OUT28	LPUART2_TRG_INPUT	-
XBAR1_OUT29	LPUART3_TRG_INPUT	-
XBAR1_OUT30	LPUART4_TRG_INPUT	-

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

On this device, only 32-bit write is supported for the DMA_CH_MUX registers.

5.2 Introduction

5.2.1 Overview

The Direct Memory Access Multiplexer (DMAMUX) routes DMA sources, called slots, to any of the 16 DMA channels. This process is illustrated in the following figure.

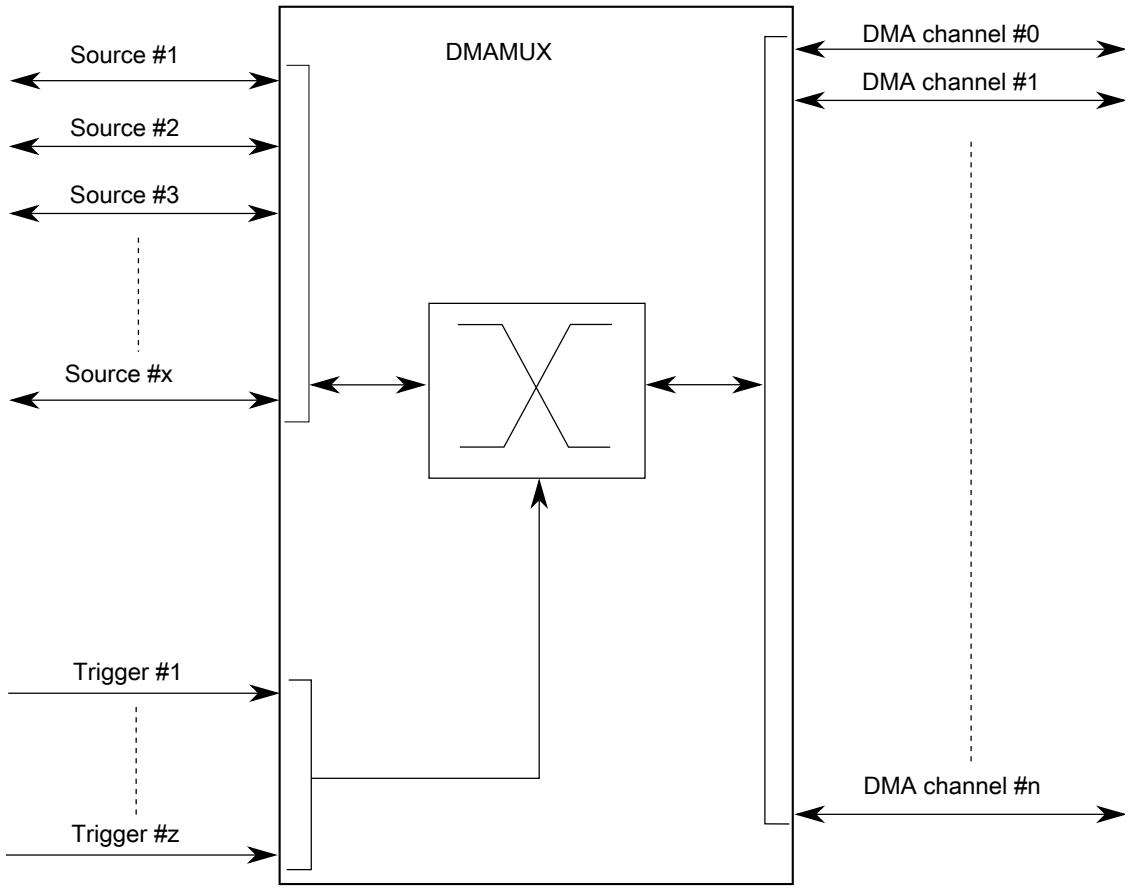


Figure 5-1. DMAMUX block diagram

5.2.2 Features

The DMAMUX module provides these features:

- Up to 128 peripheral slots can be routed to 16 channels.
- 16 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.

5.2.3 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 External signal description

The DMAMUX has no external pins.

5.4 Memory map/register definition

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

5.4.1 DMAMUX register descriptions

5.4.1.1 DMAMUX Memory map

DMAMUX base address: 400E_C000h

Offset	Register	Width (In bits)	Access	Reset value
0h - 3Ch	Channel a Configuration Register (CHCFG0 - CHCFG15)	32	RW	0000_0000h

5.4.1.2 Channel a Configuration Register (CHCFG0 - CHCFG15)

5.4.1.2.1 Offset

For a = 0 to 15:

Register	Offset
CHCFGa	0h + (a × 4h)

5.4.1.2.2 Function

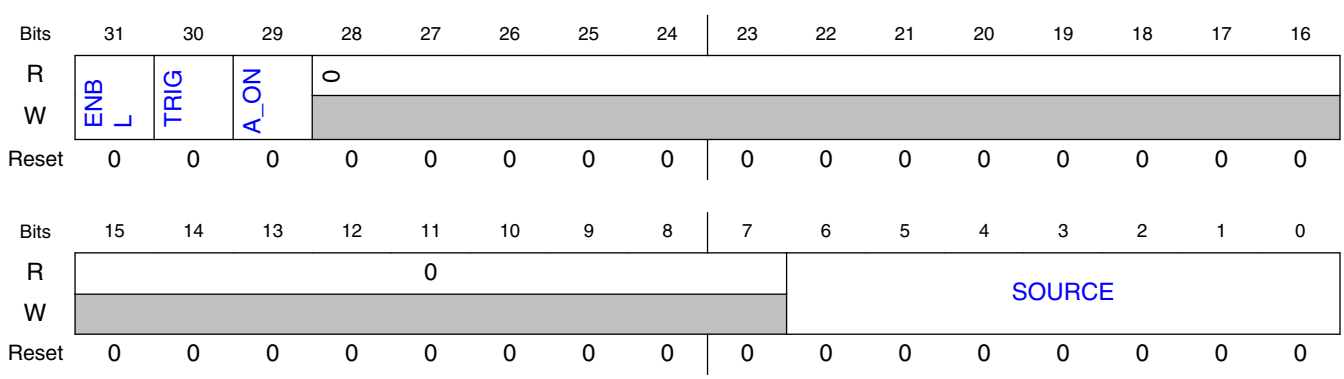
Each of the DMA channels can be independently enabled/disabled and associated with one of the DMA slots (peripheral slots or always-on slots) in the system.

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].

5.4.1.2.3 Diagram



5.4.1.2.4 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 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.

5.5 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

5.5.1 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.

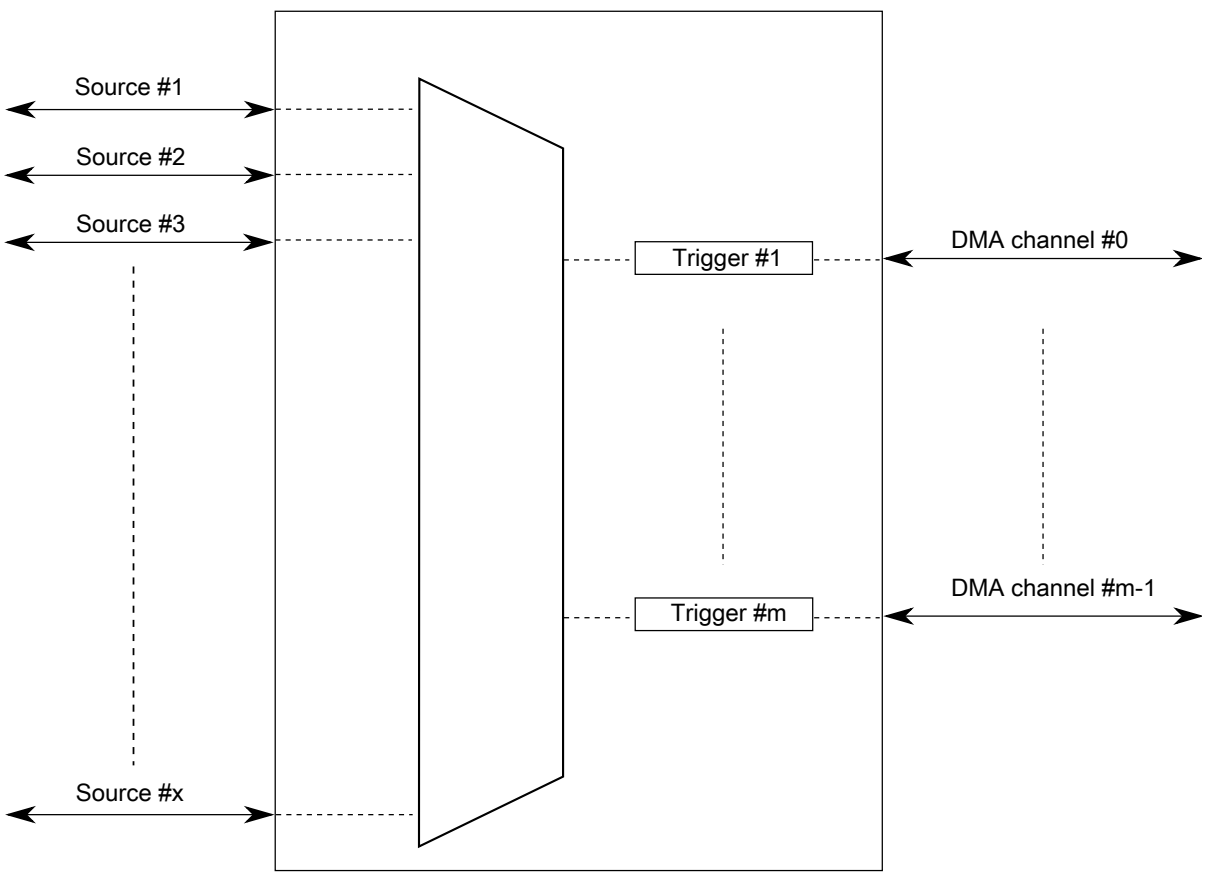


Figure 5-2. DMAMUX triggered channels

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.

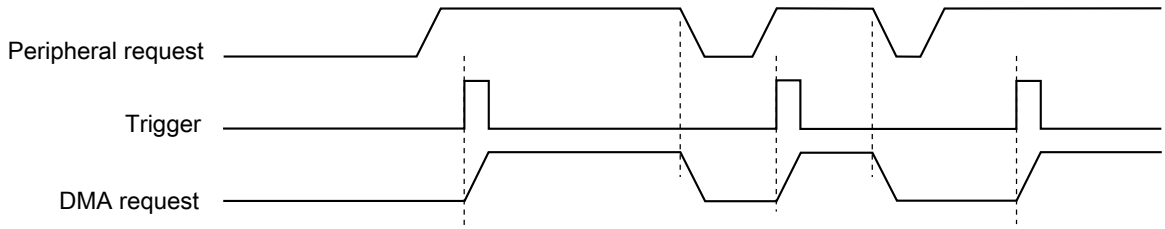


Figure 5-3. DMAMUX channel triggering: normal operation

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.

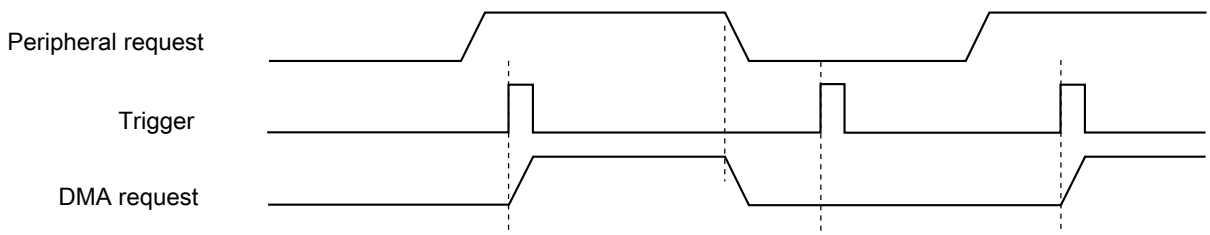


Figure 5-4. DMAMUX channel triggering: ignored trigger

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.

5.5.2 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.

5.6 Initialization/application information

This section provides instructions for initializing the DMA channel MUX.

5.6.1 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.

5.6.2 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;
```

5.6.2.1 Configuration options

Table 5-2. 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

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 Introduction

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 16 channels

6.2.1 eDMA system block diagram

Figure 6-1 illustrates the components of the eDMA system, including the eDMA module ("engine").

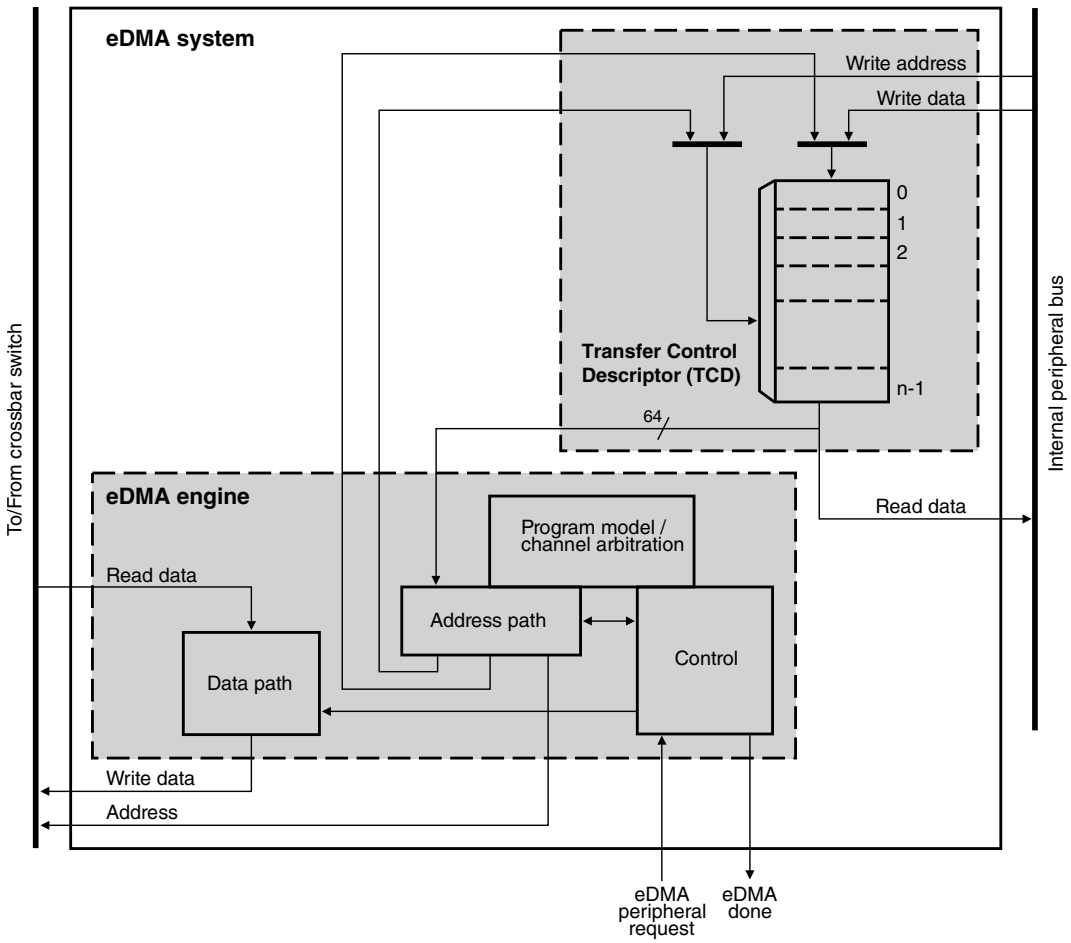


Figure 6-1. eDMA system block diagram

6.2.2 Block parts

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

The eDMA engine is further partitioned into four submodules:

Table 6-2. eDMA engine submodules

Submodule	Function
Address path	This block implements registered versions of two channel transfer control descriptors, channel x and channel y, and manages all master bus-address calculations. All the channels provide the same functionality. This structure allows data transfers associated with one channel to be preempted after the completion of a read/write sequence if a higher priority channel activation is

Table continues on the next page...

Table 6-2. eDMA engine submodules (continued)

Submodule	Function
	asserted while the first channel is active. After a channel is activated, it runs until the minor loop is completed, unless preempted by a higher priority channel. This provides a mechanism (enabled by DCHPRI*n*[ECP]) where a large data move operation can be preempted to minimize the time another channel is blocked from execution. When any channel is selected to execute, the contents of its TCD are read from local memory and loaded into the address path channel x registers for a normal start and into channel y registers for a preemption start. After the minor loop completes execution, the address path hardware writes the new values for the TCD*n*_{SADDR, DADDR, CITER} back to local memory. If the major iteration count is exhausted, additional processing is performed, including the final address pointer updates, reloading the TCD*n*_CITER field, and a possible fetch of the next TCD*n* from memory as part of a scatter/gather operation.
Data path	This block implements the bus master read/write datapath. 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 2-stage pipelined internal bus. The address path module represents the 1st stage of the bus pipeline (address phase), while the data path module implements the 2nd stage of the pipeline (data phase).
Program model/channel arbitration	This block implements the first section of the eDMA programming model as well as the channel arbitration logic. The programming model registers are connected to the internal peripheral bus. The eDMA peripheral request inputs and interrupt request outputs are also connected to this block (via control logic).
Control	This block provides all the control functions for the eDMA engine. For data transfers where the source and destination sizes are equal, the eDMA engine performs a series of source read/ destination write operations until the number of bytes specified in the minor loop byte count has moved. For descriptors where the sizes are not equal, multiple accesses of the smaller size data are required for each reference of the larger size. As an example, if the source size references 16-bit data and the destination is 32-bit data, two reads are performed, then one 32-bit write.

The transfer-control descriptor local memory is further partitioned into:

Table 6-3. Transfer control descriptor memory

Submodule	Description
Memory controller	This logic implements the required dual-ported controller, managing accesses from the eDMA engine as well as references from the internal peripheral bus. As noted earlier, in the event of simultaneous accesses, the eDMA engine is given priority and the peripheral transaction is stalled.
Memory array	TCD storage for each channel's transfer profile.

6.2.3 Features

The eDMA is a highly programmable data-transfer engine optimized to minimize any required intervention from the host processor. It is intended for use in applications where the data size to be transferred is statically known and not defined 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
- 16-channel implementation that 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 for bus mastering the data movement
- Transfer control descriptor (TCD) organized to support 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 reported via programmable interrupt requests
  - One interrupt per channel, which can be asserted at completion of major iteration count
  - 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

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

6.3 Modes of operation

The eDMA operates in the following modes:

Table 6-4. Modes of operation

Mode	Description
Normal	In Normal mode, the eDMA transfers data between a source and a destination. The source and destination can be a memory block or an I/O block capable of operation with the eDMA. A service request initiates a transfer of a specific number of bytes (NBYTES) as specified in the transfer control descriptor (TCD). The minor loop is the sequence of read-write operations that transfers these NBYTES per service request. Each service request executes one iteration of the major loop, which transfers NBYTES of data.
Debug	DMA operation is configurable in Debug mode via the control register: • If CR[EDBG] is cleared, the DMA continues to operate. • If CR[EDBG] is set, the eDMA stops transferring data. If Debug mode is entered while a channel is active, the eDMA continues operation until the channel retires.
Wait	Before entering Wait mode, the DMA attempts to complete its current transfer. After the transfer completes, the device enters Wait mode.

6.4 Memory map/register definition

The eDMA's programming model is partitioned into two regions:

- The first region defines a number of registers providing control functions
- The second region corresponds to the local transfer control descriptor (TCD) memory

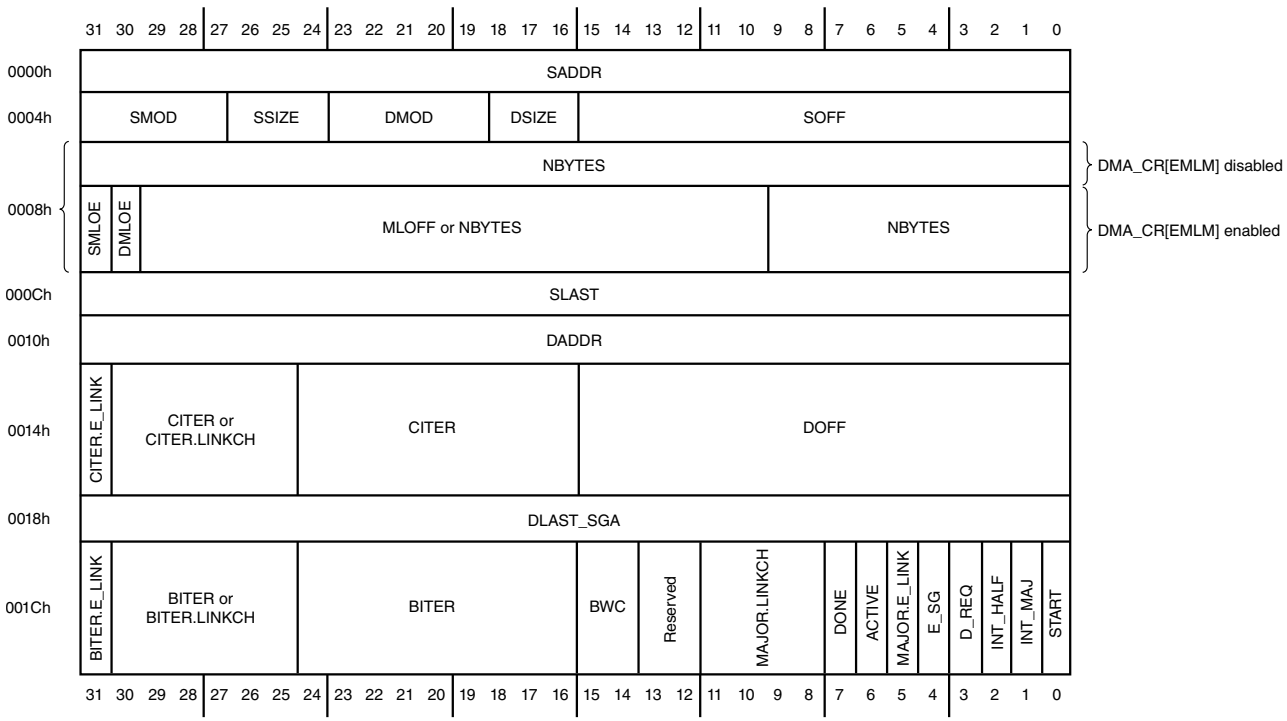
6.4.1 TCD memory

Each channel requires a 32-byte transfer control descriptor for defining the desired data movement operation. The channel descriptors are stored in the local memory in sequential order: channel 0, channel 1, ... channel 15. Each TCD*n* definition is presented as 11 registers of 16 or 32 bits.

6.4.2 TCD initialization

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

6.4.3 TCD structure



6.4.4 Reserved memory and bit fields

- Reading reserved bits in a register returns the value of zero.
- Writes to reserved bits in a register are ignored.
- Reading or writing a reserved memory location generates a bus error.

6.4.5 DMA register descriptions

6.4.5.1 DMA memory map

DMA base address: 400E_8000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Control Register (CR)	32	RW	Table 6-4
4h	Error Status Register (ES)	32	RO	0000_0000h
Ch	Enable Request Register (ERQ)	32	RW	0000_0000h
14h	Enable Error Interrupt Register (EEI)	32	RW	0000_0000h
18h	Clear Enable Error Interrupt Register (CEEI)	8	WORZ	00h
19h	Set Enable Error Interrupt Register (SEEI)	8	WORZ	00h
1Ah	Clear Enable Request Register (CERQ)	8	WORZ	00h
1Bh	Set Enable Request Register (SERQ)	8	WORZ	00h
1Ch	Clear DONE Status Bit Register (CDNE)	8	WORZ	00h
1Dh	Set START Bit Register (SSRT)	8	WORZ	00h
1Eh	Clear Error Register (CERR)	8	WORZ	00h
1Fh	Clear Interrupt Request Register (CINT)	8	WORZ	00h
24h	Interrupt Request Register (INT)	32	W1C	0000_0000h
2Ch	Error Register (ERR)	32	W1C	0000_0000h
34h	Hardware Request Status Register (HRS)	32	RO	0000_0000h
44h	Enable Asynchronous Request in Stop Register (EARS)	32	RW	0000_0000h
100h	Channel Priority Register (DCHPRI3)	8	RW	03h
101h	Channel Priority Register (DCHPRI2)	8	RW	02h
102h	Channel Priority Register (DCHPRI1)	8	RW	01h
103h	Channel Priority Register (DCHPRI0)	8	RW	00h
104h	Channel Priority Register (DCHPRI7)	8	RW	07h
105h	Channel Priority Register (DCHPRI6)	8	RW	06h
106h	Channel Priority Register (DCHPRI5)	8	RW	05h
107h	Channel Priority Register (DCHPRI4)	8	RW	04h
108h	Channel Priority Register (DCHPRI11)	8	RW	0Bh
109h	Channel Priority Register (DCHPRI10)	8	RW	0Ah
10Ah	Channel Priority Register (DCHPRI9)	8	RW	09h
10Bh	Channel Priority Register (DCHPRI8)	8	RW	08h
10Ch	Channel Priority Register (DCHPRI15)	8	RW	0Fh
10Dh	Channel Priority Register (DCHPRI14)	8	RW	0Eh
10Eh	Channel Priority Register (DCHPRI13)	8	RW	0Dh
10Fh	Channel Priority Register (DCHPRI12)	8	RW	0Ch
1000h - 11E0h	TCD Source Address (TCD0_SADDR - TCD15_SADDR)	32	RW	Table 6-4
1004h - 11E4h	TCD Signed Source Address Offset (TCD0_SOFF - TCD15_SOFF)	16	RW	Table 6-4
1006h - 11E6h	TCD Transfer Attributes (TCD0_ATTR - TCD15_ATTR)	16	RW	Table 6-4
1008h - 11E8h	TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD0_NBYTES_MLNO - TCD15_NBYTES_MLNO)	32	RW	Table 6-4

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
1008h - 11E8h	TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD0_NBYTES_MLOFFNO - TCD15_NBYTES_MLOFFNO)	32	RW	Table 6-4
1008h - 11E8h	TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD0_NBYTES_MLOFFYES - TCD15_NBYTES_MLOFFYES)	32	RW	Table 6-4
100Ch - 11ECh	TCD Last Source Address Adjustment (TCD0_SLAST - TCD15_SLAST)	32	RW	Table 6-4
1010h - 11F0h	TCD Destination Address (TCD0_DADDR - TCD15_DADDR)	32	RW	Table 6-4
1014h - 11F4h	TCD Signed Destination Address Offset (TCD0_DOFF - TCD15_DOFF)	16	RW	Table 6-4
1016h - 11F6h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD0_CITER_ELINKNO - TCD15_CITER_ELINKNO)	16	RW	Table 6-4
1016h - 11F6h	TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD0_CITER_ELINKYES - TCD15_CITER_ELINKYES)	16	RW	Table 6-4
1018h - 11F8h	TCD Last Destination Address Adjustment/Scatter Gather Address (TCD0_DLASTSGA - TCD15_DLASTSGA)	32	RW	Table 6-4
101Ch - 11FCh	TCD Control and Status (TCD0_CSR - TCD15_CSR)	16	RW	Table 6-4
101Eh - 11FEh	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD0_BITER_ELINKNO - TCD15_BITER_ELINKNO)	16	RW	Table 6-4
101Eh - 11FEh	TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD0_BITER_ELINKYES - TCD15_BITER_ELINKYES)	16	RW	Table 6-4

6.4.5.2 Control Register (CR)

6.4.5.2.1 Offset

Register	Offset
CR	0h

6.4.5.2.2 Function

The CR defines the basic operating configuration of the DMA.

Arbitration can be configured to use either a fixed-priority or a round-robin scheme. For fixed-priority arbitration, the highest priority channel requesting service is selected to execute. The channel priority registers assign the priorities; see the DCHPRIn registers. For round-robin arbitration, the channel priorities are ignored and channels are cycled through (from high to low channel number) without regard to priority.

NOTE

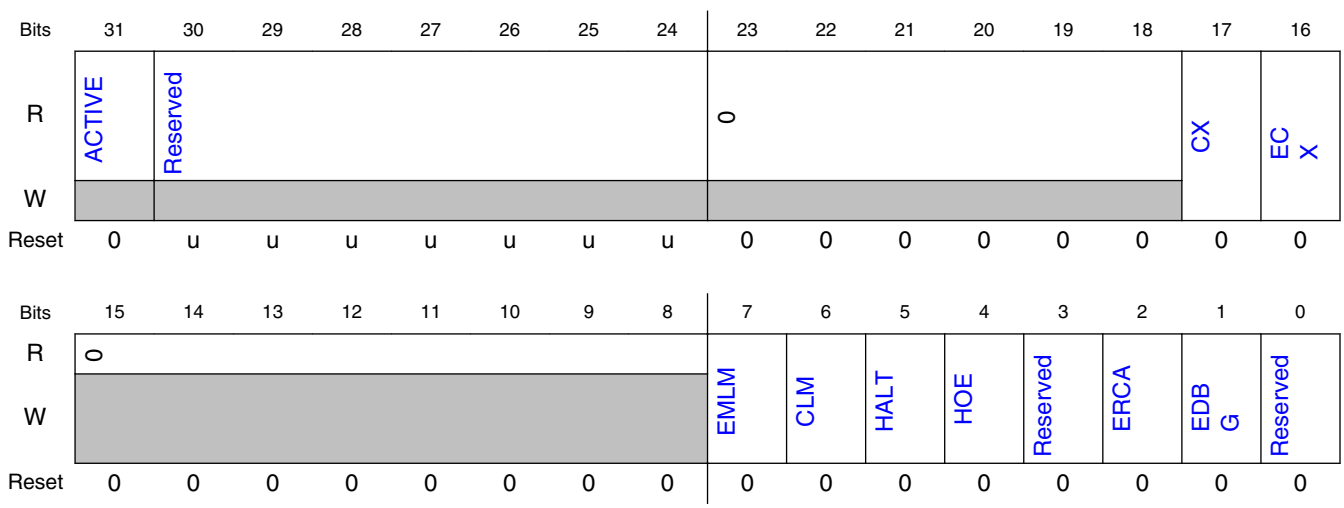
For correct operation, writes to the CR register must be performed only when the DMA channels are inactive; that is, when TCDn_CSR[ACTIVE] bits are cleared.

Minor loop offsets are address offset values added to the final source address (TCDn_SADDR) or destination address (TCDn_DADDR) upon minor loop completion. When minor loop offsets are enabled, the minor loop offset (MLOFF) is added to the final source address (TCDn_SADDR), to the final destination address (TCDn_DADDR), or to both prior to the addresses being written back into the TCD. If the major loop is complete, the minor loop offset is ignored and the major loop address offsets (TCDn_SLAST and TCDn_DLAST_SGA) are used to compute the next TCDn_SADDR and TCDn_DADDR values.

When minor loop mapping is enabled (EMLM is 1), TCDn word2 is redefined. A portion of TCDn word2 is used to specify multiple fields: a source enable bit (SMLOE) to specify the minor loop offset should be applied to the source address (TCDn_SADDR) upon minor loop completion, a destination enable bit (DMLOE) to specify the minor loop offset should be applied to the destination address (TCDn_DADDR) upon minor loop completion, and the sign extended minor loop offset value (MLOFF). The same offset value (MLOFF) is used for both source and destination minor loop offsets. When either minor loop offset is enabled (SMLOE set or DMLOE set), the NBYTES field is reduced to 10 bits. When both minor loop offsets are disabled (SMLOE cleared and DMLOE cleared), the NBYTES field is a 30-bit vector.

When minor loop mapping is disabled (EMLM is 0), all 32 bits of TCDn word2 are assigned to the NBYTES field.

6.4.5.2.3 Diagram



6.4.5.2.4 Fields

Field	Function
31 ACTIVE	DMA Active Status 0b - eDMA is idle. 1b - eDMA is executing a channel.
30-24 —	eDMA version number Reserved
23-18 —	Reserved
17 CX	Cancel Transfer 0b - Normal operation 1b - Cancel the remaining data transfer. Stop the executing channel and force the minor loop to finish. The cancel takes effect after the last write of the current read/write sequence. The CX bit clears itself after the cancel has been honored. This cancel retires the channel normally as if the minor loop was completed.
16 ECX	Error Cancel Transfer 0b - Normal operation 1b - Cancel the remaining data transfer in the same fashion as the CX bit. Stop the executing channel and force the minor loop to finish. The cancel takes effect after the last write of the current read/write sequence. The ECX bit clears itself after the cancel is honored. In addition to cancelling the transfer, ECX treats the cancel as an error condition, thus updating the Error Status register (DMAx_ES) and generating an optional error interrupt.
15-8 —	Reserved
7 EMLM	Enable Minor Loop Mapping 0b - Disabled. TCDn.word2 is defined as a 32-bit NBYTES field. 1b - Enabled. TCDn.word2 is redefined to include individual enable fields, an offset field, and 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 is reduced when either offset is enabled.
6 CLM	Continuous Link Mode **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 - A minor loop channel link made to itself goes through channel arbitration before being activated again. 1b - A minor loop channel link made to itself does not go through channel arbitration before being activated again. Upon minor loop completion, 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.
5 HALT	Halt DMA Operations 0b - Normal operation 1b - Stall the start of any new channels. Executing channels are allowed to complete. Channel execution resumes when this bit is cleared.
4 HOE	Halt On Error 0b - Normal operation 1b - Any error causes the HALT bit to set. Subsequently, all service requests are ignored until the HALT bit is cleared.

Table continues on the next page...

Field	Function
3 —	Reserved Reserved
2 ERCA	Enable Round Robin Channel Arbitration 0b - Fixed priority arbitration is used for channel selection . 1b - Round robin arbitration is used for channel selection .
1 EDBG	Enable Debug 0b - When in debug mode, the DMA continues to operate. 1b - When in debug mode, the DMA stalls the start of a new channel. Executing channels are allowed to complete. Channel execution resumes when the system exits debug mode or the EDBG bit is cleared.
0 —	Reserved Reserved

6.4.5.3 Error Status Register (ES)

6.4.5.3.1 Offset

Register	Offset
ES	4h

6.4.5.3.2 Function

The ES provides information concerning the last recorded channel error. Channel errors can be caused by:

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

See Fault reporting and handling for more details.

6.4.5.3.3 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	0	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

6.4.5.3.4 Fields

Field	Function
31 VLD	VLD Logical OR of all ERR status bits 0b - No ERR bits are set. 1b - At least one ERR bit is set indicating a valid error exists that has not been cleared.
30-17 —	Reserved
16 ECX	Transfer Canceled 0b - No canceled transfers 1b - The last recorded entry was a canceled transfer by the error cancel transfer input
15 —	Reserved
14 CPE	Channel Priority Error 0b - No channel priority error 1b - The last recorded error was a configuration error in the channel priorities . Channel priorities are not unique.
13-12 —	Reserved
11-8 ERRCHN	Error Channel Number or Canceled Channel Number The channel number of the last recorded error, excluding CPE errors, or last recorded error canceled transfer.
7 SAE	Source Address Error 0b - No source address configuration error. 1b - The last 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 last 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

Table continues on the next page...

Field	Function
	1b - The last 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 last 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 last 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], orTCDn_CITER[CITER] is equal to zero, orTCDn_CITER[ELINK] is not equal to TCDn_BITER[ELINK]
2 SGE	Scatter/Gather Configuration Error 0b - No scatter/gather configuration error 1b - The last recorded error was a configuration error detected in the TCDn_DLASTSGA field. This field is checked 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.
1 SBE	Source Bus Error 0b - No source bus error 1b - The last recorded error was a bus error on a source read
0 DBE	Destination Bus Error 0b - No destination bus error 1b - The last recorded error was a bus error on a destination write

6.4.5.4 Enable Request Register (ERQ)

6.4.5.4.1 Offset

Register	Offset
ERQ	Ch

6.4.5.4.2 Function

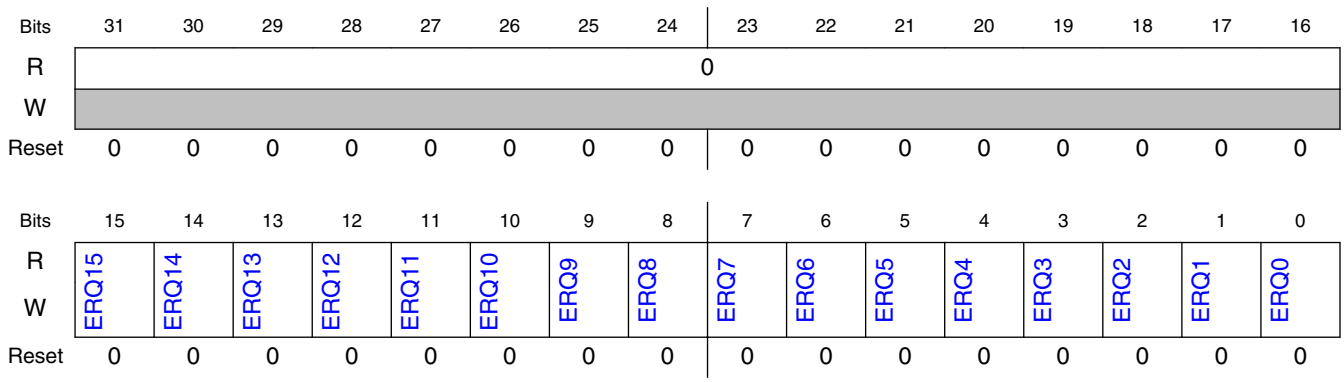
The ERQ register provides a bit map for the 16 channels to enable the request signal for each channel. The state of any given channel enable is directly affected by writes to this register; it is also affected by writes to the SERQ and CERQ registers. These registers are provided so the request enable 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 flag must be asserted before a channel's hardware service request is accepted. The state of the DMA enable request flag 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 clearing the channel's ERQ bit.

6.4.5.4.3 Diagram



6.4.5.4.4 Fields

Field	Function
31-16 —	Reserved
15 ERQ15	Enable DMA Request 15 0b - The DMA request signal for the corresponding channel is disabled 1b - The DMA request signal for the corresponding channel is enabled
14 ERQ14	Enable DMA Request 14 0b - The DMA request signal for the corresponding channel is disabled 1b - The DMA request signal for the corresponding channel is enabled
13 ERQ13	Enable DMA Request 13 0b - The DMA request signal for the corresponding channel is disabled 1b - The DMA request signal for the corresponding channel is enabled
12 ERQ12	Enable DMA Request 12 0b - The DMA request signal for the corresponding channel is disabled 1b - The DMA request signal for the corresponding channel is enabled
11 ERQ11	Enable DMA Request 11 0b - The DMA request signal for the corresponding channel is disabled 1b - The DMA request signal for the corresponding channel is enabled
10 ERQ10	Enable DMA Request 10 0b - The DMA request signal for the corresponding channel is disabled 1b - The DMA request signal for the corresponding channel is enabled
9 ERQ9	Enable DMA Request 9 0b - The DMA request signal for the corresponding channel is disabled 1b - The DMA request signal for the corresponding channel is enabled
8 ERQ8	Enable DMA Request 8 0b - The DMA request signal for the corresponding channel is disabled 1b - The DMA request signal for the corresponding channel is enabled

Table continues on the next page...

Field	Function
7 ERQ7	Enable DMA Request 7 0b - The DMA request signal for the corresponding channel is disabled 1b - The DMA request signal for the corresponding channel is enabled
6 ERQ6	Enable DMA Request 6 0b - The DMA request signal for the corresponding channel is disabled 1b - The DMA request signal for the corresponding channel is enabled
5 ERQ5	Enable DMA Request 5 0b - The DMA request signal for the corresponding channel is disabled 1b - The DMA request signal for the corresponding channel is enabled
4 ERQ4	Enable DMA Request 4 0b - The DMA request signal for the corresponding channel is disabled 1b - The DMA request signal for the corresponding channel is enabled
3 ERQ3	Enable DMA Request 3 0b - The DMA request signal for the corresponding channel is disabled 1b - The DMA request signal for the corresponding channel is enabled
2 ERQ2	Enable DMA Request 2 0b - The DMA request signal for the corresponding channel is disabled 1b - The DMA request signal for the corresponding channel is enabled
1 ERQ1	Enable DMA Request 1 0b - The DMA request signal for the corresponding channel is disabled 1b - The DMA request signal for the corresponding channel is enabled
0 ERQ0	Enable DMA Request 0 0b - The DMA request signal for the corresponding channel is disabled 1b - The DMA request signal for the corresponding channel is enabled

6.4.5.5 Enable Error Interrupt Register (EEI)

6.4.5.5.1 Offset

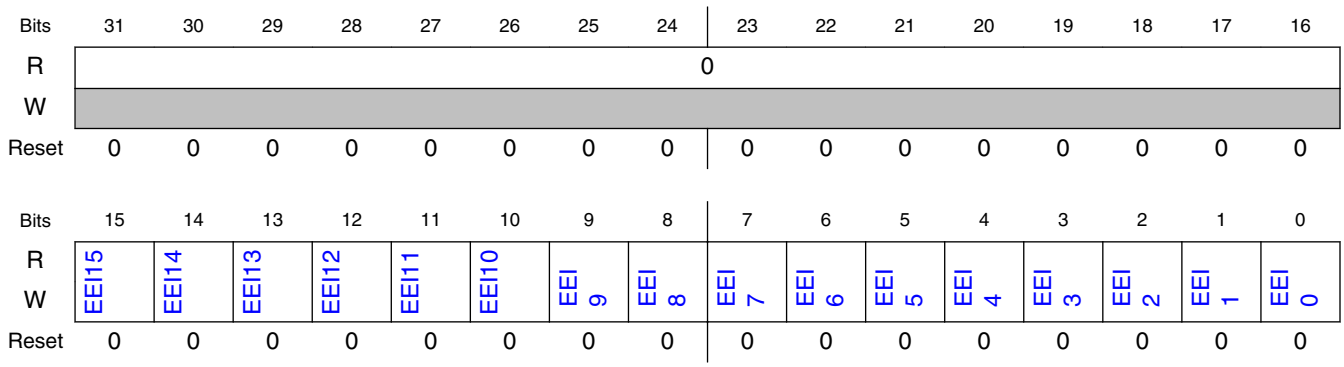
Register	Offset
EEI	14h

6.4.5.5.2 Function

The EEI register provides a bit map for the 16 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. 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 flag must be asserted before an error interrupt request for a given channel is asserted to the interrupt controller.

6.4.5.5.3 Diagram



6.4.5.5.4 Fields

Field	Function
31-16 —	Reserved
15 EEI15	Enable Error Interrupt 15 0b - The error signal for corresponding channel does not generate an error interrupt 1b - The assertion of the error signal for corresponding channel generates an error interrupt request
14 EEI14	Enable Error Interrupt 14 0b - The error signal for corresponding channel does not generate an error interrupt 1b - The assertion of the error signal for corresponding channel generates an error interrupt request
13 EEI13	Enable Error Interrupt 13 0b - The error signal for corresponding channel does not generate an error interrupt 1b - The assertion of the error signal for corresponding channel generates an error interrupt request
12 EEI12	Enable Error Interrupt 12 0b - The error signal for corresponding channel does not generate an error interrupt 1b - The assertion of the error signal for corresponding channel generates an error interrupt request
11 EEI11	Enable Error Interrupt 11 0b - The error signal for corresponding channel does not generate an error interrupt 1b - The assertion of the error signal for corresponding channel generates an error interrupt request
10 EEI10	Enable Error Interrupt 10 0b - The error signal for corresponding channel does not generate an error interrupt 1b - The assertion of the error signal for corresponding channel generates an error interrupt request
9 EEI9	Enable Error Interrupt 9 0b - The error signal for corresponding channel does not generate an error interrupt 1b - The assertion of the error signal for corresponding channel generates an error interrupt request
8 EEI8	Enable Error Interrupt 8 0b - The error signal for corresponding channel does not generate an error interrupt 1b - The assertion of the error signal for corresponding channel generates an error interrupt request
7	Enable Error Interrupt 7

Table continues on the next page...

Field	Function
EEI7	0b - The error signal for corresponding channel does not generate an error interrupt 1b - The assertion of the error signal for corresponding channel generates an error interrupt request
6 EEI6	Enable Error Interrupt 6 0b - The error signal for corresponding channel does not generate an error interrupt 1b - The assertion of the error signal for corresponding channel generates an error interrupt request
5 EEI5	Enable Error Interrupt 5 0b - The error signal for corresponding channel does not generate an error interrupt 1b - The assertion of the error signal for corresponding channel generates an error interrupt request
4 EEI4	Enable Error Interrupt 4 0b - The error signal for corresponding channel does not generate an error interrupt 1b - The assertion of the error signal for corresponding channel generates an error interrupt request
3 EEI3	Enable Error Interrupt 3 0b - The error signal for corresponding channel does not generate an error interrupt 1b - The assertion of the error signal for corresponding channel generates an error interrupt request
2 EEI2	Enable Error Interrupt 2 0b - The error signal for corresponding channel does not generate an error interrupt 1b - The assertion of the error signal for corresponding channel generates an error interrupt request
1 EEI1	Enable Error Interrupt 1 0b - The error signal for corresponding channel does not generate an error interrupt 1b - The assertion of the error signal for corresponding channel generates an error interrupt request
0 EEI0	Enable Error Interrupt 0 0b - The error signal for corresponding channel does not generate an error interrupt 1b - The assertion of the error signal for corresponding channel generates an error interrupt request

6.4.5.6 Clear Enable Error Interrupt Register (CEEI)

6.4.5.6.1 Offset

Register	Offset
CEEI	18h

6.4.5.6.2 Function

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

If the NOP bit is set, the command is ignored. This allows you to set a single, byte-wide register with a 32-bit write while not affecting the other registers addressed in the write. In such a case the other three bytes of the word would all have their NOP bit set so that that these register will not be affected by the write.

Reads of this register return all zeroes.

6.4.5.6.3 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

6.4.5.6.4 Fields

Field	Function
7 NOP	No Op enable 0b - Normal operation 1b - No operation, ignore the other bits in this register
6 CAEE	Clear All Enable Error Interrupts 0b - Clear only the EEI bit specified in the CEEI field 1b - Clear all bits in EEI
5-4 —	Reserved
3-0 CEEI	Clear Enable Error Interrupt Clears the corresponding bit in EEI

6.4.5.7 Set Enable Error Interrupt Register (SEEI)

6.4.5.7.1 Offset

Register	Offset
SEEI	19h

6.4.5.7.2 Function

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

If the NOP bit is set, the command is ignored. This allows you to set a single, byte-wide register with a 32-bit write while not affecting the other registers addressed in the write. In such a case the other three bytes of the word would all have their NOP bit set so that that these register will not be affected by the write.

Reads of this register return all zeroes.

6.4.5.7.3 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

6.4.5.7.4 Fields

Field	Function
7 NOP	No Op enable 0b - Normal operation 1b - No operation, ignore the other bits in this register
6 SAEE	Sets All Enable Error Interrupts 0b - Set only the EEI bit specified in the SEEI field. 1b - Sets all bits in EEI
5-4 —	Reserved
3-0 SEEI	Set Enable Error Interrupt Sets the corresponding bit in EEI

6.4.5.8 Clear Enable Request Register (CERQ)

6.4.5.8.1 Offset

Register	Offset
CERQ	1Ah

6.4.5.8.2 Function

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

If the NOP bit is set, the command is ignored. This allows you to set a single, byte-wide register with a 32-bit write while not affecting the other registers addressed in the write. In such a case the other three bytes of the word would all have their NOP bit set so that that these register will not be affected by the write.

Reads of this register return all zeroes.

NOTE

Disable a channel's hardware service request at the source before clearing the channel's ERQ bit.

6.4.5.8.3 Diagram

Bits	7	6	5	4	3	2	1	0
R	0	0			0			
W	NOP	CAER	0		CERQ			
Reset	0	0	0	0	0	0	0	0

6.4.5.8.4 Fields

Field	Function
7 NOP	No Op enable 0b - Normal operation 1b - No operation, ignore the other bits in this register
6 CAER	Clear All Enable Requests 0b - Clear only the ERQ bit specified in the CERQ field 1b - Clear all bits in ERQ
5-4 —	Reserved
3-0 CERQ	Clear Enable Request Clears the corresponding bit in ERQ.

6.4.5.9 Set Enable Request Register (SERQ)

6.4.5.9.1 Offset

Register	Offset
SERQ	1Bh

6.4.5.9.2 Function

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

If the NOP bit is set, the command is ignored. This allows you to set a single, byte-wide register with a 32-bit write while not affecting the other registers addressed in the write. In such a case the other three bytes of the word would all have their NOP bit set so that that these register will not be affected by the write.

Reads of this register return all zeroes.

6.4.5.9.3 Diagram

Bits	7	6	5	4	3	2	1	0
R	0	0			0			
W	NOP	SAER	0		SERQ			
Reset	0	0	0	0	0	0	0	0

6.4.5.9.4 Fields

Field	Function
7 NOP	No Op enable 0b - Normal operation 1b - No operation, ignore the other bits in this register
6 SAER	Set All Enable Requests 0b - Set only the ERQ bit specified in the SERQ field 1b - Set all bits in ERQ
5-4 —	Reserved

Table continues on the next page...

Field	Function
3-0 SERQ	Set Enable Request Sets the corresponding bit in ERQ.

6.4.5.10 Clear DONE Status Bit Register (CDNE)

6.4.5.10.1 Offset

Register	Offset
CDNE	1Ch

6.4.5.10.2 Function

The CDNE provides a simple memory-mapped mechanism to clear the DONE bit in the TCD of the given channel. The data value on a register write causes the DONE bit in the corresponding transfer control descriptor to be cleared. Setting the CADN bit provides a global clear function, forcing all DONE bits to be cleared.

If the NOP bit is set, the command is ignored. This allows you to set a single, byte-wide register with a 32-bit write while not affecting the other registers addressed in the write. In such a case the other three bytes of the word would all have their NOP bit set so that that these register will not be affected by the write.

Reads of this register return all zeroes.

6.4.5.10.3 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

6.4.5.10.4 Fields

Field	Function
7	No Op enable

Table continues on the next page...

Field	Function
NOP	0b - Normal operation 1b - No operation, ignore the other bits in this register
6 CADN	Clears All DONE Bits 0b - Clears only the TCDn_CSR[DONE] bit specified in the CDNE field 1b - Clears all bits in TCDn_CSR[DONE]
5-4 —	Reserved
3-0 CDNE	Clear DONE Bit Clears the corresponding bit in TCDn_CSR[DONE]

6.4.5.11 Set START Bit Register (SSRT)

6.4.5.11.1 Offset

Register	Offset
SSRT	1Dh

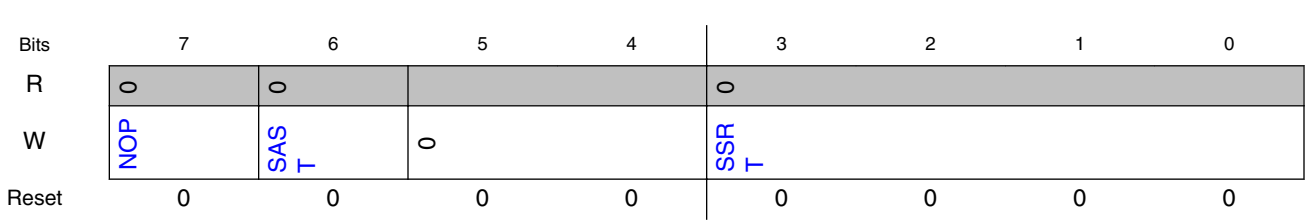
6.4.5.11.2 Function

The SSRT provides a simple memory-mapped mechanism to set the START bit in the TCD of the given channel. The data value on a register write causes the START bit in the corresponding transfer control descriptor to be set. Setting the SAST bit provides a global set function, forcing all START bits to be set.

If the NOP bit is set, the command is ignored. This allows you to set a single, byte-wide register with a 32-bit write while not affecting the other registers addressed in the write. In such a case the other three bytes of the word would all have their NOP bit set so that that these register will not be affected by the write.

Reads of this register return all zeroes.

6.4.5.11.3 Diagram



6.4.5.11.4 Fields

Field	Function
7 NOP	No Op enable 0b - Normal operation 1b - No operation, ignore the other bits in this register
6 SAST	Set All START Bits (activates all channels) 0b - Set only the TCDn_CSR[START] bit specified in the SSRT field 1b - Set all bits in TCDn_CSR[START]
5-4 —	Reserved
3-0 SSRT	Set START Bit Sets the corresponding bit in TCDn_CSR[START]

6.4.5.12 Clear Error Register (CERR)

6.4.5.12.1 Offset

Register	Offset
CERR	1Eh

6.4.5.12.2 Function

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

6.4.5.12.3 Diagram

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

6.4.5.12.4 Fields

Field	Function
7 NOP	No Op enable 0b - Normal operation 1b - No operation, ignore the other bits in this register
6 CAEI	Clear All Error Indicators 0b - Clear only the ERR bit specified in the CERR field 1b - Clear all bits in ERR
5-4 —	Reserved
3-0 CERR	Clear Error Indicator Clears the corresponding bit in ERR

6.4.5.13 Clear Interrupt Request Register (CINT)

6.4.5.13.1 Offset

Register	Offset
CINT	1Fh

6.4.5.13.2 Function

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

If the NOP bit is set, the command is ignored. This allows you to set a single, byte-wide register with a 32-bit write while not affecting the other registers addressed in the write. In such a case the other three bytes of the word would all have their NOP bit set so that that these register will not be affected by the write.

Reads of this register return all zeroes.

6.4.5.13.3 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

6.4.5.13.4 Fields

Field	Function
7 NOP	No Op enable 0b - Normal operation 1b - No operation, ignore the other bits in this register
6 CAIR	Clear All Interrupt Requests 0b - Clear only the INT bit specified in the CINT field 1b - Clear all bits in INT
5-4 —	Reserved
3-0 CINT	Clear Interrupt Request Clears the corresponding bit in INT

6.4.5.14 Interrupt Request Register (INT)

6.4.5.14.1 Offset

Register	Offset
INT	24h

6.4.5.14.2 Function

The INT register provides a bit map for the 16 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 clear 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 zero in any bit position has no affect 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.

6.4.5.14.3 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	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

6.4.5.14.4 Fields

Field	Function
31-16 —	Reserved
15 INT15	Interrupt Request 15 0b - The interrupt request for corresponding channel is cleared 1b - The interrupt request for corresponding channel is active
14 INT14	Interrupt Request 14 0b - The interrupt request for corresponding channel is cleared 1b - The interrupt request for corresponding channel is active
13 INT13	Interrupt Request 13 0b - The interrupt request for corresponding channel is cleared 1b - The interrupt request for corresponding channel is active
12 INT12	Interrupt Request 12 0b - The interrupt request for corresponding channel is cleared 1b - The interrupt request for corresponding channel is active
11 INT11	Interrupt Request 11 0b - The interrupt request for corresponding channel is cleared 1b - The interrupt request for corresponding channel is active
10 INT10	Interrupt Request 10 0b - The interrupt request for corresponding channel is cleared 1b - The interrupt request for corresponding channel is active

Table continues on the next page...

Field	Function
9 INT9	Interrupt Request 9 0b - The interrupt request for corresponding channel is cleared 1b - The interrupt request for corresponding channel is active
8 INT8	Interrupt Request 8 0b - The interrupt request for corresponding channel is cleared 1b - The interrupt request for corresponding channel is active
7 INT7	Interrupt Request 7 0b - The interrupt request for corresponding channel is cleared 1b - The interrupt request for corresponding channel is active
6 INT6	Interrupt Request 6 0b - The interrupt request for corresponding channel is cleared 1b - The interrupt request for corresponding channel is active
5 INT5	Interrupt Request 5 0b - The interrupt request for corresponding channel is cleared 1b - The interrupt request for corresponding channel is active
4 INT4	Interrupt Request 4 0b - The interrupt request for corresponding channel is cleared 1b - The interrupt request for corresponding channel is active
3 INT3	Interrupt Request 3 0b - The interrupt request for corresponding channel is cleared 1b - The interrupt request for corresponding channel is active
2 INT2	Interrupt Request 2 0b - The interrupt request for corresponding channel is cleared 1b - The interrupt request for corresponding channel is active
1 INT1	Interrupt Request 1 0b - The interrupt request for corresponding channel is cleared 1b - The interrupt request for corresponding channel is active
0 INT0	Interrupt Request 0 0b - The interrupt request for corresponding channel is cleared 1b - The interrupt request for corresponding channel is active

6.4.5.15 Error Register (ERR)

6.4.5.15.1 Offset

Register	Offset
ERR	2Ch

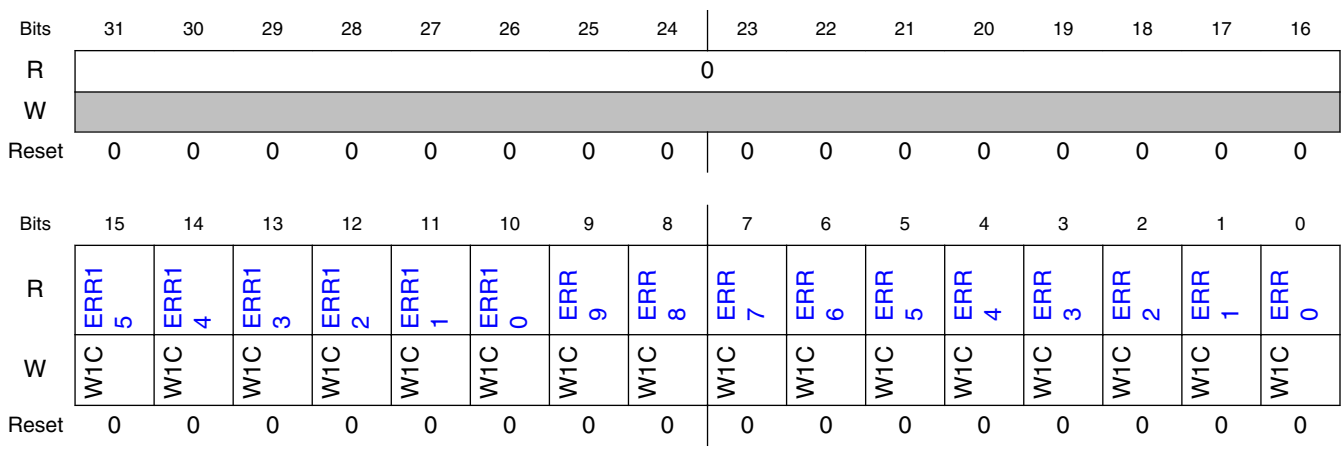
6.4.5.15.2 Function

The ERR register provides a bit map for the 16 channels, signaling the presence of an error for each channel. The eDMA engine signals the occurrence of an error condition by setting the appropriate bit in this register. The outputs of this register are enabled by the

contents of the EEI register, and 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 clear the appropriate bit, 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 transfer control descriptor DONE flag and the possible assertion 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 one in any bit position clears the corresponding channel's error status. A zero in any bit position has no affect on the corresponding channel's current error status. The CERR is provided so the error indicator for a single channel can easily be cleared.

6.4.5.15.3 Diagram



6.4.5.15.4 Fields

Field	Function
31-16 —	Reserved
15 ERR15	Error In Channel 15 0b - An error in this channel has not occurred 1b - An error in this channel has occurred
14 ERR14	Error In Channel 14 0b - An error in this channel has not occurred 1b - An error in this channel has occurred
13	Error In Channel 13

Table continues on the next page...

Field	Function
ERR13	0b - An error in this channel has not occurred 1b - An error in this channel has occurred
12 ERR12	Error In Channel 12 0b - An error in this channel has not occurred 1b - An error in this channel has occurred
11 ERR11	Error In Channel 11 0b - An error in this channel has not occurred 1b - An error in this channel has occurred
10 ERR10	Error In Channel 10 0b - An error in this channel has not occurred 1b - An error in this channel has occurred
9 ERR9	Error In Channel 9 0b - An error in this channel has not occurred 1b - An error in this channel has occurred
8 ERR8	Error In Channel 8 0b - An error in this channel has not occurred 1b - An error in this channel has occurred
7 ERR7	Error In Channel 7 0b - An error in this channel has not occurred 1b - An error in this channel has occurred
6 ERR6	Error In Channel 6 0b - An error in this channel has not occurred 1b - An error in this channel has occurred
5 ERR5	Error In Channel 5 0b - An error in this channel has not occurred 1b - An error in this channel has occurred
4 ERR4	Error In Channel 4 0b - An error in this channel has not occurred 1b - An error in this channel has occurred
3 ERR3	Error In Channel 3 0b - An error in this channel has not occurred 1b - An error in this channel has occurred
2 ERR2	Error In Channel 2 0b - An error in this channel has not occurred 1b - An error in this channel has occurred
1 ERR1	Error In Channel 1 0b - An error in this channel has not occurred 1b - An error in this channel has occurred
0 ERR0	Error In Channel 0 0b - An error in this channel has not occurred 1b - An error in this channel has occurred

6.4.5.16 Hardware Request Status Register (HRS)

6.4.5.16.1 Offset

Register	Offset
HRS	34h

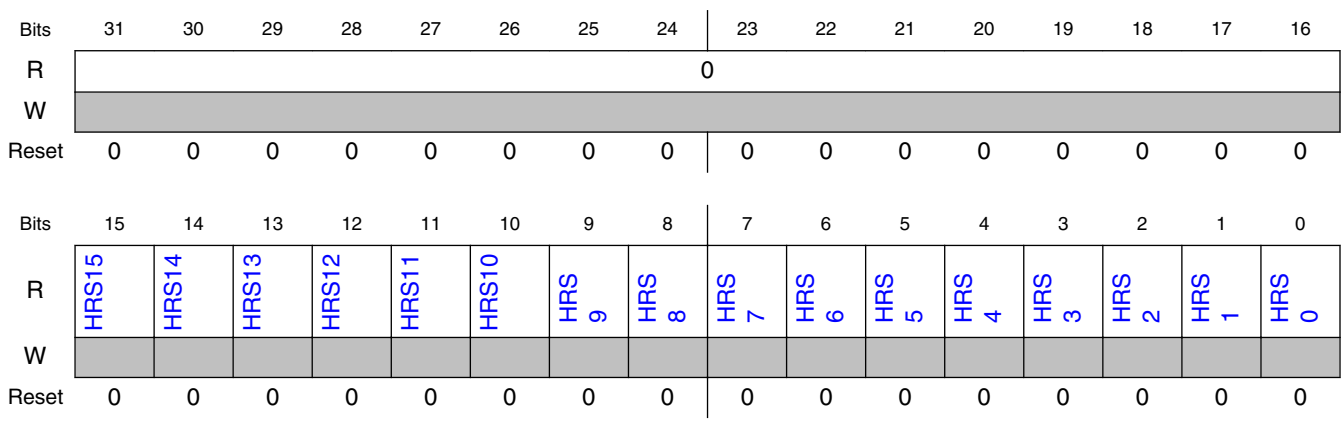
6.4.5.16.2 Function

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.

6.4.5.16.3 Diagram



6.4.5.16.4 Fields

Field	Function
31-16 —	Reserved
15 HRS15	Hardware Request Status Channel 15 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware.

Table continues on the next page...

Field	Function
	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 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 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 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 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 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 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 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 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 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 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 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 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 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 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 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0b - A hardware service request for channel 7 is not present 1b - A hardware service request for channel 7 is present

Table continues on the next page...

Field	Function
6 HRS6	Hardware Request Status Channel 6 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 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 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 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 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 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 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 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 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 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 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 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 The HRS bit for its respective channel remains asserted for the period when a Hardware Request is Present on the Channel. After the Request is completed and Channel is free, the HRS bit is automatically cleared by hardware. 0b - A hardware service request for channel 0 is not present 1b - A hardware service request for channel 0 is present

6.4.5.17 Enable Asynchronous Request in Stop Register (EARS)

6.4.5.17.1 Offset

Register	Offset
EARS	44h

6.4.5.17.2 Function

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

6.4.5.17.3 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	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
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

6.4.5.17.4 Fields

Field	Function
31-16 —	Reserved
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	Enable asynchronous DMA request in stop mode for channel 11 0b - Disable asynchronous DMA request for channel 11.

Table continues on the next page...

Field	Function
EDREQ_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 EDREQ_5	Enable asynchronous DMA request in stop mode for channel 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.

6.4.5.18 Channel Priority Register (DCHPRI0 - DCHPRI15)

6.4.5.18.1 Offset

Register	Offset
DCHPRI3	100h
DCHPRI2	101h
DCHPRI1	102h
DCHPRI0	103h

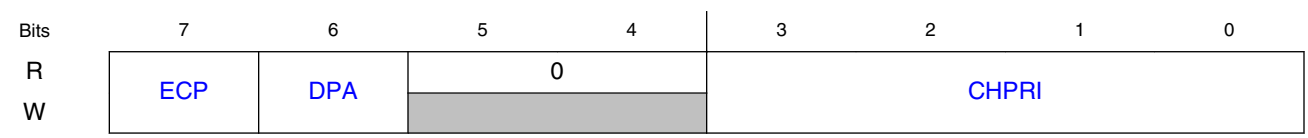
Table continues on the next page...

Register	Offset
DCHPRI7	104h
DCHPRI6	105h
DCHPRI5	106h
DCHPRI4	107h
DCHPRI11	108h
DCHPRI10	109h
DCHPRI9	10Ah
DCHPRI8	10Bh
DCHPRI15	10Ch
DCHPRI14	10Dh
DCHPRI13	10Eh
DCHPRI12	10Fh

6.4.5.18.2 Function

When fixed-priority channel arbitration is enabled (CR[ERCA] = 0), the contents of these registers define the unique priorities associated with each channel. The channel priorities are evaluated by numeric value; for example, 0 is the lowest priority, 1 is the next higher priority, then 2, 3, etc. 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.

6.4.5.18.3 Diagram



See Register reset values.

6.4.5.18.4 Register reset values

Register	Reset value
DCHPRI0	00h
DCHPRI1	01h
DCHPRI2	02h
DCHPRI3	03h
DCHPRI4	04h

Table continues on the next page...

Register	Reset value
DCHPRI5	05h
DCHPRI6	06h
DCHPRI7	07h
DCHPRI8	08h
DCHPRI9	09h
DCHPRI10	0Ah
DCHPRI11	0Bh
DCHPRI12	0Ch
DCHPRI13	0Dh
DCHPRI14	0Eh
DCHPRI15	0Fh

6.4.5.18.5 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 —	Reserved
3-0 CHPRI	Channel n Arbitration Priority Channel priority when fixed-priority arbitration is enabled

6.4.5.19 TCD Source Address (TCD0_SADDR - TCD15_SADDR)

6.4.5.19.1 Offset

For n = 0 to 15:

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

6.4.5.19.2 Function

This register contains the source address of the transfer.

6.4.5.19.3 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

6.4.5.19.4 Fields

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

6.4.5.20 TCD Signed Source Address Offset (TCD0_SOFF - TCD15_SOFF)

6.4.5.20.1 Offset

For n = 0 to 15:

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

6.4.5.20.2 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

6.4.5.20.3 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.

6.4.5.21 TCD Transfer Attributes (TCD0_ATTR - TCD15_ATTR)

6.4.5.21.1 Offset

For n = 0 to 15:

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

6.4.5.21.2 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

6.4.5.21.3 Fields

Field	Function
15-11 SMOD	Source Address Modulo 00000b - Source address modulo feature is disabled 00001-11111b - This value 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.
10-8 SSIZE	Source data transfer size **NOTE:** Using a Reserved value causes a configuration error.

Table continues on the next page...

Field	Function
	NOTE: The eDMA defaults to privileged data access for all transactions. 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

6.4.5.22 TCD Minor Byte Count (Minor Loop Mapping Disabled) (TCD0_NBYTES_MLNO - TCD15_NBYTES_MLNO)

6.4.5.22.1 Offset

For n = 0 to 15:

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

6.4.5.22.2 Function

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

6.4.5.22.3 Diagram

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

6.4.5.22.4 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 perform 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, 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.

6.4.5.23 TCD Signed Minor Loop Offset (Minor Loop Mapping Enabled and Offset Disabled) (TCD0_NBYTES_MLOFFNO - TCD15_NBYTES_MLOFFNO)

6.4.5.23.1 Offset

For n = 0 to 15:

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

6.4.5.23.2 Function

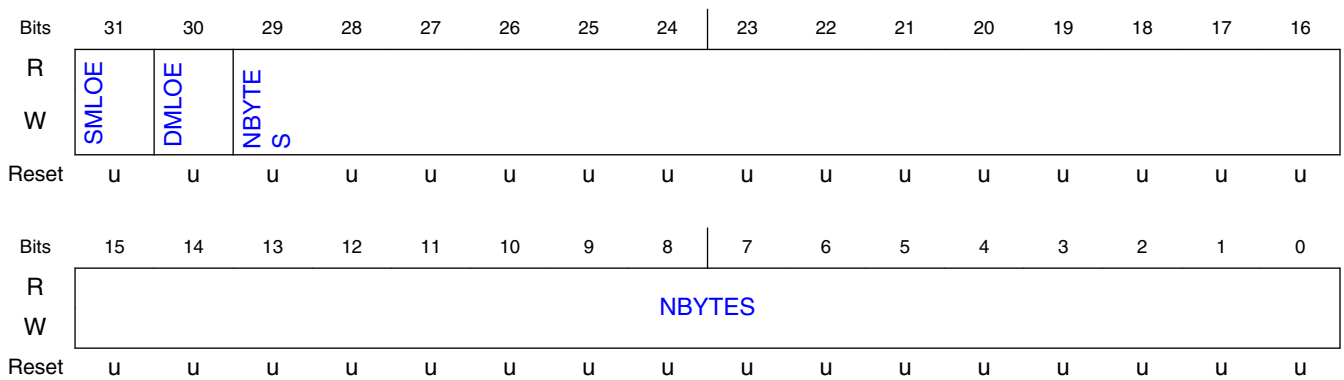
One of three registers (this register, TCD_NBYTES_MLNO, or TCD_NBYTES_MLOFFYES), defines the number of bytes to transfer per request. Which register to use depends on whether minor loop mapping is disabled, enabled but not used for this channel, or enabled and used.

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 or DMLOE is set, then refer to the TCD_NBYTES_MLOFFYES register description. If minor loop mapping is disabled, then refer to the TCD_NBYTES_MLNO register description.

6.4.5.23.3 Diagram



6.4.5.23.4 Fields

Field	Function
31 SMLOE	Source Minor Loop Offset Enable Selects whether the minor loop offset is applied to the source address upon minor loop completion. 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 Selects whether the minor loop offset is applied to the destination address upon minor loop completion. 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 perform 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, the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed.

6.4.5.24 TCD Signed Minor Loop Offset (Minor Loop Mapping and Offset Enabled) (TCD0_NBYTES_MLOFFYES - TCD15_NBYTES_MLOFFYES)

6.4.5.24.1 Offset

For n = 0 to 15:

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

6.4.5.24.2 Function

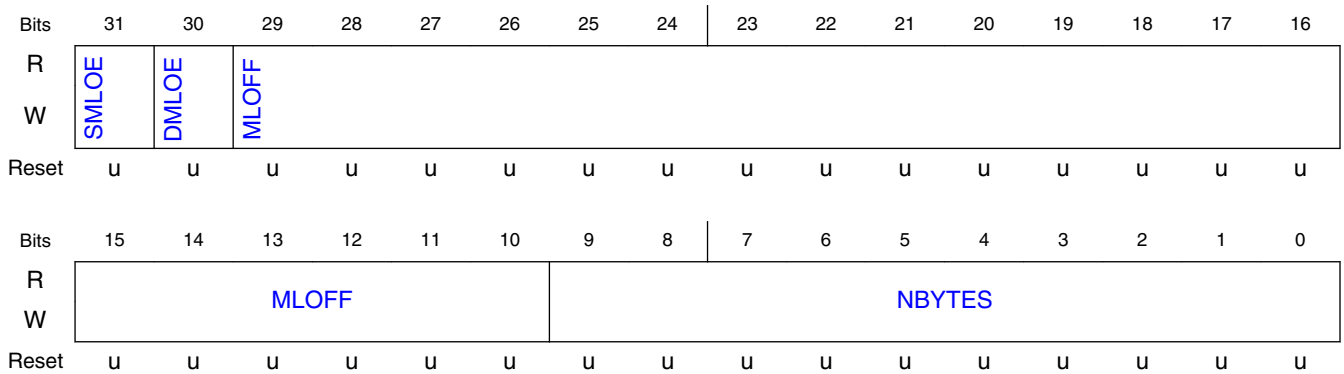
One of three registers (this register, TCD_NBYTES_MLNO, or TCD_NBYTES_MLOFFNO), defines the number of bytes to transfer per request. Which register to use depends on whether minor loop mapping is disabled, enabled but not used for this channel, or enabled and used.

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 and DMLOE are cleared, then refer to the TCD_NBYTES_MLOFFNO register description. If minor loop mapping is disabled, then refer to the TCD_NBYTES_MLNO register description.

6.4.5.24.3 Diagram



6.4.5.24.4 Fields

Field	Function
31 SMLOE	Source Minor Loop Offset Enable Selects whether the minor loop offset is applied to the source address upon minor loop completion. 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 Selects whether the minor loop offset is applied to the destination address upon minor loop completion. 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 or DMLOE is set, 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 perform 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, the major iteration count is decremented and restored to the TCD memory. If the major iteration count is completed, additional processing is performed.

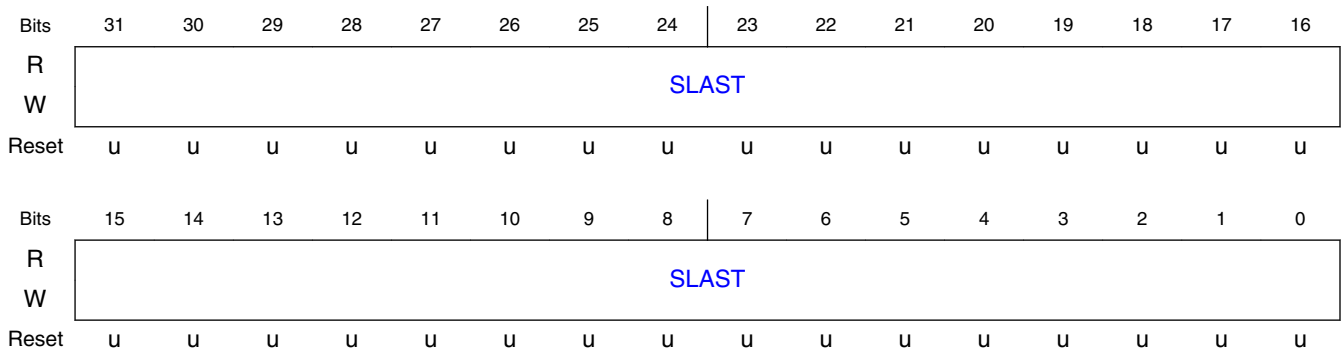
6.4.5.25 TCD Last Source Address Adjustment (TCD0_SLAST - TCD15_SLAST)

6.4.5.25.1 Offset

For n = 0 to 15:

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

6.4.5.25.2 Diagram



6.4.5.25.3 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.

6.4.5.26 TCD Destination Address (TCD0_DADDR - TCD15_DADDR)

6.4.5.26.1 Offset

For n = 0 to 15:

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

6.4.5.26.2 Function

This register contains the destination address of the transfer.

6.4.5.26.3 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

6.4.5.26.4 Fields

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

6.4.5.27 TCD Signed Destination Address Offset (TCD0_DOFF - TCD15_DOFF)

6.4.5.27.1 Offset

For n = 0 to 15:

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

6.4.5.27.2 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

6.4.5.27.3 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.

6.4.5.28 TCD Current Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD0_CITER_ELINKNO - TCD15_CITER_ELINKNO)

6.4.5.28.1 Offset

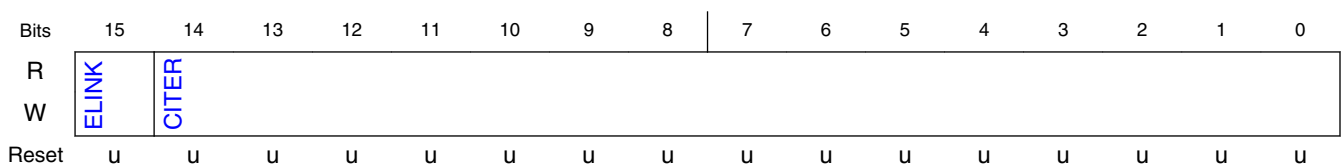
For n = 0 to 15:

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

6.4.5.28.2 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 - TCD15_CITER_ELINKYES), but its fields are defined differently based on the state of the ELINK field. If the ELINK field is cleared, this register is defined as follows.

6.4.5.28.3 Diagram



6.4.5.28.4 Fields

Field	Function
15	Enable channel-to-channel linking on minor-loop complete

Table continues on the next page...

Field	Function
ELINK	As the channel completes the minor loop, this flag 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 the TCDn_CSR[START] bit 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 bit must be equal to the BITER[ELINK] bit; otherwise, a configuration error is reported. 0b - The channel-to-channel linking is disabled 1b - The 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, optionally generating an interrupt to signal channel completion before reloading the CITER field from the Beginning Iteration Count (BITER) field. **NOTE:** When the CITER field is initially loaded by software, it must be set to the same value as that contained in the BITER field. **NOTE:** If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

6.4.5.29 TCD Current Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD0_CITER_ELINKYES - TCD15_CITER_ELINKYES)

6.4.5.29.1 Offset

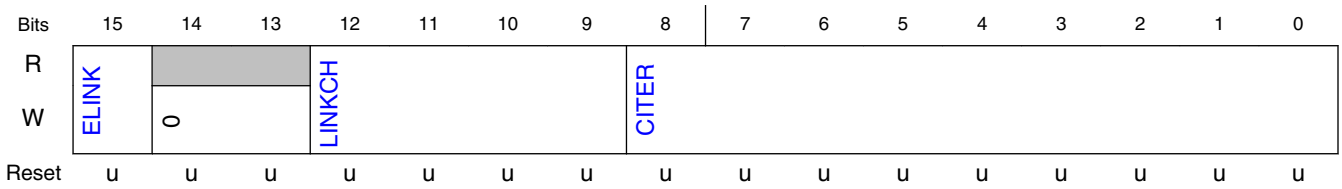
For n = 0 to 15:

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

6.4.5.29.2 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 - TCD15_CITER_ELINKNO), but its fields are defined differently based on the state of the ELINK field. If the ELINK field is set, this register is defined as follows.

6.4.5.29.3 Diagram



6.4.5.29.4 Fields

Field	Function
15 ELINK	Enable channel-to-channel linking on minor-loop complete As the channel completes the minor loop, this flag 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 the TCDn_CSR[START] bit 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 bit must be equal to the BITER[ELINK] bit; otherwise, a configuration error is reported. 0b - The channel-to-channel linking is disabled 1b - The channel-to-channel linking is enabled
14-13 —	Reserved
12-9 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] bit.
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, optionally generating an interrupt to signal channel completion before reloading the CITER field from the Beginning Iteration Count (BITER) field. **NOTE:** When the CITER field is initially loaded by software, it must be set to the same value as that contained in the BITER field. **NOTE:** If the channel is configured to execute a single service request, the initial values of BITER and CITER should be 0x0001.

6.4.5.30 TCD Last Destination Address Adjustment/Scatter Gather Address (TCD0_DLASTSGA - TCD15_DLASTSGA)

6.4.5.30.1 Offset

For n = 0 to 15:

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

6.4.5.30.2 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

6.4.5.30.3 Fields

Field	Function
31-0 DLASTSGA	DLASTSGA Destination last address adjustment or the memory address for the next transfer control descriptor to be loaded into this channel (scatter/gather). If (TCDn_CSR[ESG] = 0) then: • 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 transfer control descriptor 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.

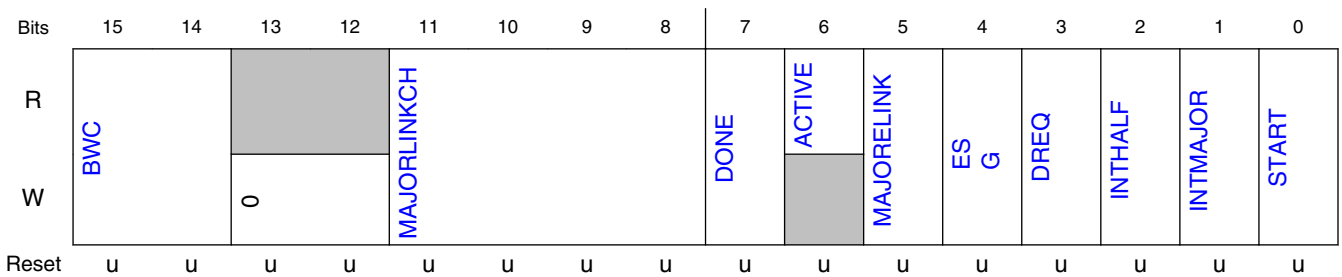
6.4.5.31 TCD Control and Status (TCD0_CSR - TCD15_CSR)

6.4.5.31.1 Offset

For n = 0 to 15:

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

6.4.5.31.2 Diagram



6.4.5.31.3 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.
13-12 —	Reserved
11-8 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 TCDn_CSR[START] bit.
7 DONE	Channel Done

Table continues on the next page...

Field	Function
	This flag indicates the eDMA has completed the major loop. The eDMA engine sets it as the CITER count reaches zero. The software clears it, or the hardware when the channel is activated. **NOTE:** This bit must be cleared to write the MAJORELINK or ESG bits.
6 ACTIVE	Channel Active This flag signals the channel is currently in execution. It is set when channel service begins, and is cleared by the eDMA 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 flag enables the linking to another channel, defined by MAJORLINKCH. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit of the specified channel. **NOTE:** To support the dynamic linking coherency model, this field is forced to zero when written to while the TCDn_CSR[DONE] bit is set. 0b - The channel-to-channel linking is disabled. 1b - The channel-to-channel linking is enabled.
4 ESG	Enable Scatter/Gather Processing As the channel completes the major loop, this flag enables scatter/gather processing in the current channel. If enabled, the eDMA engine uses DLASTSGA as a memory pointer to a 0-modulo-32 address containing a 32-byte data structure loaded as the transfer control descriptor into the local memory. **NOTE:** To support the dynamic scatter/gather coherency model, this field is forced to zero when written to while the TCDn_CSR[DONE] bit is set. 0b - The current channel's TCD is normal format. 1b - The current channel's TCD specifies a scatter gather format. The DLASTSGA field provides a memory pointer to the next TCD to be loaded into this channel after the major loop completes its execution.
3 DREQ	Disable Request If this flag is set, the eDMA hardware automatically clears the corresponding ERQ bit when the current major iteration count reaches zero. 0b - The channel's ERQ bit is not affected. 1b - The channel's ERQ bit is cleared when the major loop is complete.
2 INTHALF	Enable an interrupt when major counter is half complete. If this flag is set, the channel generates an interrupt request by setting the appropriate bit 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 - The half-point interrupt is disabled. 1b - The half-point interrupt is enabled.
1 INTMAJOR	Enable an interrupt when major iteration count completes. If this flag is set, the channel generates an interrupt request by setting the appropriate bit in the INT when the current major iteration count reaches zero. 0b - The end-of-major loop interrupt is disabled. 1b - The end-of-major loop interrupt is enabled.
0 START	Channel Start If this flag is set, the channel is requesting service. The eDMA hardware automatically clears this flag after the channel begins execution. 0b - The channel is not explicitly started. 1b - The channel is explicitly started via a software initiated service request.

6.4.5.32 TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Disabled) (TCD0_BITER_ELINKNO - TCD15_BITER_ELINKNO)

6.4.5.32.1 Offset

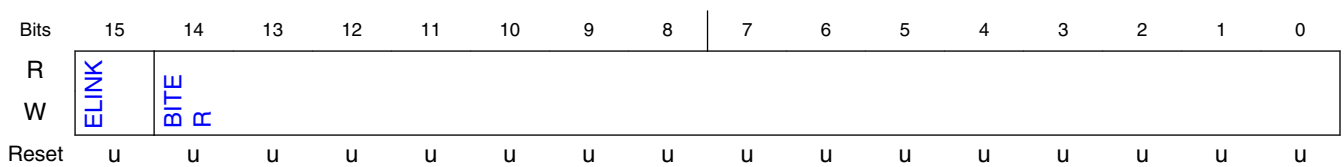
For n = 0 to 15:

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

6.4.5.32.2 Function

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

6.4.5.32.3 Diagram



6.4.5.32.4 Fields

Field	Function
15 ELINK	Enables channel-to-channel linking on minor loop complete As the channel completes the minor loop, this flag enables the linking to another channel, defined by BITER[LINKCH]. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit 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 - The channel-to-channel linking is disabled 1b - The channel-to-channel linking is enabled
14-0 BITER	Starting Major Iteration Count

Field	Function
	As the transfer control descriptor 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 is 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.

6.4.5.33 TCD Beginning Minor Loop Link, Major Loop Count (Channel Linking Enabled) (TCD0_BITER_ELINKYES - TCD15_BITER_ELINKYES)

6.4.5.33.1 Offset

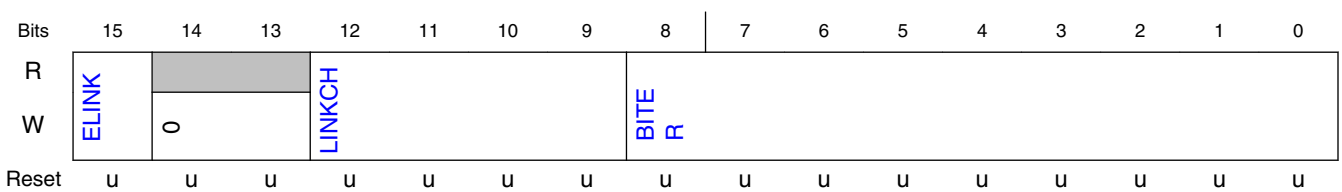
For n = 0 to 15:

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

6.4.5.33.2 Function

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

6.4.5.33.3 Diagram



6.4.5.33.4 Fields

Field	Function
15 ELINK	Enables channel-to-channel linking on minor loop complete

Table continues on the next page...

Field	Function
	As the channel completes the minor loop, this flag enables the linking to another channel, defined by BITER[LINKCH]. The link target channel initiates a channel service request via an internal mechanism that sets the TCDn_CSR[START] bit 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 - The channel-to-channel linking is disabled 1b - The channel-to-channel linking is enabled
14-13 —	Reserved
12-9 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] bit. **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 transfer control descriptor 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 Functional description

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

6.5.1 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:

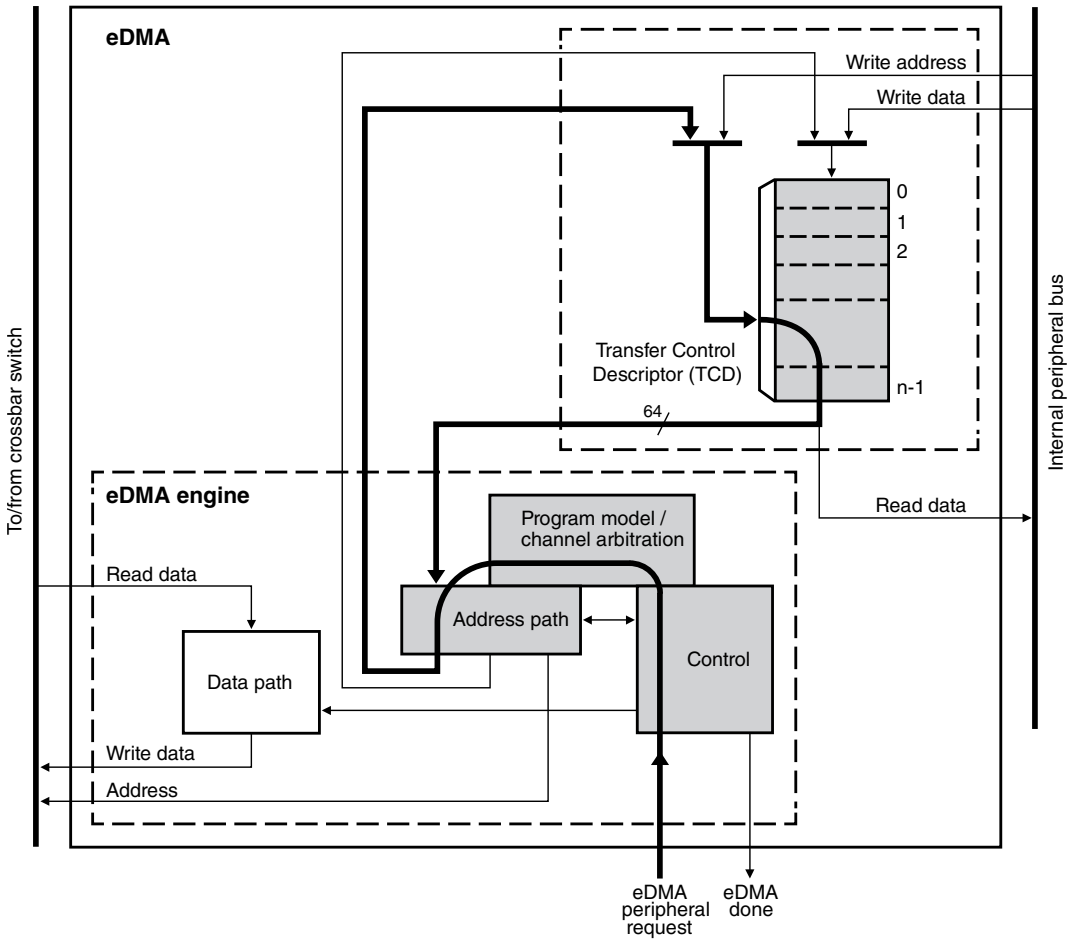


Figure 6-2. eDMA operation, part 1

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 performs, using 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 address path channel x or y registers. The TCD memory is 64 bits wide to minimize the time needed to fetch the activated channel descriptor and load it into the address path channel x or y registers.

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

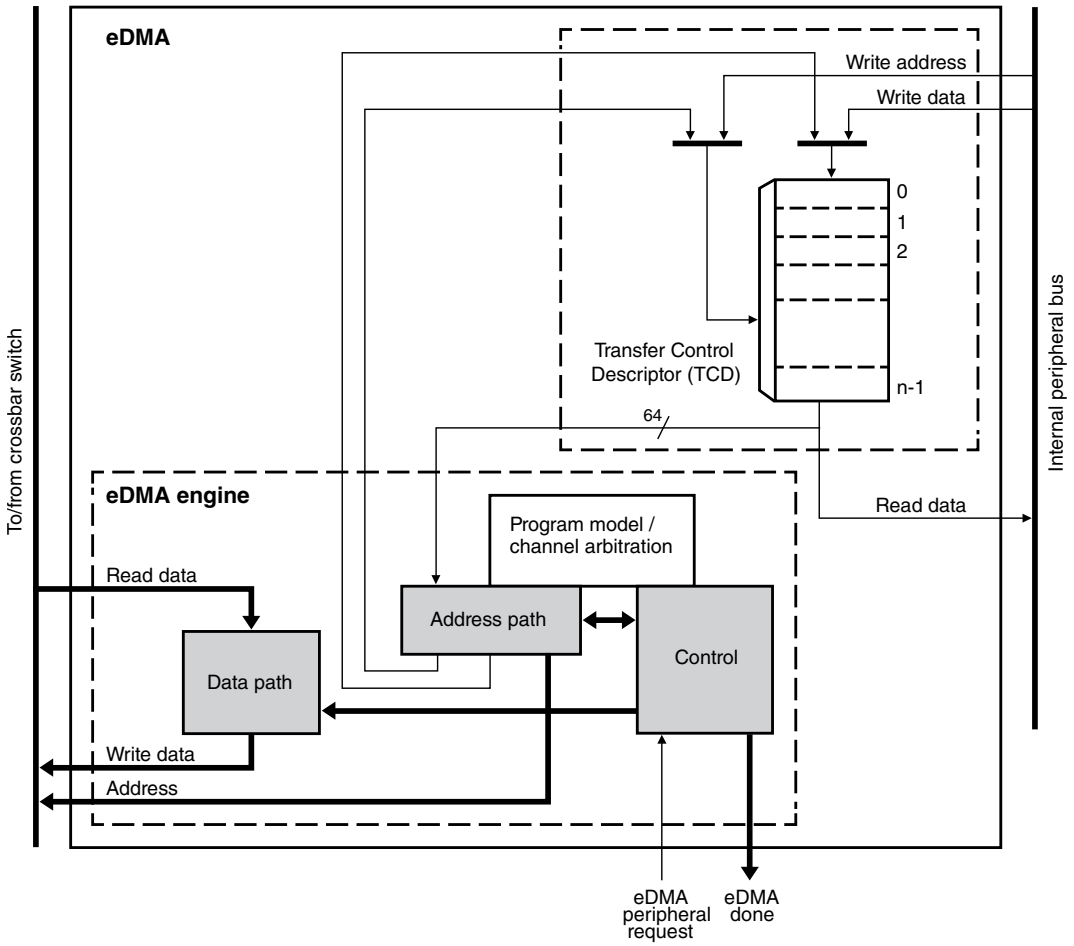


Figure 6-3. eDMA operation, part 2

The modules associated with the data transfer (address path, data path, and control) sequence through the required source reads and destination writes to perform the actual 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.

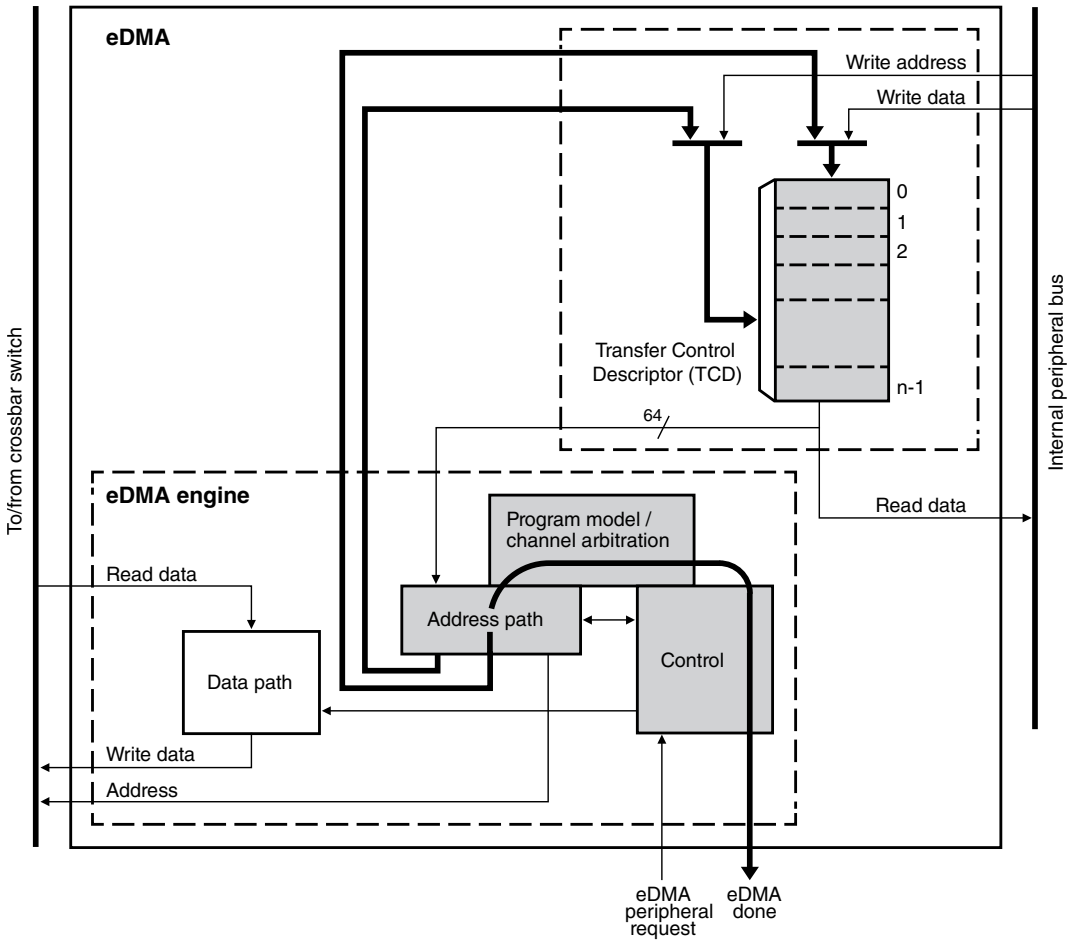


Figure 6-4. eDMA operation, part 3

6.5.2 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 are 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 will be reported in the Error Status register. However, the channel number will not be reported in the Error Status register. When all of the channel priorities within a group are not unique, the channel number selected by arbitration in undetermined.

To aid in Channel Priority Error (CPE) debug, set the Halt On Error bit in the DMA's Control Register. If all of the channel priorities within a group are not unique, the DMA will be halted after the CPE error is recorded. The DMA will remain halted and will not process any channel service requests. Once 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[E_LINK] bit does not equal the TCDn_BITER[E_LINK] 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 cancelled 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 cancelled channel number and ECX is set. The TCD of a cancelled 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 cancelled 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 allows the user to stop a large data transfer in the event 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 handle 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 since 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.

6.5.3 Channel preemption

Channel preemption is enabled on a per-channel basis by setting the DCHPRIn[ECP] bit. Channel preemption allows the executing channel's data transfers to temporarily suspend in favor of starting a higher priority channel. After the preempting channel has completed all 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 preempt ability is disabled, that channel cannot suspend a lower priority channel's data transfer, regardless of the lower priority channel's ECP setting. This allows for 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.

6.6 Initialization/application information

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

6.6.1 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 so 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 the 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 written into the programmer's model. The eDMA engine reads the entire TCD, including the TCD control and status fields, as shown in the following table, for the selected channel into its internal address path module.

As the TCD is read, the first transfer is initiated on the internal 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 are 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 6-5. 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)
D_REQ	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
E_SG	Control bit to enable scatter-gather feature
INT_HALF	Control bit to enable interrupt when major loop is half complete
INT_MAJ	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).

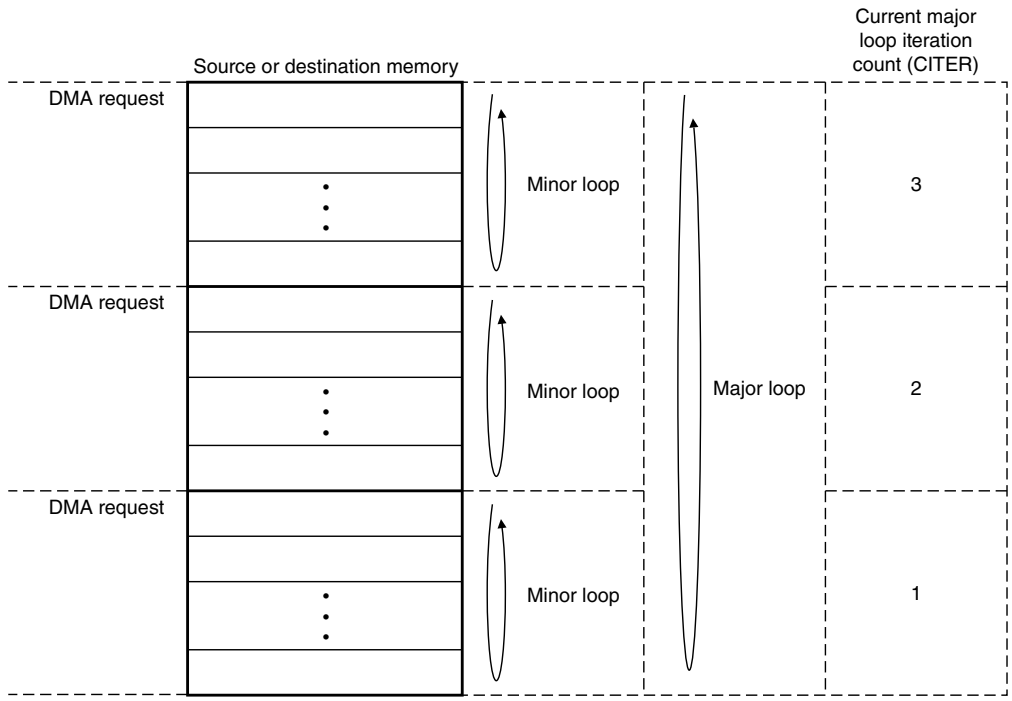


Figure 6-5. Example of multiple loop iterations

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

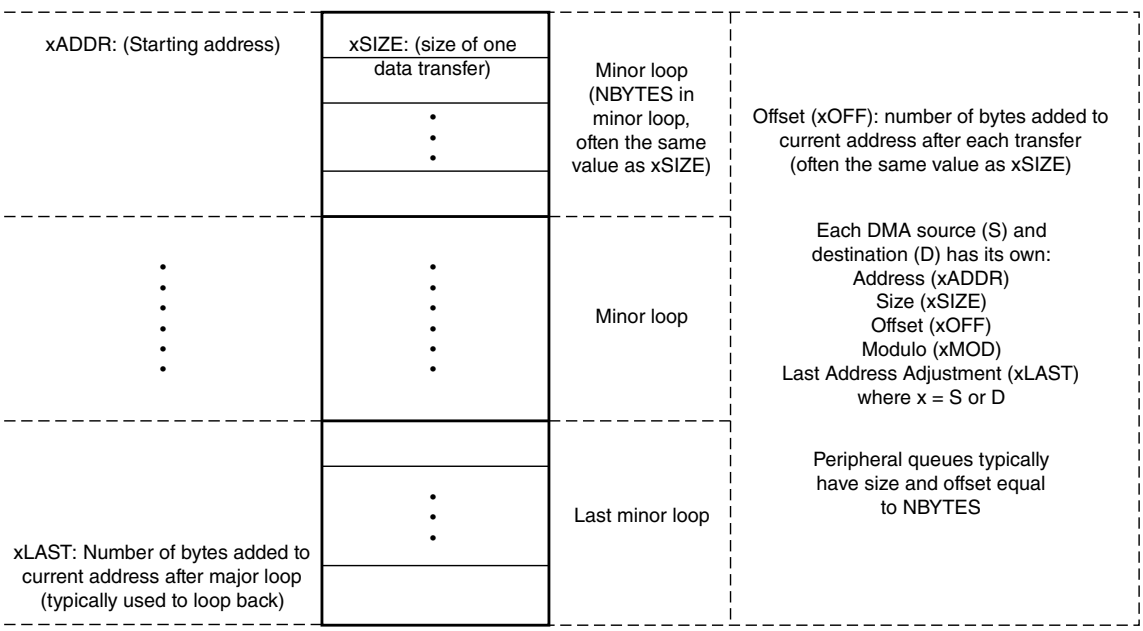


Figure 6-6. Memory array terms

6.6.2 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 channel priority error, 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.

If priority levels are not unique, when any channel requests service, a channel priority error is reported. The highest channel 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 is associated with the selected channel.

6.6.3 Arbitration mode considerations

This section discusses arbitration considerations for the eDMA.

6.6.3.1 Fixed channel arbitration

In this mode, the channel service request from the highest priority channel is selected to execute.

6.6.3.2 Round-robin channel arbitration

Channels are serviced starting with the highest channel number and rotating through to the lowest channel number without regard to the channel priority levels.

6.6.4 Performing DMA transfers

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

6.6.4.1 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, the TCD*n*_CSR[DONE] bit 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 a byte wide 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[INT_MAJ] = 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.

6.6.4.2 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, eDMA peripheral, request for 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.

6.6.4.3 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 0 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 6-6. Modulo example

Transfer Number	Address
1	0x12345670
2	0x12345674
3	0x12345678
4	0x1234567C
5	0x12345670
6	0x12345674

6.6.5 Monitoring transfer descriptor status

This section discusses how to monitor eDMA status.

6.6.5.1 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 the TCD*n*_CSR[START] bit and the TCD*n*_CSR[ACTIVE] bit. The minor-loop-complete condition is indicated by both bits reading zero after the 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 hardware, that is, peripheral, initiated service requests 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.

6.6.5.2 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 while a channel executes. The true values of the 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.

6.6.5.3 Checking channel preemption status

Preemption is available only when fixed arbitration is selected as the channel arbitration mode. 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 channel arbitration mode, the determination of the actively running relative priority outstanding requests become undefined. Channel 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.

6.6.6 Channel Linking

Channel linking (or chaining) is a mechanism where one channel sets the TCD*n*_CSR[START] bit of another channel (or itself), therefore 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[E_LINK] 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[E_LINK] = 1
TCDn_CITER[LINKCH] = 0xC
TCDn_CITER[CITER] value = 0x4
```

```
TCDn_CSR[MAJOR_E_LINK] = 1
TCDn_CSR[MAJOR_LINKCH] = 0x7
```

executes as:

1. Minor loop done → set TCD12_CSR[START] bit
2. Minor loop done → set TCD12_CSR[START] bit
3. Minor loop done → set TCD12_CSR[START] bit
4. Minor loop done, major loop done→ set TCD7_CSR[START] bit

When minor loop linking is enabled (TCD*n*_CITER[E_LINK] = 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[E_LINK] = 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[E_LINK] bit and the TCD*n*_BITER[E_LINK] bit must 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, i.e, use another channel's TCD, at the end of a loop.

Table 6-7. Channel Linking Parameters

Desired Link Behavior	TCD Control Field Name	Description
Link at end of Minor Loop	CITER[E_LINK]	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_E_LINK]	Enable channel-to-channel linking on major loop completion
	CSR[MAJOR_LINKCH]	Link channel number when linking at end of major loop

6.6.7 Dynamic programming

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

6.6.7.1 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.

6.6.7.2 Dynamic channel linking

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

Because the user is allowed to change the configuration during execution, a coherency model is needed. Consider the scenario where the user attempts to execute a dynamic channel link by enabling the TCDn_CSR[MAJORELINK] bit at the same time the eDMA engine is retiring the channel. The 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 1 to the TCDn_CSR[MAJORELINK] bit.
2. Read back the TCDn_CSR[MAJORELINK] bit.
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 the TCDn_CSR[MAJORELINK] bit 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

The user must clear the The TCDn_CSR[DONE] bit before writing the TCDn_CSR[MAJORELINK] bit. The

TCDn_CSR[DONE] bit is cleared automatically by the eDMA engine after a channel begins execution.

6.6.7.3 Dynamic scatter/gather

Scatter/gather is the process of automatically loading a new TCD into a channel. It allows 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 the user is allowed to change the configuration during execution, a coherency model is needed. Consider the scenario where the user attempts 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.E_LINK and E_SG 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.

6.6.7.3.1 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 the TCDn_CSR[MAJORELINK] bit is zero, the TCDn_CSR[MAJORLINKCH] field is not used by the eDMA. In this case, the MAJORLINKCH field 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 the TCDn_CSR[MAJORLINKCH] field for each TCD associated with a channel using dynamic scatter/gather.
2. Write 1b to the TCDn_CSR[DREQ] bit.

   Should a dynamic scatter/gather attempt fail, setting the DREQ bit will prevent a future hardware activation of this 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 final offset value.
3. Write the TCDn_DLASTSGA register with the scatter/gather address.
4. Write 1b to the TCDn_CSR[ESG] bit.
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 = 1b, the dynamic link attempt was successful.

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

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

6.6.7.3.2 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 1b to the TCDn_CSR[DREQ] bit.

   Should a dynamic scatter/gather attempt fail, setting the DREQ bit will prevent a future hardware activation of this 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 final offset value.
2. Write the TCDn_DLASTSGA register with the scatter/gather address.
3. Write 1b to the TCDn_CSR[ESG] bit.
4. Read back the ESG bit.
5. Test the ESG request status:

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

If ESG = 0b, read the 32 bit TCDn_DLASTSGA field.

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

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

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

The DMA allows the user to move data from memory or peripheral registers to another location in memory or peripheral registers without CPU interaction. Once 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.

6.6.8.1 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.

6.6.8.2 Resume a DMA channel

To resume a DMA channel:

1. Enable the DMA service request on the appropriate channel by setting the its 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 will transfer the next command and data to the TXFIFO upon the request. If the user needs to suspend the DMA/SPI transfer loop, perform the following steps:

1. Disable the DMA service request at the source by writing 0 to SPI_RSER[TFFF_RE]. Confirm that SPI_RSER[TFFF_RE] is 0.
2. Ensure there is no DMA service request from the SPI by verifying that DMA_HRS[HRS*n*] is 0 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.

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.

- 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, with Cryptographic Hash Engine, Public Key Cryptography Engine and Random Number Generation
- 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
- On-the-Fly AES Decryption (OTFAD)

7.2 Feature summary

This figure shows a simplified diagram of the security subsystem:

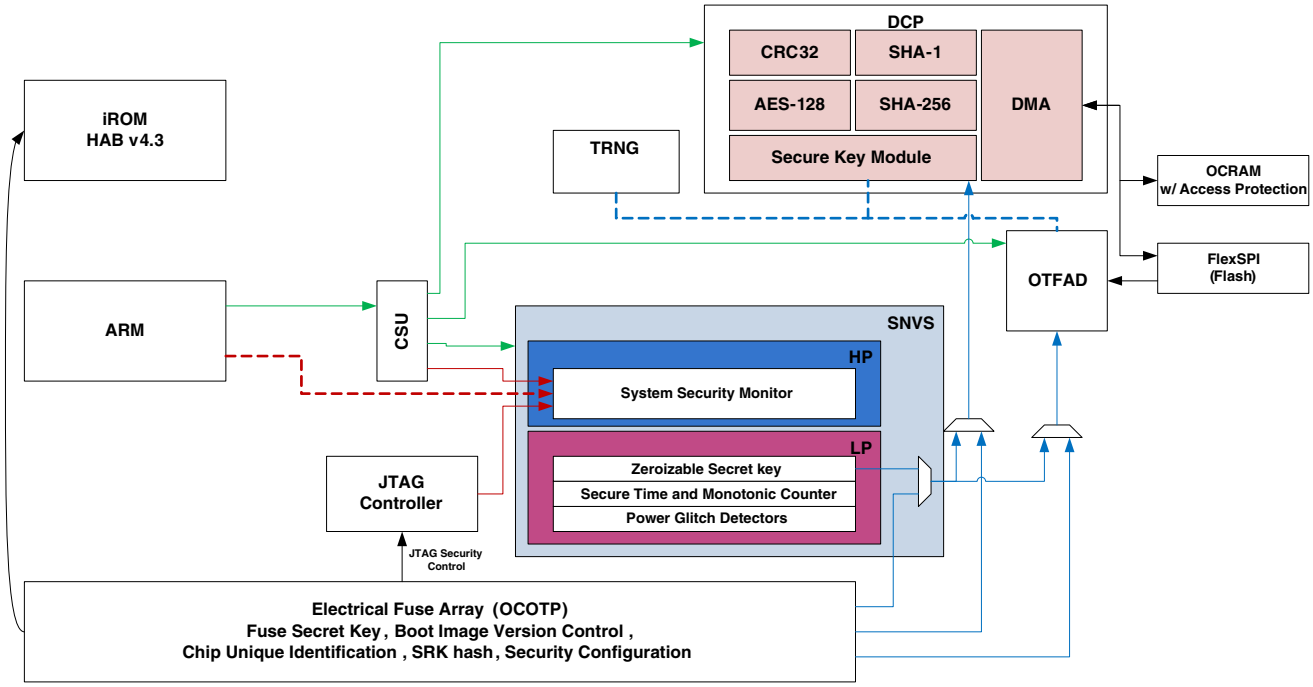


Figure 7-1. Security subsystem (simplified)

This diagram represents an example of the CSU use.

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 the automotive or industrial platforms. Each market must be protected against different kinds of attacks. The platform designers need an appropriate set of counter measures 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 tamper-proof 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,
- True and pseudorandom number generator
- On-chip secure real-time clock with autonomous power domain
- 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
- Universal unique ID
- Electrical fuses (OTP Memory)
- AES-128 counter mode On-The-Fly Decryption
    - 128-bit key and 128-bit data sizes
    - Receives 64-bit encrypted data from FlexSPI
- 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 repurposing the platform. The unauthorized software can enter the platform during upgrades or reprovisioning, 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.

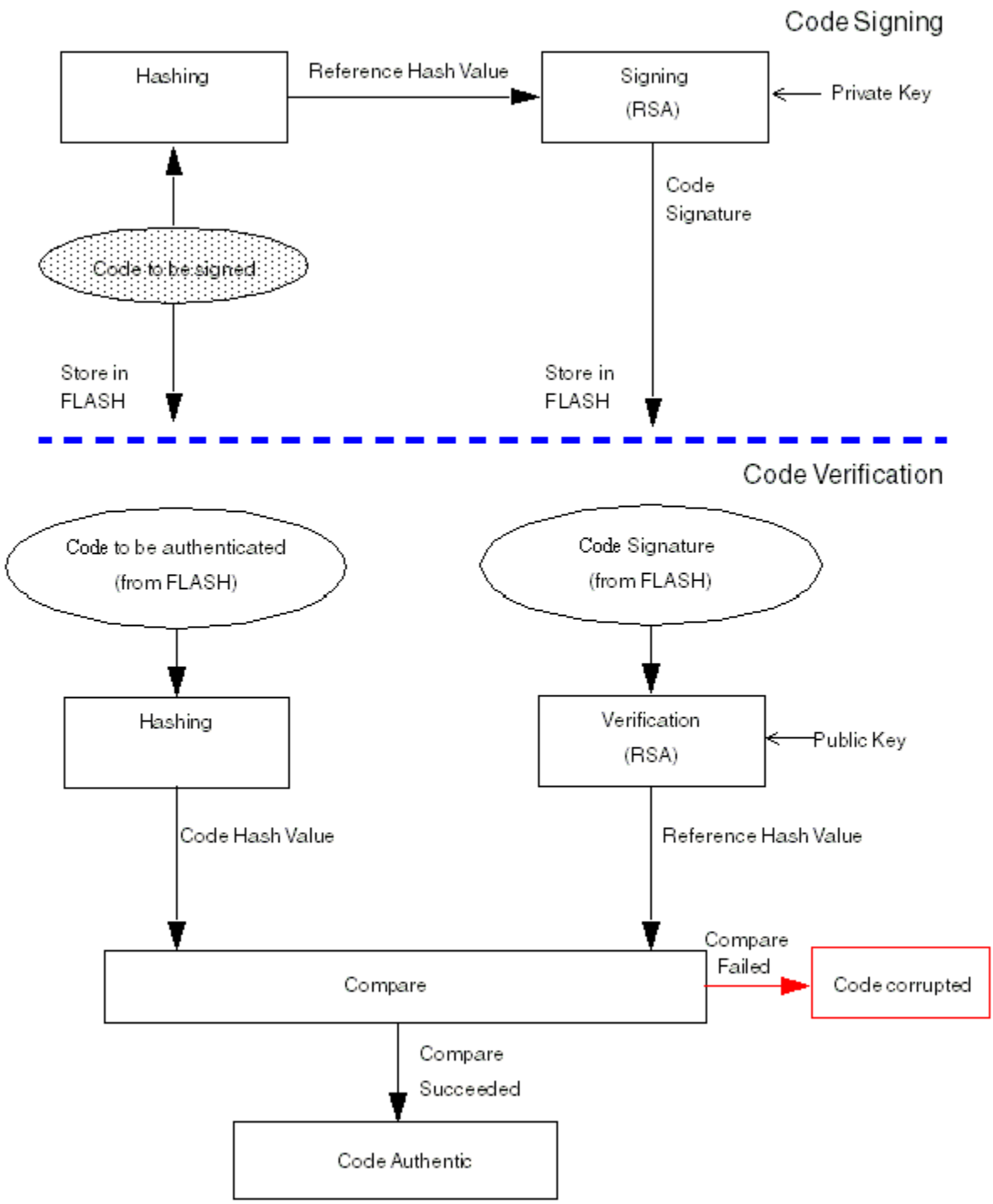


Figure 7-2. Code signing and authentication processes

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 the RSA public keys (from 1024-bit to 4096-bit), and the SHA-256 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
- Authenticated USB download fall-over on any security failure
- 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.
- 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
- Power glitch detector
- 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

For security purposes, the fuses protect the confidentiality or integrity of the critical security data against both the software attacks and the board-level hardware attacks.

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 e-fuse 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 On-the-Fly AES Decryption (OTFAD)

7.10.1 OTFAD Overview

OTFAD co-work with FlexSPI to provide superior cryptographic decryption capabilities without compromising system performance.

It has the following features:

- AES-128 Counter Mode On-the-Fly Decryption

    - 128-bit key and 128-bit data sizes
    - Receives 64-bit encrypted data from FlexSPI
- Functionally acts as a slave submodule to the FlexSPI
    - Private 64-bit data buses for encrypted (ciphertext) and decrypted (plaintext) data
    - Programming model mapped into the upper 1 KB of the FlexSPI's IPS address space

See the "On-the-Fly AES Decryption Module (OTFAD)" chapter in the Security Reference Manual for more information on the OTFAD.

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
- Resources associated with complex IP blocks
- Chip-wide resources
- Interface to the external debug and development tools

The debug and trace architecture is designed around the following:

- Arm CoreSight architecture, adapted to SoC (for core debug), including a cross-trigger subsystem for cross-domain triggering of debug resources
- 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
- Various chip-wide resources, such as debug features built into the IP blocks and critical signal visibility available through alternate pin functions or observability muxes

8.2 Chip and Arm Platform Debug Architecture

The Arm Debug architecture is based on the CoreSight architecture by Arm. The CoreSight architecture provides a system-wide solution to real-time debug and trace.

The CoreSight architecture is embodied in a set of CoreSight components and compliant processors that form the CoreSight systems. 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

The chip includes Arm CoreSight components for debug and trace solutions.

8.2.2 Debug system components

The CoreSight components include:

- ETM (Embedded Trace Macrocell) supporting instruction trace
- ITM (Instrumentation Trace Macrocell)
- TPIU (Trace Port Interface)
- Cross Triggering logic for event routing, including CTIs

Other related IPs and functionality:

- Flash Patch and Breakpoint unit (FPB)

8.2.2.1 AMBA Trace Bus (ATB)

ATB transfers trace data though the chip CoreSight infrastructure. The trace sources are ATB masters and the sinks are ATB slaves. The Arm (via PTM) cores are the data generators. Link components such as the Trace Funnel and Replicator provide both the master and slave interfaces.

The ATB protocol supports:

- Stalling of trace sources to enable the CoreSight components to funnel and combine the sources into a single trace stream
- Association of the trace data with the generating source using trace source IDs. The CoreSight system can trace up to 111 different items at any time
- Capture and transfer of multiple byte bus widths, currently to 32 bits
- A flushing mechanism to force the historic trace to drain from any sources, links, or sinks up to the point that the request is initiated

8.2.2.2 CoreSight trace port interface (TPIU)

TPIU (Cortex-M7)

TPIU is one of the CoreSight trace sink components. It acts as a bridge between the on-chip trace data and a data stream that is then driven out the trace port.

TPIU uses the ATB interface to accept trace data from a trace source, either directly or by using a trace funnel. TPIU has 4 bit port connected to the chip pad.

The APB interface is the programming interface for the TPIU configuration.

The features of the sub-blocks are as follows:

- Formatter—Inserts source ID signals into the data packet stream so that the trace data can be re-associated with the trace source.
- Asynchronous FIFO—Enables trace data to be driven out at a speed that is not dependent on the on-chip bus clock.
- Register Bank—Contains the management, control and status registers for triggers, flushing behavior and external control.
- Trace out—The Trace out block serializes the formatted data before it goes off-chip.
- Pattern Generator—The Pattern Generator unit provides a simple set of defined bit sequences or patterns that can be output over the Trace Port and be detected by the TPA or other associated Trace Capture Device (TCD). The TCD can use these patterns to indicate if it is possible to increase or decrease the trace port clock speed.

The TPIU accepts trace data from a trace source, either direct from a trace source or using a Trace Funnel. The APB interface is the programming interface for the TPIU. The Trace Clock driving the data out to external pins can be obtained from either an on chip or off chip source, selectable via a mux.

The output of the TPIU is connected via external pins (MPS of TRACEDATA (ARM_TRACEn), which can be 1, 2, or 4 bits). 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), and should be selectable via an on chip or off chip clock. TRACECLK (ARM_TRACE_CLK) is equal to TRACECLKIN / 2, and is divided in the TPIU to clock trace data at the trace capture unit.

TPIU used to be part of the Arm platform sub blocks in previous versions of i.MX products, placing the TPIU off platform allows future debug trace sources from the chip level to connect to the TPIU by means of a funnel.

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*.

8.2.3 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 8-1. i.MX JTAG ID

Device	Silicon revision	JTAG ID
i.MX RT1010	Rev 1.0	088C_701Dh

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; puting 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

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. Similarly, there is a register bit in the CCM which allows internal software to turn debug clocks on as well because the CDBGPWRUPREQ is in the TCLK domain and is inaccessible to software.

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
LPUART	1	LPUART1_TX	GPIO_10	ALT0
		LPUART1_RX	GPIO_09	ALT0
FlexSPI	1	FLEXSPI_A_DQS	GPIO_SD_12	ALT0
		FLEXSPI_A_SS0_B	GPIO_SD_06	ALT0
		FLEXSPI_A_SS1_B	GPIO_SD_05	ALT0
		FLEXSPI_A_SCLK	GPIO_SD_10	ALT0
		FLEXSPI_A_DATA0	GPIO_SD_09	ALT0
		FLEXSPI_A_DATA1	GPIO_SD_07	ALT0
		FLEXSPI_A_DATA2	GPIO_SD_08	ALT0
		FLEXSPI_A_DATA3	GPIO_SD_11	ALT0
		FLEXSPI_B_DQS	GPIO_00	ALT0
		FLEXSPI_B_SS0_B	GPIO_SD_00	ALT0
		FLEXSPI_B_SCLK	GPIO_SD_13	ALT0
		FLEXSPI_B_DATA0	GPIO_SD_03	ALT0
		FLEXSPI_B_DATA1	GPIO_SD_01	ALT0
		FLEXSPI_B_DATA2	GPIO_SD_02	ALT0
		FLEXSPI_B_DATA3	GPIO_SD_04	ALT0
		FLEXSPI Reset Pin	GPIO_13	ALT5

NOTE

The ROM bootloader always tries to seek the FLASH configuration block (at FLASH offset 0x400) from FLEXSPI PORTA first, and enables the PORTB if the value of "sflashB1Size" field in it is non-zero.

9.2 Overview

The boot process begins at any Reset where the hardware reset logic forces the Arm core to begin the 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 (USB-HID and UART)
- Digital signature based High-Assurance Boot (HAB)
- Wake-up from the low-power modes
- Encrypted XIP on Serial NOR via FlexSPI interface powered by On-the-Fly AES Decryption (OTFAD) controller

The boot ROM supports boot device as below:

- Serial NOR Flash via FlexSPI

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 to download the 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.

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 programs. 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, the 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 (bootloaders) 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 core'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 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 BOOT_MODE internal 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 boot via USB, 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

Table continues on the next page...

Table 9-2. Boot MODE pin settings (continued)

BOOT_MODE[1:0]	Boot Type
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.

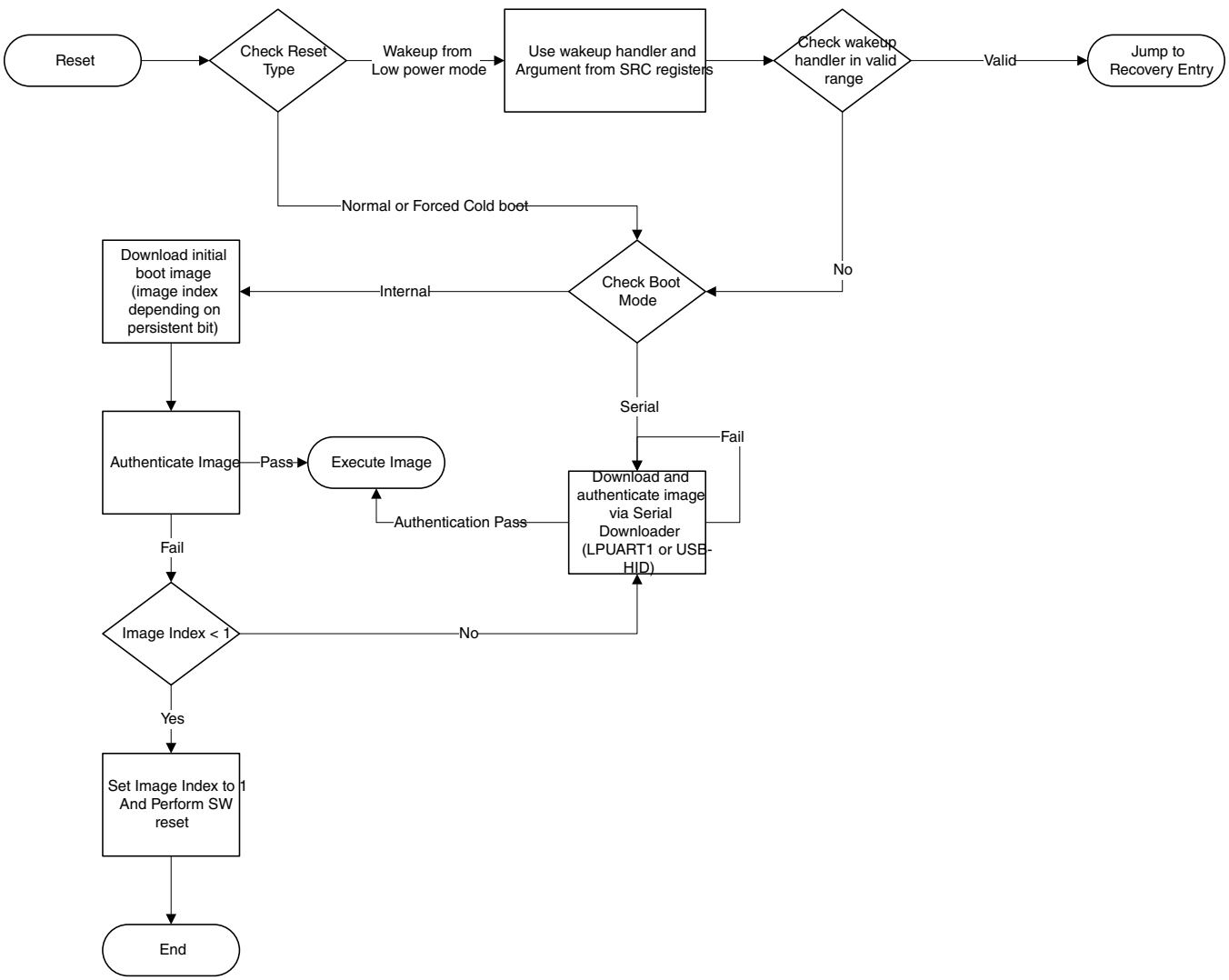


Figure 9-1. Boot flow

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] = 0b10) with one difference. In this mode, the GPIO boot override 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.

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 chip 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] = 0b10)

A value of 0b10 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.

- 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 as for a closed device. 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.

NOTE

If the DIR_BT_DIS eFuse is not blown, the authentication may be bypassed. In this case the system is not secure.

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), 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]
DIR_BT_DIS	OEM	Disables the NXP reserved modes. Must be set for the secure boot. .	NA	0	0—The reserved NXP modes are enabled. 1—The reserved NXP modes are disabled.
BT_FUSE_SEL	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	NA	0	If BOOT_MODE[1:0] = 0b10: • 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] = 0b00 • 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.

Table continues on the next page...

Table 9-3. Boot eFUSE descriptions (continued)

Fuse	Configuration	Definition	GPIO[1]	Shipped value	Settings[2]
		indicates whether the bit configuration eFuses are programmed.			
SEC_CONFIG[1:0]	SEC_CONFIG[0] - NXP SEC_CONFIG[1] - OEM	Security Configuration, as defined in Boot security settings	NA	01	00—Reserved 01—Open (allows any program image, even if the authentication fails) 1x—Closed (The program image executes only if authenticated)
FIELD_RETURN	OEM	Enables the NXP reserved modes.			0—The NXP reserved modes are enabled/ disabled based on the DIR_BT_DIS value. 1—The NXP reserved modes are enabled.
SRK_HASH[255:0]	OEM	256-bit hash value of the super root key (SRK_HASH)	NA	0	Settings vary—used by HAB
UNIQUE_ID[63:0]	NXP	Device Unique ID, 64-bit UID	NA	Unique ID	Settings vary—used by HAB
WDOG_ENABLE	OEM	Watchdog reset counter enable	No	0	0—The watchdog reset counter is disabled during the serial downloader. 1—The watchdog reset counter is enabled during the serial downloader.

1. This setting can be 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:

Table 9-4. GPIO Boot Overrides

Package Pin	Direction on reset	eFuse
GPIO_SD_04/BOOT_MODE0	Input	Boot Mode selection
GPIO_SD_03/BOOT_MODE1	Input	

NOTE

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

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

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:

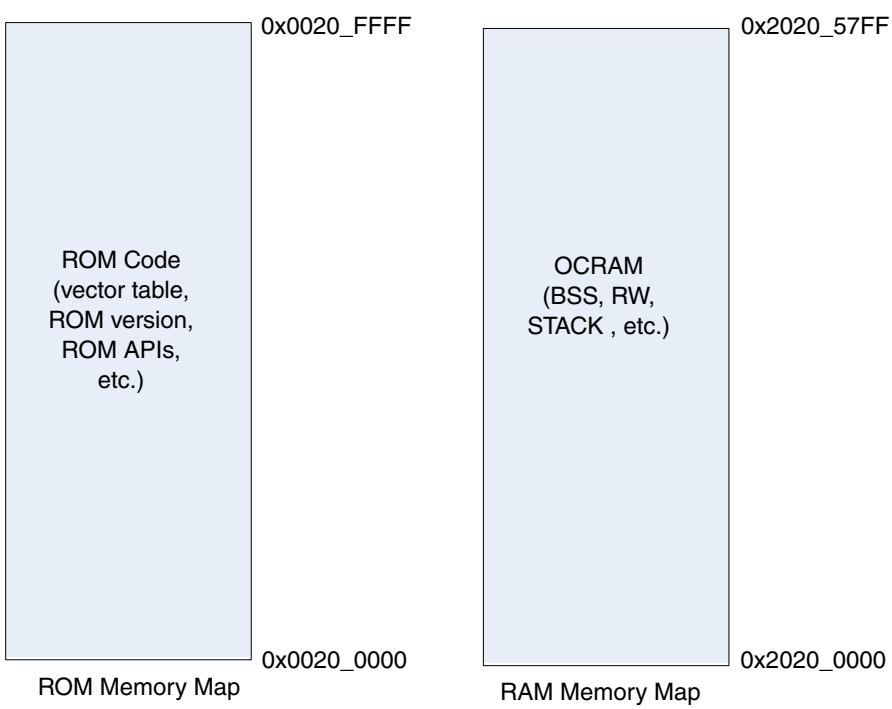


Figure 9-2. Internal ROM and RAM memory map

NOTE

The entire OCRAM region can be used freely after the boot.

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
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
SNVS	Secure Non-Volatile Storage
SRC	System Reset Controller
USB	Used for Serial download of a boot device provisioning program
WDOG 1	WatchDog Timer
TRNG	True Random Number Generator
PIT	Periodic Interrupt Timer

9.5.3 Clocks at boot time

The boot ROM always configures the core clock frequency to 396 MHz during boot. User application needs to take care of the clocks configured by ROM and re-configure them to expected states properly.

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 the CCM Chapter's System Clocks section for information about the CCM output clocks' system-level connectivity.

Table 9-5. List of Disabled Clock Gate Enables

Clock Name
CCM_CCGR0_CG6

Table continues on the next page...

Table 9-5. List of Disabled Clock Gate Enables (continued)

Clock Name
CCM_CCGR0_CG7
CCM_CCGR0_CG8
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

Table continues on the next page...

Table 9-5. List of Disabled Clock Gate Enables (continued)

Clock Name
CCM_CCGR5_CG3
CCM_CCGR5_CG4
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

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. And, the L1 data cache is enabled at the start of image authentication.

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 9-6. 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 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.
ROMAPI_CONTEXT[31:0]	SRC_GPR4[31:0]	Holds the context used for ROM re-entrance API.

9.6 Boot devices

The chip supports the following boot flash devices:

- Serial NOR flash via FlexSPI Interface

9.6.1 Serial NOR Flash Boot via FlexSPI

9.6.1.1 FlexSPI Serial NOR Flash Boot Operation

The Boot ROM attempts to boot from Serial NOR flash , 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 0x400 in Serial NOR flash attached to FLEXSPI_A_SS0_B. The ROM reads these configuration parameters 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 9-11 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.

9.6.1.2 FlexSPI NOR boot flow chart

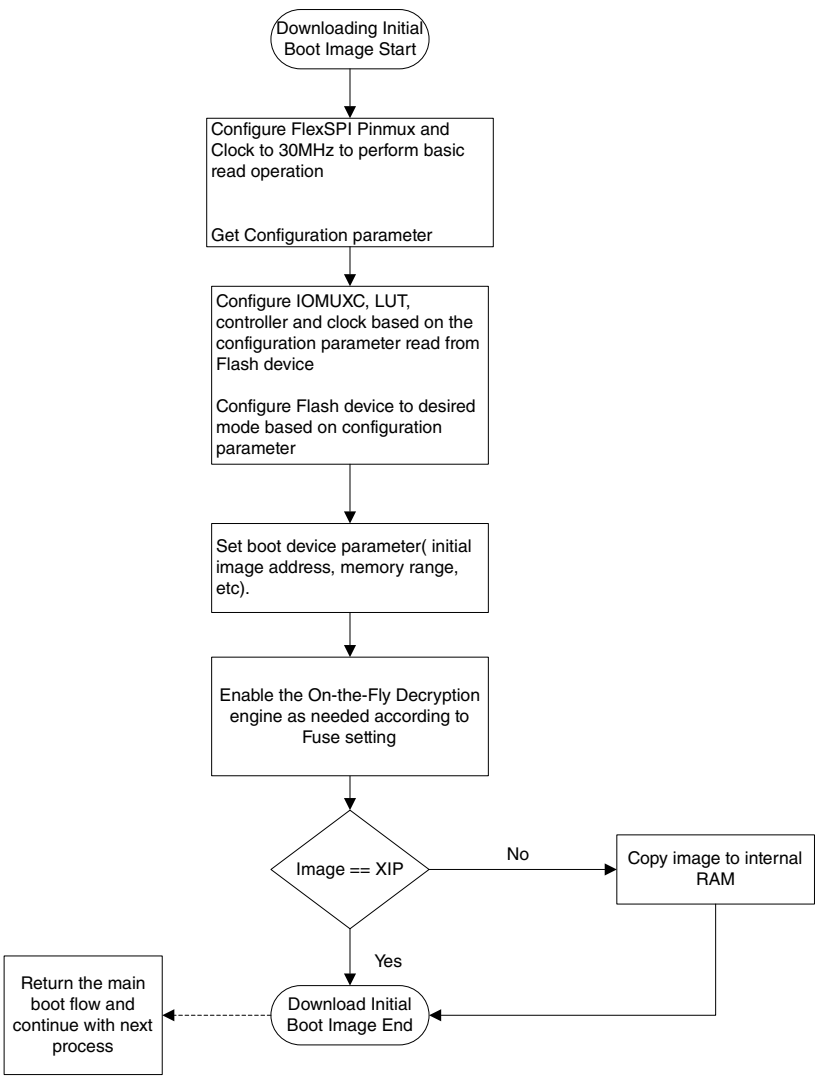


Figure 9-3. FlexSPI NOR boot flow

9.6.2 Encrypted Boot via OTFAD

9.6.2.1 Encrypted Boot on Serial NOR via FlexSPI Interface

Boot images can be encrypted for enhanced security. The BootROM supports boot on the Serial NOR flash device directly with On-the-fly decryption feature (using AES-CTR-128) powered by OTFAD (On-The-Fly AES Decryption) controller. The decryption process is transparent to users.

The OTFAD supports up to 4 separate encrypted regions using separate AES keys. Before doing On-the-fly Decryption boot, the BootROM needs to set the OTFAD controller according to eFUSE configurations. See the table below for eFUSE configurations.

Table 9-7. eFUSE configuration for Encrypt XIP via OTFAD

Fuse	Config	Definition	Settings
OTFAD_KEY_SEL0	OEM	AES KEK selection for OTFAD	0 – Select SNVS_KEY[127:0] 1 – Select SNVS_KEY[255:128] **NOTE:** SNVS_KEY is unique chip to chip and invisible.
OTFAD_KEY_SEL1	OEM	AES KEK selection for OTFAD	0 – From Key selected by OTFAD_KEY_SEL0 1 – From SW_GP2
SW_GP2	OEM	SW general purpose key	128-bit KEK if OTFAD_KEY_SEL1 set 1 **Note**: Be careful of the endianness. E.g. KEK={0x00,0x11,0x22,0x33,0x44,0x55,0x66,0x77,0x88,0x99,0xaa,0xbb,0xcc,0xdd,0xee, 0xff}, SW_GP2 should be configured as follows: [127:96]=0x33221100, [95:64]=0x77665544, [63:32]=0xbbaa9988, [31:0]=0xffeeddcc.
ENB_OTFAD_KEY_SCRAMBLE	OEM	Enable KEK scramble during key blob processing	0 – disable 1 – enable **NOTE:** Set disabled for use with SNVS_KEYs.
OTFAD_KEY_SCRAMBLE	OEM	Key scramble data value for key blob unwrap	

Table continues on the next page...

Table 9-7. eFUSE configuration for Encrypt XIP via OTFAD (continued)

Fuse	Config	Definition	Settings
OTFAD_KEY_SCRAMBLE_ALIGN	OEM	4x2-bit key scramble align select for key blob unwrap	
OTFAD_ENABLE	OEM	Enable OTFAD	0 – disable 1 – enable

A keyblob contains encryption keys for OTFAD, and is always encrypted with a KEK. The KEK can be scrambled for each encryption region. It is equivalent to different KEKs for different regions, bringing in better randomness and complexity for the encrypted keyblob. If ENB_OTFAD_KEY_SCRAMBLE is configured to 1, OTFAD_KEY_SCRAMBLE_ALIGN is used to specify for each region, which part of the original KEK is scrambled. And OTFAD_KEY_SCRAMBLE is the value to scramble the KEK. See "Key blob KEK details" section in the OTFAD chapter for more details of key scrambler.

9.6.2.2 FlexSPI NOR Image Layout

The FlexSPI NOR image layout is as follows:

Table 9-8. FlexSPI NOR Boot Image Layout

Offset	Description
0x0000~0x00FF	OTFAD Keyblob (if present)
0x0100~0x03FF	Reserved
0x0400~0x05FF	Flash Configuration Block
0x0600~0x0FFF	Reserved
0x1000~0x1000+App image size	App image (IVT, Application, CSF, etc.)

NXP provides software tools, or the flash loader, to encrypt app images and generate keyblobs.

9.6.3 Serial NOR configuration based on FlexSPI interface

The ROM SW supports Serial NOR based on FlexSPI module, using a 448-bytes common FlexSPI configuration block and several specified parameters for Serial NOR . 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 9-9. 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 0 – Other devices
deviceModeCfgEnable	0x010	1	Device Mode Configuration Enable feature 0 – Disabled 1 – Enabled
-	0x011	1	Reserved
waitTimeCfgCommands	0x013	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
configCmdEnable	0x01C	1	Config Command Enable feature 0 – Disabled 1 – Enabled

Table continues on the next page...

Table 9-9. FlexSPI Configuration block (continued)

Name	Offset	Size(bytes)	Description
-	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 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 Bit9 – SecondDqsPinmux
deviceType	0x044	1	1 – Serial NOR
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 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
sflashA1Size	0x050	4	For SPI NOR, need to fill with actual size
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

Table continues on the next page...

Table 9-9. FlexSPI Configuration block (continued)

Name	Offset	Size(bytes)	Description
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	0	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 9-10. 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
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

Table continues on the next page...

Table 9-10. LUT sequence definition for Serial NOR (continued)

Command Index	Name	Index in lookup table	Description
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

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 9-11. 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	4	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

Table continues on the next page...

Table 9-11. Serial NOR configuration block (continued)

Name	Offset	Size (Bytes)	Description
			5 – 80 MHz 6 – 100 MHz 7 – 133 MHz
Reserved	0x1CC	52	Reserved for future use

9.6.4 Redundant boot image support

Failures may occur during boot. In this case, ROM supports loading an extra fail-safe boot image, i.e. the redundant boot image, as shown in the figure "Boot flow" of "High-level boot sequence" section in this chapter. Before the redundant image is loaded, FlexSPI is configured to remap the redundant image, and ROM then loads the redundant image from the address space of the primary (failed) image. Fuses listed in the table below need to be configured correctly as mapping parameters.

Table 9-12. eFUSE configuration

Fuse	Config	Definition	Settings
XSPI_FLASH_SEC_IMG_OFFSET	OEM	Redundant image offset in 256K	0 – No redundant image Others – Offset in 256K
XSPI_FLASH_IMG_SIZE	OEM	Redundant image size in 256K, floored to integer	n – n*256K ≤ Redundant image size < (n+1)*256K

A redundant image's initial offset should be 256K-aligned. For example, a 64K-sized image can be written to offset 0x40000 (physical address 0x60040000) as the redundant image. The fuse XSPI_FLASH_SEC_IMG_OFFSET should be configured 0x40000/256K=1, and XSPI_FLASH_IMG_SIZE configured *floor*(64K/256K)=0. If the primary image fails, FlexSPI will then be configured, mapping data within 0x60040000~0x6007FFFF to 0x60000000~0x6003FFFF. So ROM will load redundant image directly from the primary image's space address.

NOTE

When building a redundant image, the load address space should be the remapped address space, i.e. the address space of the primary image. Do **not** use its physical storage address space.

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 9-13. Image Vector Table Offset and Initial Load Region Size

Boot Device Type	Image Vector Table Offset
FlexSPI NOR	4 Kbyte = 0x1000 bytes

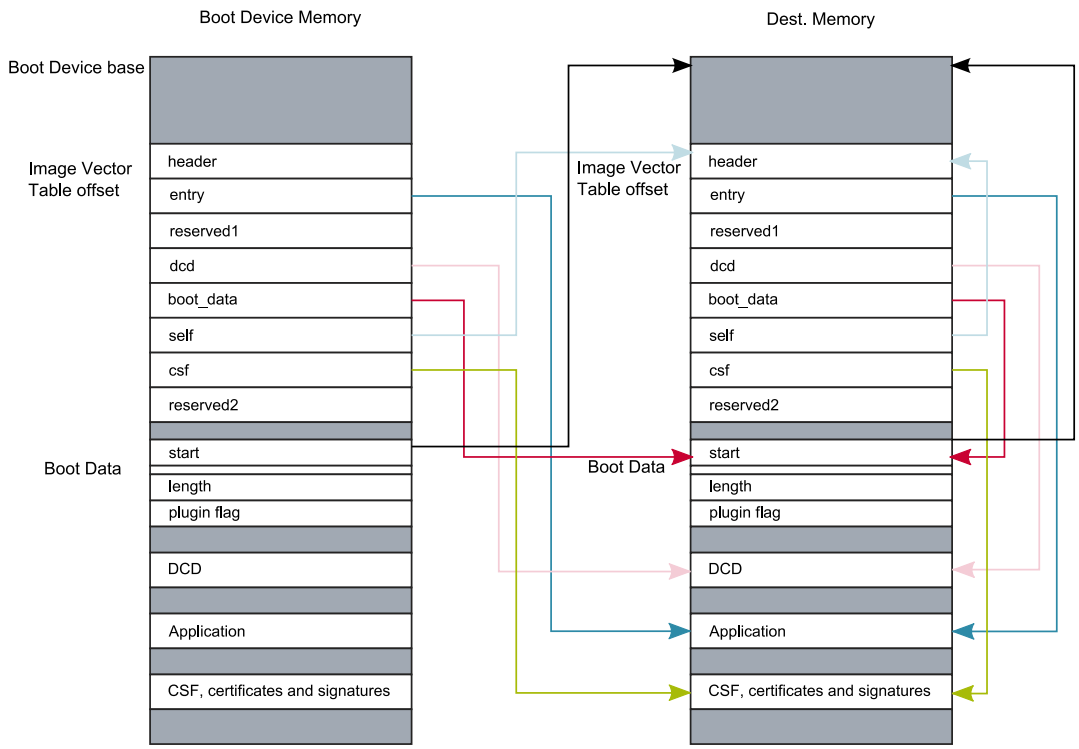


Figure 9-4. Image Vector Table

9.7.1.1 Image vector table structure

The IVT has the following format where each entry is a 32-bit word:

Table 9-14. IVT format

header
entry: Absolute address of the first instruction to execute from the image
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 9-5 shows the IVT header format:

Tag	Length	Version

Figure 9-5. 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 0x40/0x41/0x42/0x43/0x44
```

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 9-15. Boot data format

start	Absolute address of the image
length	Size of the program image
plugin	Plugin flag (must be 0 in this device)

9.8 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:

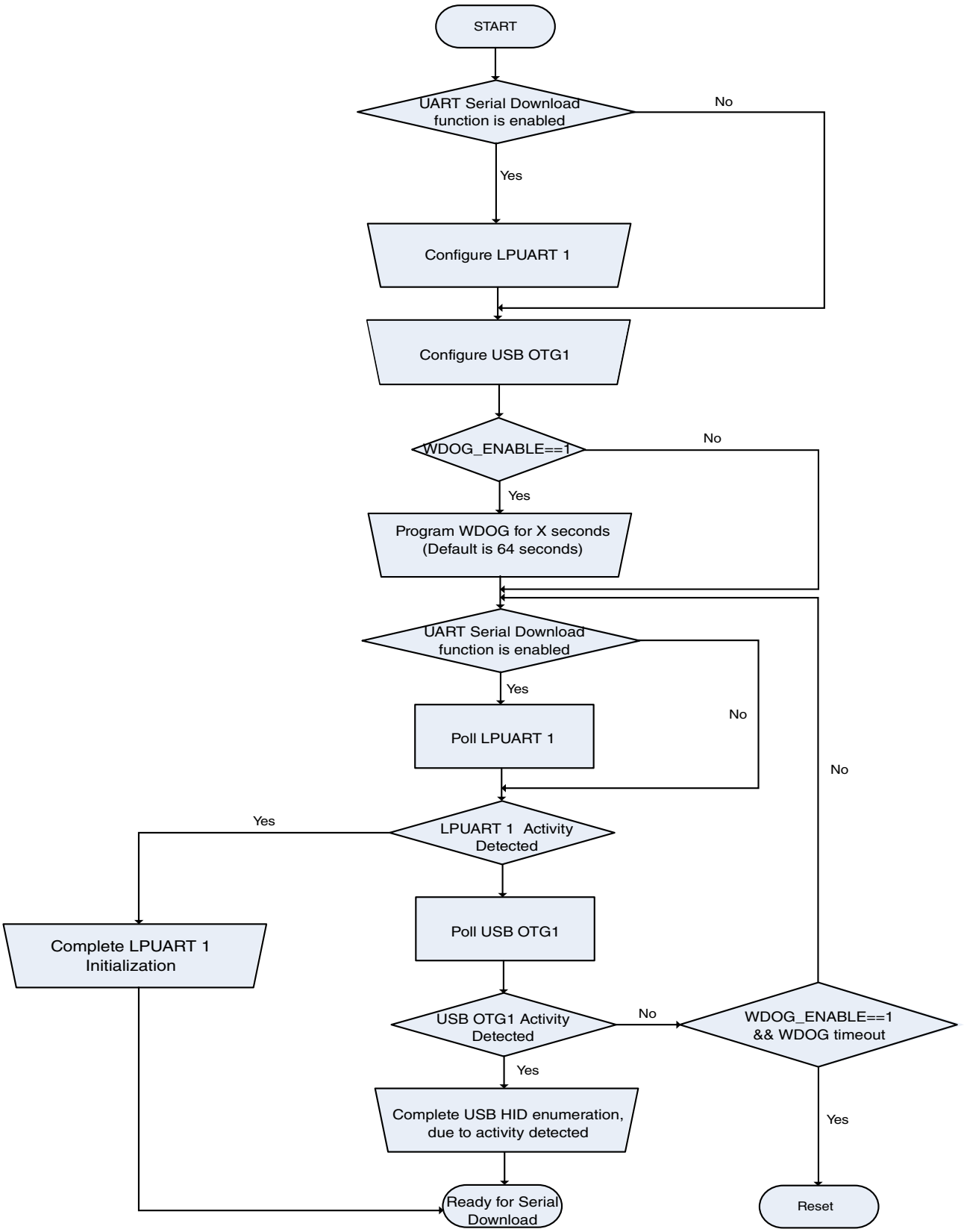


Figure 9-6. Serial Downloader boot flow

9.8.1 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 9-16.

Table 9-16. 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.

9.8.1.1 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 9-17. VID/PID and strings for USB device driver

Descriptor	Value
VID	0x1FC9 (NXP vendor ID)
PID[1]	0x0145
String Descriptor1 (manufacturer)	NXP Semiconductors
String Descriptor2 (product)	S Blank

Table continues on the next page...

Table 9-17. 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)

9.8.1.2 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.

9.8.2 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 9-18. 16-byte SDP command data structure

BYTE offset	Size	Name	Description
0	2	COMMAND TYPE	These commands are supported for the ROM: • 0x0404 WRITE_FILE • 0x0505 ERROR_STATUS • 0x0B0B JUMP_ADDRESS • 0x0D0D SET_BAUDRATE (only applicable to UART)
2	4	ADDRESS	Only relevant for these commands: WRITE_FILE, and JUMP_ADDRESS. 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	
7	4	DATA COUNT	Size of the data to read or write in bytes. Only relevant for the WRITE_FILE commands.
11	4	DATA	
15	1	RESERVED	Reserved

9.8.2.1 SDP commands

The SDP commands are described in the following sections.

9.8.2.1.1 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.

9.8.2.1.2 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

9.8.2.1.3 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.

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

9.8.2.1.4 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.

9.9 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.

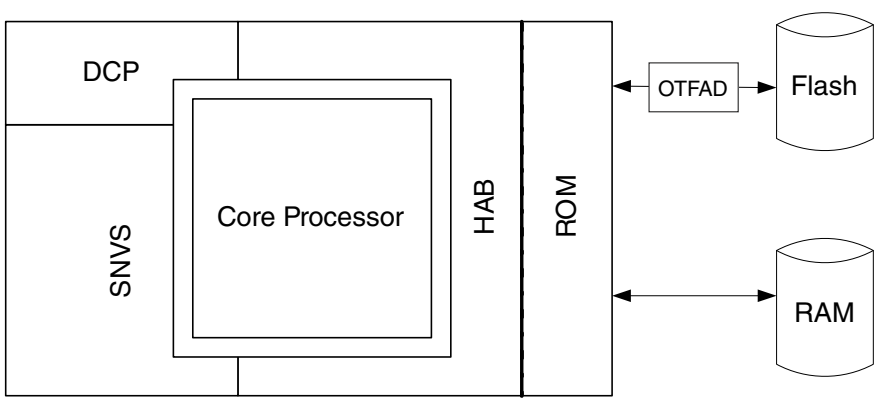


Figure 9-7. Secure boot components

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 RSA key sizes supported are 1024, 2048, 3072 and 4096 bits. The RSA signature verification operations are performed by a software implementation contained in the HAB library. 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.

NOTE

For further details on using the secure boot feature using HAB, contact your local NXP representative.

9.9.1 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_01E0.

NOTE

For additional information on the secure boot including the HAB API, contact your local NXP representative.

9.10 ROM APIs

9.10.1 Introduction

The ROM bootloader provides a set of ROM APIs.

The ROM bootloader supports the following APIs:

- runBootloader API

The ROM API root pointer is located at address 0x0020001c.

The ROM API structure definitions are as below:

1. 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 uint32_t *reserved1;                                  //!< Reserved

    } bootloader_api_entry_t;
    #define g_bootloaderTree (*(bootloader_api_entry_t**)(0x0020001c))
```

9.10.2 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
Reserved	[15:04]	
Boot image selection	[03: 00]	0 - Image 0 (Primary image) 1 - Image 1 (Redundant image, with primary image bypassed) **NOTE:** It takes effect only if the boot mode is 0.

Typical use cases

1. Enter Serial downloader mode .

```
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);
```

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

Table 10-1. Muxing Options

Instance	Port	Pad	Mode
ADC1	ADC1_IN0	GPIO_AD_00	—
	ADC1_IN1	GPIO_AD_01	—
	ADC1_IN2	GPIO_AD_02	—
	ADC1_IN3	GPIO_AD_03	—
	ADC1_IN4	GPIO_AD_04	—
	ADC1_IN5	GPIO_AD_05	—
	ADC1_IN6	GPIO_AD_06	—
	ADC1_IN7	GPIO_AD_07	—
	ADC1_IN8	GPIO_AD_08	—
	ADC1_IN9	GPIO_AD_09	—
	ADC1_IN10	GPIO_AD_10	—
	ADC1_IN11	GPIO_AD_11	—
	ADC1_IN12	GPIO_AD_12	—
	ADC1_IN13	GPIO_AD_13	—

Table continues on the next page...

Table 10-1. Muxing Options (continued)

Instance	Port	Pad	Mode
	ADC1_IN14	GPIO_AD_14	—
ARM	ARM_CM7_RXEV	GPIO_AD_07	ALT2
	ARM_CM7_RXEV	GPIO_SD_00	ALT2
	ARM_TRACE0	GPIO_AD_00	ALT7
	ARM_TRACE1	GPIO_13	ALT7
	ARM_TRACE2	GPIO_12	ALT7
	ARM_TRACE3	GPIO_11	ALT7
	ARM_TRACE_CLK	GPIO_AD_02	ALT7
	ARM_TRACE_SWO	GPIO_AD_09	ALT3
	ARM_TRACE_SWO	GPIO_AD_01	ALT7
	ARM_CM7_TXEV	GPIO_AD_08	ALT2
	ARM_CM7_TXEV	GPIO_SD_13	ALT2
CCM	CCM_CLKO1	GPIO_SD_02	ALT3
	CCM_CLKO2	GPIO_SD_01	ALT3
	CCM_PMIC_RDY	GPIO_SD_13	ALT3
	CCM_REF_EN_B	GPIO_SD_03	ALT3
	CCM_WAIT	GPIO_SD_04	ALT3
	CCM_STOP	GPIO_SD_00	ALT3
EWM	EWM_OUT_B	GPIO_10	ALT2
	EWM_OUT_B	GPIO_AD_08	ALT6
FLEXIO1	FLEXIO1_ IO00	GPIO_08	ALT4
	FLEXIO1_ IO01	GPIO_09	ALT4
	FLEXIO1_ IO10	GPIO_SD_04	ALT4
	FLEXIO1_ IO11	GPIO_SD_05	ALT4
	FLEXIO1_ IO12	GPIO_SD_06	ALT4
	FLEXIO1_ IO13	GPIO_SD_07	ALT4
	FLEXIO1_ IO14	GPIO_SD_08	ALT4
	FLEXIO1_ IO15	GPIO_SD_09	ALT4
	FLEXIO1_ IO16	GPIO_SD_10	ALT4
	FLEXIO1_ IO17	GPIO_SD_11	ALT4
	FLEXIO1_ IO18	GPIO_SD_12	ALT4
	FLEXIO1_ IO19	GPIO_SD_13	ALT4
	FLEXIO1_ IO02	GPIO_10	ALT4
	FLEXIO1_ IO20	GPIO_AD_00	ALT4
	FLEXIO1_ IO21	GPIO_AD_09	ALT4
	FLEXIO1_ IO22	GPIO_AD_10	ALT4
	FLEXIO1_ IO23	GPIO_AD_11	ALT4
	FLEXIO1_ IO24	GPIO_AD_12	ALT4
	FLEXIO1_ IO25	GPIO_AD_13	ALT4

Table continues on the next page...

Table 10-1. Muxing Options (continued)

Instance	Port	Pad	Mode
	FLEXIO1_ IO26	GPIO_AD_14	ALT4
	FLEXIO1_ IO03	GPIO_11	ALT4
	FLEXIO1_ IO04	GPIO_12	ALT4
	FLEXIO1_ IO05	GPIO_13	ALT4
	FLEXIO1_ IO06	GPIO_SD_00	ALT4
	FLEXIO1_ IO07	GPIO_SD_01	ALT4
	FLEXIO1_ IO08	GPIO_SD_02	ALT4
	FLEXIO1_ IO09	GPIO_SD_03	ALT4
FLEXPWM1	FLEXPWM1_PWM0_A	GPIO_SD_02	ALT2
	FLEXPWM1_PWM0_A	GPIO_02	ALT2
	FLEXPWM1_PWM1_A	GPIO_SD_04	ALT2
	FLEXPWM1_PWM1_A	GPIO_04	ALT2
	FLEXPWM1_PWM2_A	GPIO_AD_04	ALT2
	FLEXPWM1_PWM2_A	GPIO_06	ALT2
	FLEXPWM1_PWM3_A	GPIO_AD_06	ALT2
	FLEXPWM1_PWM3_A	GPIO_08	ALT2
	FLEXPWM1_PWM0_B	GPIO_SD_01	ALT2
	FLEXPWM1_PWM0_B	GPIO_01	ALT2
	FLEXPWM1_PWM1_B	GPIO_SD_03	ALT2
	FLEXPWM1_PWM1_B	GPIO_03	ALT2
	FLEXPWM1_PWM2_B	GPIO_AD_03	ALT2
	FLEXPWM1_PWM2_B	GPIO_05	ALT2
	FLEXPWM1_PWM3_B	GPIO_AD_05	ALT2
	FLEXPWM1_PWM3_B	GPIO_07	ALT2
	FLEXPWM1_PWM0_X	GPIO_AD_12	ALT1
	FLEXPWM1_PWM1_X	GPIO_AD_11	ALT1
	FLEXPWM1_PWM2_X	GPIO_AD_10	ALT1
	FLEXPWM1_PWM3_X	GPIO_AD_09	ALT1
FLEXSPI_A	FLEXSPI_A_DATA0	GPIO_SD_09	ALT0
	FLEXSPI_A_DATA1	GPIO_SD_07	ALT0
	FLEXSPI_A_DATA2	GPIO_SD_08	ALT0
	FLEXSPI_A_DATA3	GPIO_SD_11	ALT0
	FLEXSPI_A_DQS	GPIO_SD_14 [1]	ALT0
	FLEXSPI_A_DQS	GPIO_SD_12	ALT0
	FLEXSPI_A_SCLK	GPIO_SD_10	ALT0
	FLEXSPI_A_SS0_B	GPIO_SD_06	ALT0
	FLEXSPI_A_SS1_B	GPIO_SD_05	ALT0
	FLEXSPI_A_SS1_B	GPIO_09	ALT2
FLEXSPI_B	FLEXSPI_B_DATA0	GPIO_SD_03	ALT0

Table continues on the next page...

Table 10-1. Muxing Options (continued)

Instance	Port	Pad	Mode
	FLEXSPI_B_DATA1	GPIO_SD_01	ALT0
	FLEXSPI_B_DATA2	GPIO_SD_02	ALT0
	FLEXSPI_B_DATA3	GPIO_SD_04	ALT0
	FLEXSPI_B_DQS	GPIO_00	ALT0
	FLEXSPI_B_DQS	GPIO_SD_14 [1]	ALT1
	FLEXSPI_B_SCLK	GPIO_SD_13	ALT0
	FLEXSPI_B_SS0_B	GPIO_SD_00	ALT0
	FLEXSPI_B_SS1_B	GPIO_11	ALT3
GPIO2	GPIO2_IO00	GPIO_SD_00	ALT5
	GPIO2_IO01	GPIO_SD_01	ALT5
	GPIO2_IO02	GPIO_SD_02	ALT5
	GPIO2_IO03	GPIO_SD_03	ALT5
	GPIO2_IO04	GPIO_SD_04	ALT5
	GPIO2_IO05	GPIO_SD_05	ALT5
	GPIO2_IO06	GPIO_SD_06	ALT5
	GPIO2_IO07	GPIO_SD_07	ALT5
	GPIO2_IO08	GPIO_SD_08	ALT5
	GPIO2_IO09	GPIO_SD_09	ALT5
	GPIO2_IO10	GPIO_SD_10	ALT5
	GPIO2_IO11	GPIO_SD_11	ALT5
	GPIO2_IO12	GPIO_SD_12	ALT5
	GPIO2_IO13	GPIO_SD_13	ALT5
GPIO5	GPIO5_IO00	PMIC_ON_REQ	ALT5
GPIOMUX	GPIOMUX_IO00	GPIO_00	ALT5
	GPIOMUX_IO01	GPIO_01	ALT5
	GPIOMUX_IO02	GPIO_02	ALT5
	GPIOMUX_IO03	GPIO_03	ALT5
	GPIOMUX_IO04	GPIO_04	ALT5
	GPIOMUX_IO05	GPIO_05	ALT5
	GPIOMUX_IO06	GPIO_06	ALT5
	GPIOMUX_IO07	GPIO_07	ALT5
	GPIOMUX_IO08	GPIO_08	ALT5
	GPIOMUX_IO09	GPIO_09	ALT5
	GPIOMUX_IO10	GPIO_10	ALT5
	GPIOMUX_IO11	GPIO_11	ALT5
	GPIOMUX_IO12	GPIO_12	ALT5
	GPIOMUX_IO13	GPIO_13	ALT5
	GPIOMUX_IO14	GPIO_AD_00	ALT5
	GPIOMUX_IO15	GPIO_AD_01	ALT5

Table continues on the next page...

Table 10-1. Muxing Options (continued)

Instance	Port	Pad	Mode
	GPIOMUX_IO16	GPIO_AD_02	ALT5
	GPIOMUX_IO17	GPIO_AD_03	ALT5
	GPIOMUX_IO18	GPIO_AD_04	ALT5
	GPIOMUX_IO19	GPIO_AD_05	ALT5
	GPIOMUX_IO20	GPIO_AD_06	ALT5
	GPIOMUX_IO21	GPIO_AD_07	ALT5
	GPIOMUX_IO22	GPIO_AD_08	ALT5
	GPIOMUX_IO23	GPIO_AD_09	ALT5
	GPIOMUX_IO24	GPIO_AD_10	ALT5
	GPIOMUX_IO25	GPIO_AD_11	ALT5
	GPIOMUX_IO26	GPIO_AD_12	ALT5
	GPIOMUX_IO27	GPIO_AD_13	ALT5
	GPIOMUX_IO28	GPIO_AD_14	ALT5
GPT1	GPT1_CAPTURE1	GPIO_06	ALT1
	GPT1_CAPTURE2	GPIO_04	ALT1
	GPT1_CLK	GPIO_08	ALT1
	GPT1_COMPARE1	GPIO_07	ALT1
	GPT1_COMPARE2	GPIO_05	ALT1
	GPT1_COMPARE3	GPIO_03	ALT1
GPT2	GPT2_CAPTURE1	GPIO_AD_05	ALT4
	GPT2_CAPTURE2	GPIO_AD_07	ALT4
	GPT2_CLK	GPIO_AD_03	ALT4
	GPT2_COMPARE1	GPIO_AD_04	ALT4
	GPT2_COMPARE2	GPIO_AD_06	ALT4
	GPT2_COMPARE3	GPIO_AD_08	ALT4
JTAG	JTAG_MOD	GPIO_AD_11	ALT7
	JTAG_TCK	GPIO_AD_12	ALT7
	JTAG_TDI	GPIO_AD_10	ALT7
	JTAG_TDO	GPIO_AD_09	ALT7
	JTAG_TMS	GPIO_AD_13	ALT7
	JTAG_TRSTB	GPIO_AD_08	ALT7
	JTAG_DE_B	GPIO_AD_03	ALT7
KPP	KPP_COL0	GPIO_AD_14	ALT2
	KPP_COL0	GPIO_12	ALT2
	KPP_COL1	GPIO_AD_12	ALT2
	KPP_COL1	GPIO_AD_06	ALT3
	KPP_COL2	GPIO_AD_10	ALT2
	KPP_COL2	GPIO_AD_04	ALT3
	KPP_COL3	GPIO_AD_00	ALT2

Table continues on the next page...

Table 10-1. Muxing Options (continued)

Instance	Port	Pad	Mode
	KPP_COL3	GPIO_02	ALT4
	KPP_ROW0	GPIO_AD_13	ALT2
	KPP_ROW0	GPIO_11	ALT2
	KPP_ROW1	GPIO_AD_11	ALT2
	KPP_ROW1	GPIO_AD_05	ALT3
	KPP_ROW2	GPIO_AD_09	ALT2
	KPP_ROW2	GPIO_AD_03	ALT3
	KPP_ROW3	GPIO_13	ALT2
	KPP_ROW3	GPIO_01	ALT4
LPI2C1	LPI2C1_HREQ	GPIO_10	ALT1
	LPI2C1_HREQ	GPIO_AD_06	ALT6
	LPI2C1_SCL	GPIO_AD_14	ALT0
	LPI2C1_SCL	GPIO_SD_06	ALT1
	LPI2C1_SCL	GPIO_12	ALT1
	LPI2C1_SCL	GPIO_02	ALT3
	LPI2C1_SDA	GPIO_AD_13	ALT0
	LPI2C1_SDA	GPIO_SD_05	ALT1
	LPI2C1_SDA	GPIO_11	ALT1
	LPI2C1_SDA	GPIO_01	ALT3
LPI2C2	LPI2C2_SCL	GPIO_AD_08	ALT0
	LPI2C2_SCL	GPIO_SD_08	ALT1
	LPI2C2_SCL	GPIO_AD_02	ALT3
	LPI2C2_SCL	GPIO_10	ALT3
	LPI2C2_SDA	GPIO_AD_07	ALT0
	LPI2C2_SDA	GPIO_SD_07	ALT1
	LPI2C2_SDA	GPIO_AD_01	ALT3
	LPI2C2_SDA	GPIO_09	ALT3
LPSPI1	LPSPI1_PCS0	GPIO_AD_05	ALT0
	LPSPI1_PCS0	GPIO_SD_07	ALT2
	LPSPI1_PCS1	GPIO_AD_02	ALT1
	LPSPI1_PCS2	GPIO_AD_00	ALT1
	LPSPI1_PCS3	GPIO_00	ALT3
	LPSPI1_SCK	GPIO_AD_06	ALT0
	LPSPI1_SCK	GPIO_SD_08	ALT2
	LPSPI1_SDI	GPIO_AD_03	ALT0
	LPSPI1_SDI	GPIO_SD_05	ALT2
	LPSPI1_SDO	GPIO_AD_04	ALT0
	LPSPI1_SDO	GPIO_SD_06	ALT2
LPSPI2	LPSPI2_PCS0	GPIO_AD_11	ALT0

Table continues on the next page...

Table 10-1. Muxing Options (continued)

Instance	Port	Pad	Mode
	LPSPI2_PCS0	GPIO_SD_12	ALT1
	LPSPI2_PCS1	GPIO_AD_01	ALT1
	LPSPI2_PCS2	GPIO_13	ALT1
	LPSPI2_PCS3	GPIO_00	ALT2
	LPSPI2_SCK	GPIO_AD_12	ALT0
	LPSPI2_SCK	GPIO_SD_11	ALT1
	LPSPI2_SDI	GPIO_AD_09	ALT0
	LPSPI2_SDI	GPIO_SD_09	ALT1
	LPSPI2_SDO	GPIO_AD_10	ALT0
	LPSPI2_SDO	GPIO_SD_10	ALT1
LPUART1	LPUART1_CTS_B	GPIO_08	ALT6
	LPUART1_RTS_B	GPIO_07	ALT6
	LPUART1_RXD	GPIO_09	ALT0
	LPUART1_RXD	GPIO_SD_11	ALT2
	LPUART1_TXD	GPIO_10	ALT0
	LPUART1_TXD	GPIO_SD_12	ALT2
LPUART2	LPUART2_CTS_B	GPIO_AD_08	ALT3
	LPUART2_RTS_B	GPIO_AD_07	ALT3
	LPUART2_RXD	GPIO_13	ALT0
	LPUART2_RXD	GPIO_SD_09	ALT2
	LPUART2_TXD	GPIO_AD_00	ALT0
	LPUART2_TXD	GPIO_SD_10	ALT2
LPUART3	LPUART3_CTS_B	GPIO_AD_14	ALT1
	LPUART3_RTS_B	GPIO_AD_13	ALT1
	LPUART3_RXD	GPIO_11	ALT0
	LPUART3_RXD	GPIO_AD_07	ALT1
	LPUART3_RXD	GPIO_07	ALT3
	LPUART3_TXD	GPIO_12	ALT0
	LPUART3_TXD	GPIO_AD_08	ALT1
	LPUART3_TXD	GPIO_08	ALT3
LPUART4	LPUART4_CTS_B	GPIO_AD_14	ALT3
	LPUART4_RTS_B	GPIO_AD_13	ALT3
	LPUART4_RXD	GPIO_AD_01	ALT0
	LPUART4_RXD	GPIO_05	ALT3
	LPUART4_TXD	GPIO_AD_02	ALT0
	LPUART4_TXD	GPIO_06	ALT3
MQS	MQS_LEFT	GPIO_AD_01	ALT4
	MQS_RIGHT	GPIO_AD_02	ALT4
NMI	ARM_NMI	GPIO_AD_13	ALT6

Table continues on the next page...

Table 10-1. Muxing Options (continued)

Instance	Port	Pad	Mode
	ARM_NMI	GPIO_AD_00	ALT6
PIT	PIT_TRIGGER0	GPIO_AD_06	ALT1
	PIT_TRIGGER0	GPIO_00	ALT4
	PIT_TRIGGER1	GPIO_AD_05	ALT1
	PIT_TRIGGER1	GPIO_AD_12	ALT3
	PIT_TRIGGER2	GPIO_AD_04	ALT1
	PIT_TRIGGER2	GPIO_AD_11	ALT3
	PIT_TRIGGER3	GPIO_AD_03	ALT1
	PIT_TRIGGER3	GPIO_AD_10	ALT3
SAI1	SAI1_MCLK	GPIO_08	ALT0
	SAI1_RX_BCLK	GPIO_01	ALT0
	SAI1_RX_DATA0	GPIO_03	ALT0
	SAI1_RX_SYNC	GPIO_02	ALT0
	SAI1_TX_BCLK	GPIO_06	ALT0
	SAI1_TX_DATA0	GPIO_04	ALT0
	SAI1_TX_DATA1 and SAI1_RX_DATA1	GPIO_05	ALT0
	SAI1_TX_SYNC	GPIO_07	ALT0
SAI3	SAI3_MCLK	GPIO_00	ALT1
	SAI3_RX_BCLK	GPIO_SD_13	ALT1
	SAI3_RX_DATA	GPIO_SD_03	ALT1
	SAI3_RX_SYNC	GPIO_SD_04	ALT1
	SAI3_TX_BCLK	GPIO_SD_01	ALT1
	SAI3_TX_DATA	GPIO_SD_02	ALT1
	SAI3_TX_SYNC	GPIO_SD_00	ALT1
SNVS	SNVS_VIO_5_B	GPIO_AD_03	ALT6
	SNVS_VIO_5_CTL	GPIO_AD_04	ALT6
	SNVS_PMIC_ON_REQ	PMIC_ON_REQ	ALT0
SPDIF	SPDIF_EXT_CLK	GPIO_06	ALT4
	SPDIF_EXT_CLK	GPIO_12	ALT6
	SPDIF_IN	GPIO_04	ALT4
	SPDIF_IN	GPIO_10	ALT6
	SPDIF_LOCK	GPIO_07	ALT4
	SPDIF_LOCK	GPIO_13	ALT6
	SPDIF_OUT	GPIO_05	ALT4
	SPDIF_OUT	GPIO_11	ALT6
	SPDIF_SR_CLK	GPIO_03	ALT4
	SPDIF_SR_CLK	GPIO_09	ALT6
SRC	SRC_BOOT_MODE0	GPIO_SD_04	ALT6
	SRC_BOOT_MODE1	GPIO_SD_03	ALT6

Table continues on the next page...

Table 10-1. Muxing Options (continued)

Instance	Port	Pad	Mode
	SRC_BT_CFG0	GPIO_SD_02	ALT6
	SRC_BT_CFG1	GPIO_SD_01	ALT6
	SRC_BT_CFG2	GPIO_SD_00	ALT6
	SRC_BT_CFG3	GPIO_SD_13	ALT6
	SRC_POR_B	POR_B	ALT0
	SRC_RESET_B	ONOFF	ALT0
USB	USB_OTG1_ID	GPIO_13	ALT3
	USB_OTG1_ID	GPIO_AD_10	ALT6
	USB_OTG1_OC	GPIO_12	ALT3
	USB_OTG1_OC	GPIO_AD_01	ALT6
	USB_OTG1_PWR	GPIO_AD_00	ALT3
	USB_OTG1_PWR	GPIO_AD_12	ALT6
WDOG1	WDOG1_ANY	GPIO_01	ALT1
	WDOG1_ANY	GPIO_AD_01	ALT2
	WDOG1_B	GPIO_09	ALT1
	WDOG1_B	GPIO_AD_11	ALT6
	WDOG1_RST_B_DEB	GPIO_SD_11	ALT6
WDOG2	WDOG2_B	GPIO_02	ALT1
	WDOG2_B	GPIO_AD_02	ALT2
	WDOG2_RST_B_DEB	GPIO_SD_12	ALT6
XBAR1	XBAR1_INOUT02	GPIO_AD_14	ALT7
	XBAR1_INOUT03	GPIO_AD_07	ALT7
XTAL OSC	REF_CLK_24M	GPIO_AD_14	ALT6
	REF_CLK_32K	GPIO_AD_09	ALT6

1. GPIO_SD_14 pin is not applicable for LQFP80 package.

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.

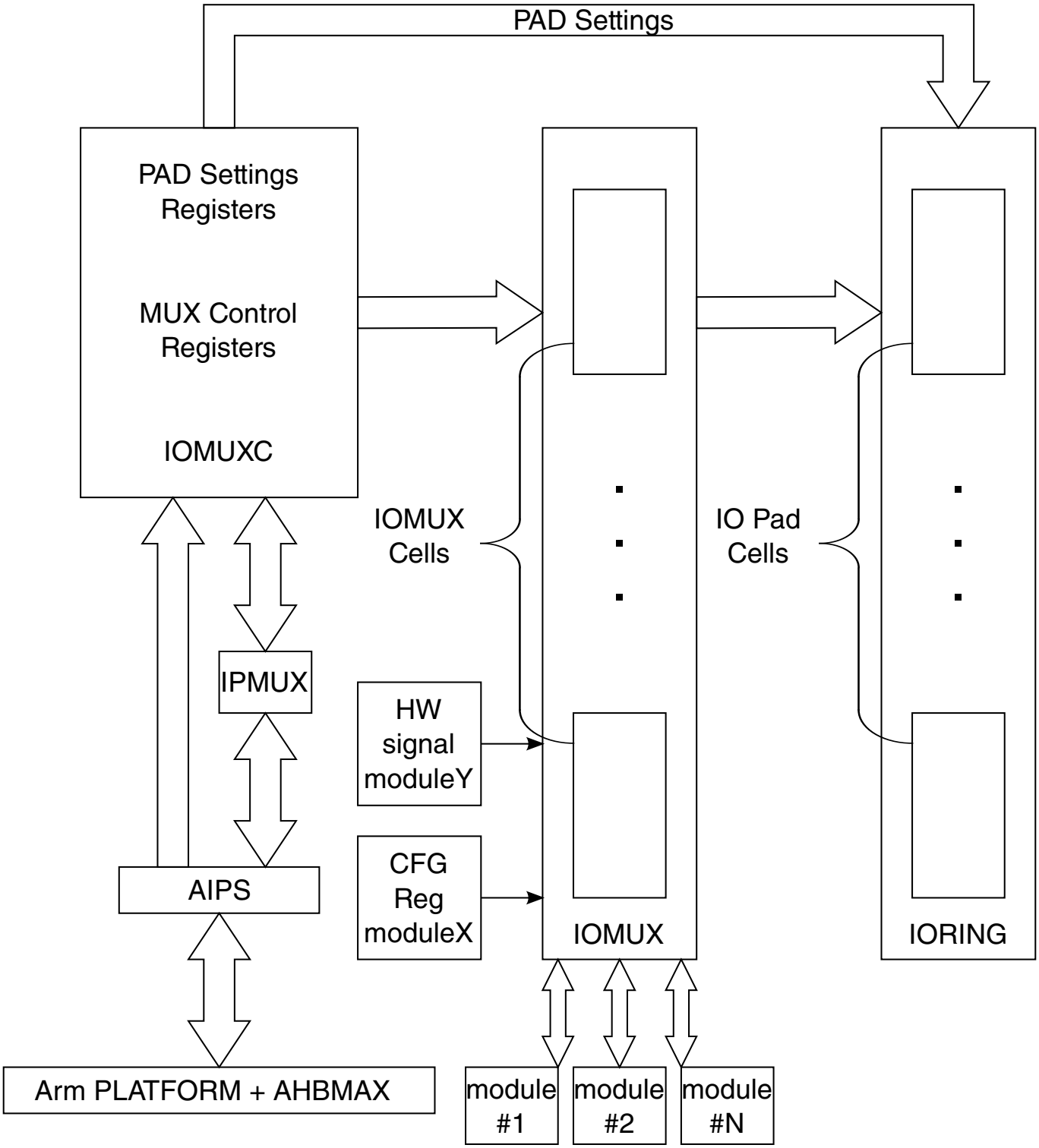


Figure 11-1. IOMUX SoC Level Block Diagram

11.1.1 Features

The IOMUXC features are:

- 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 feilds 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 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 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.

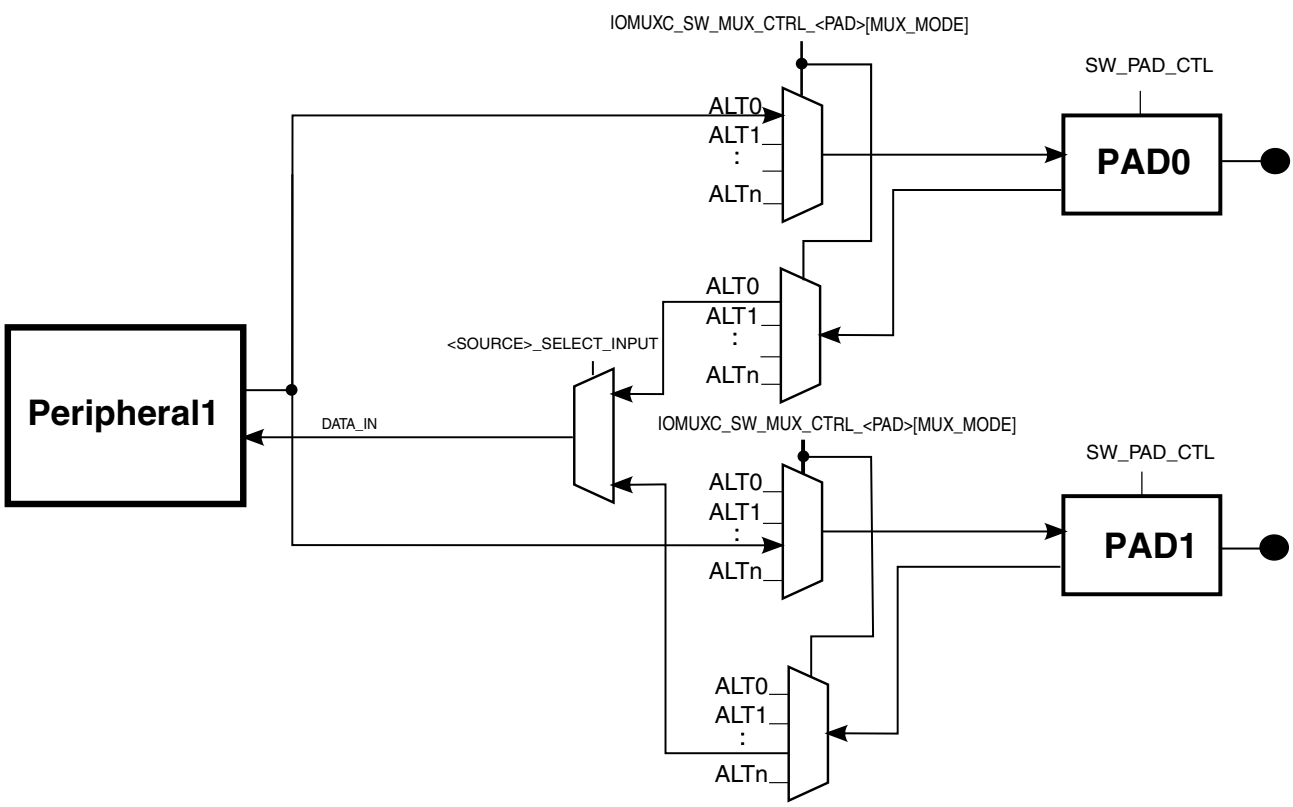


Figure 11-2. IOMUX Cell Block Diagram

11.3.1 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.

11.3.2 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.

11.3.3 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.

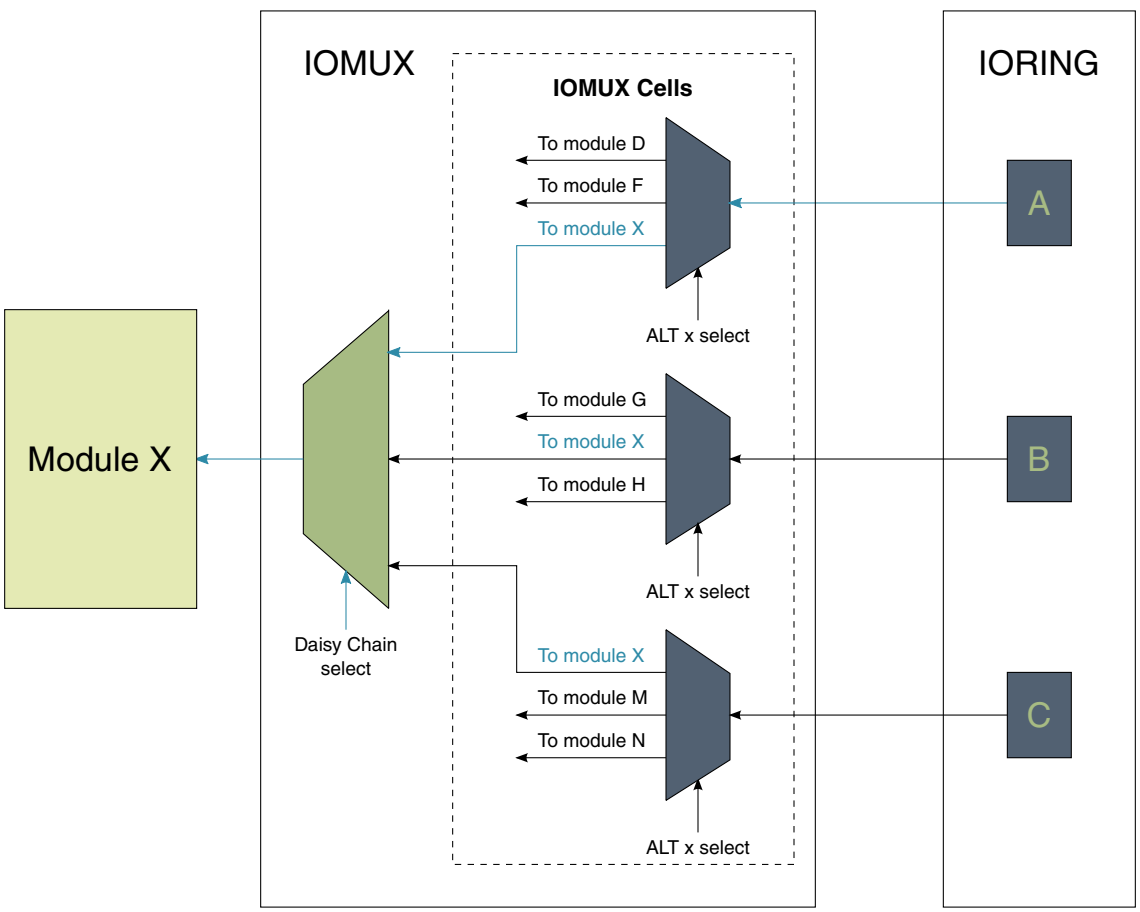


Figure 11-3. Daisy chain illustration

11.4 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	11.4.1/206
400A_C004	GPR1 General Purpose Register (IOMUXC_GPR_GPR1)	32	R/W	0000_0000h	11.4.2/206
400A_C008	GPR2 General Purpose Register (IOMUXC_GPR_GPR2)	32	R/W	0000_0000h	11.4.3/209
400A_C00C	GPR3 General Purpose Register (IOMUXC_GPR_GPR3)	32	R/W	0000_0FF0h	11.4.4/211
400A_C010	GPR4 General Purpose Register (IOMUXC_GPR_GPR4)	32	R/W	0000_0000h	11.4.5/212
400A_C014	GPR5 General Purpose Register (IOMUXC_GPR_GPR5)	32	R/W	0000_0000h	11.4.6/215
400A_C018	GPR6 General Purpose Register (IOMUXC_GPR_GPR6)	32	R/W	0000_0000h	11.4.7/217
400A_C01C	GPR7 General Purpose Register (IOMUXC_GPR_GPR7)	32	R/W	0000_0000h	11.4.8/218
400A_C020	GPR8 General Purpose Register (IOMUXC_GPR_GPR8)	32	R/W	0000_0000h	11.4.9/221
400A_C024	GPR9 General Purpose Register (IOMUXC_GPR_GPR9)	32	R/W	0000_0000h	11.4.10/223
400A_C028	GPR10 General Purpose Register (IOMUXC_GPR_GPR10)	32	R/W	0000_0007h	11.4.11/224
400A_C02C	GPR11 General Purpose Register (IOMUXC_GPR_GPR11)	32	R/W	0000_0000h	11.4.12/226
400A_C030	GPR12 General Purpose Register (IOMUXC_GPR_GPR12)	32	R/W	0000_0000h	11.4.13/228
400A_C034	GPR13 General Purpose Register (IOMUXC_GPR_GPR13)	32	R/W	0000_0000h	11.4.14/229
400A_C038	GPR14 General Purpose Register (IOMUXC_GPR_GPR14)	32	R/W	0088_0000h	11.4.15/230
400A_C03C	GPR15 General Purpose Register (IOMUXC_GPR_GPR15)	32	R/W	FFFF_FFFFh	11.4.16/231
400A_C040	GPR16 General Purpose Register (IOMUXC_GPR_GPR16)	32	R/W	0020_0003h	11.4.17/231
400A_C044	GPR17 General Purpose Register (IOMUXC_GPR_GPR17)	32	R/W	0000_0000h	11.4.18/232
400A_C048	GPR18 General Purpose Register (IOMUXC_GPR_GPR18)	32	R/W	0000_0000h	11.4.19/233
400A_C04C	GPR19 General Purpose Register (IOMUXC_GPR_GPR19)	32	R/W	0000_0000h	11.4.20/234
400A_C050	GPR20 General Purpose Register (IOMUXC_GPR_GPR20)	32	R/W	0000_0000h	11.4.21/235
400A_C054	GPR21 General Purpose Register (IOMUXC_GPR_GPR21)	32	R/W	0000_0000h	11.4.22/236
400A_C058	GPR22 General Purpose Register (IOMUXC_GPR_GPR22)	32	R/W	0000_0000h	11.4.23/237
400A_C05C	GPR23 General Purpose Register (IOMUXC_GPR_GPR23)	32	R/W	0000_0000h	11.4.24/238
400A_C060	GPR24 General Purpose Register (IOMUXC_GPR_GPR24)	32	R/W	0000_0000h	11.4.25/239
400A_C064	GPR25 General Purpose Register (IOMUXC_GPR_GPR25)	32	R/W	0000_0000h	11.4.26/240
400A_C068	GPR26 General Purpose Register (IOMUXC_GPR_GPR26)	32	R/W	0000_0000h	11.4.27/241
400A_C06C	GPR27 General Purpose Register (IOMUXC_GPR_GPR27)	32	R/W	0000_0000h	11.4.28/241
400A_C070	GPR28 General Purpose Register (IOMUXC_GPR_GPR28)	32	R/W	0000_0000h	11.4.29/242
400A_C074	GPR29 General Purpose Register (IOMUXC_GPR_GPR29)	32	R/W	0000_0000h	11.4.30/242

11.4.1 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	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_GPR0 field descriptions

Field	Description
-	Reserved This field is reserved. Reserved. General purpose bits to be used by SoC integration. Bit field type: DEFAULT

11.4.2 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							Reserved	EXC_MON	SAI3_MCLK_DIR	Reserved	SAI1_MCLK_DIR	Reserved	Reserved	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	Reserved	Reserved	GINT	SAI3_MCLK3_SEL	Reserved			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 -	This field is reserved. Reserved
22 EXC_MON	Exclusive monitor response select of illegal command This bit sets the bus response value when CM7 exclusive access fails. 0 OKAY response 1 SLVError response
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 -	This field is reserved. Reserved
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 -	This field is reserved. Reserved
17 -	This field is reserved. Reserved
16 -	This field is reserved. Reserved
15 -	This field is reserved. Reserved
14 -	This field is reserved. Reserved
13 -	This field is reserved. Reserved
12 GINT	Global Interrupt This is the global interrupt bit. It is connected to Arm M7 IRQ#41. 0 Global interrupt request is not asserted. 1 Global interrupt request is asserted.

Table continues on the next page...

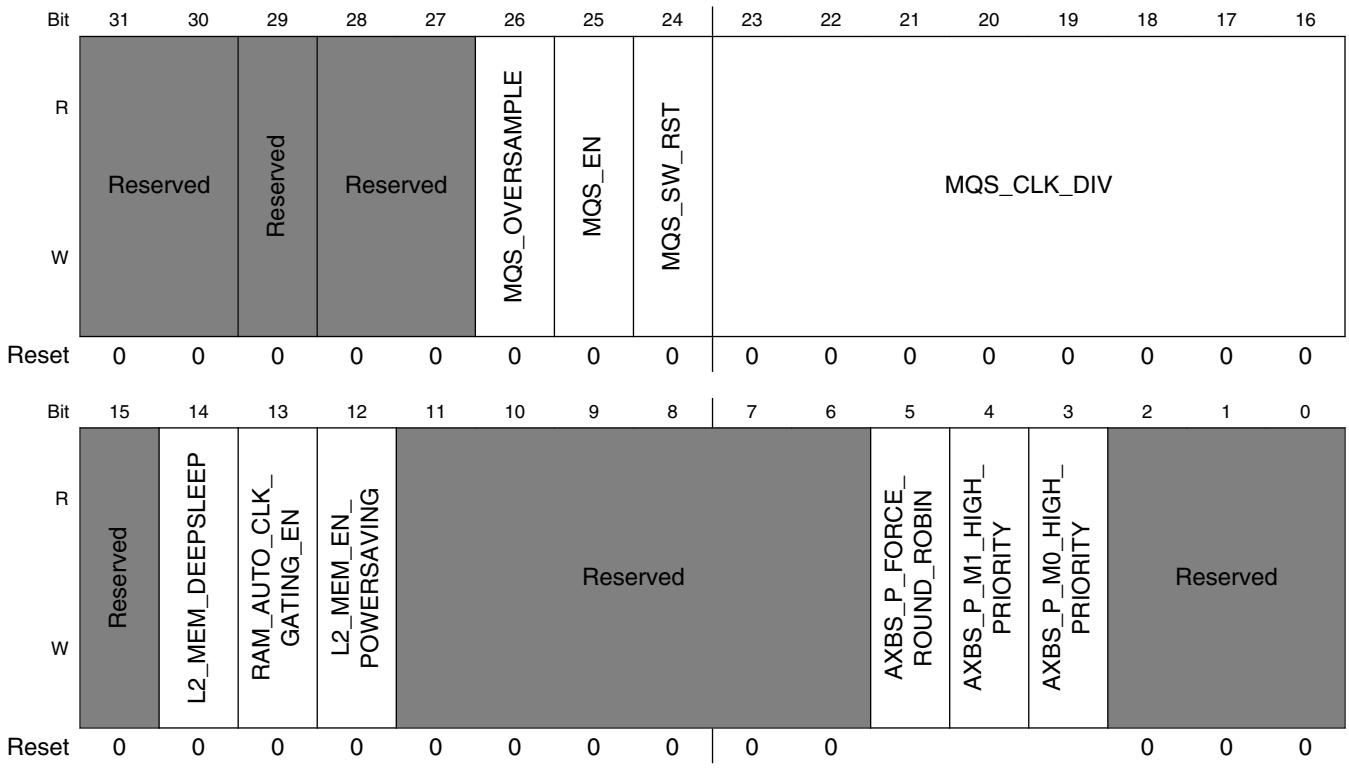
IOMUXC_GPR_GPR1 field descriptions (continued)

Field	Description
11–10 SAI3_MCLK3_SEL	SAI3 MCLK3 source select See the Audio subsystem clocking diagram in the Audio Overview Chapter for more information. 00 ccm.spdif0_clk_root 01 SPDIF_EXT_CLK 10 spdif.spdif_srclk 11 spdif.spdif_outclock
9–8 -	This field is reserved. Reserved
7–6 SAI1_MCLK3_SEL	SAI1 MCLK3 source select See the Audio subsystem clocking diagram in the Audio Overview Chapter for more information. 00 ccm.spdif0_clk_root 01 SPDIF_EXT_CLK 10 spdif.spdif_srclk 11 spdif.spdif_outclock
5–3 SAI1_MCLK2_SEL	SAI1 MCLK2 source select See the Audio subsystem clocking diagram in the Audio Overview Chapter for more information. 000 ccm.ssi1_clk_root 001 Reserved 010 ccm.ssi3_clk_root 011 iomux.sai1_ipg_clk_sai_mclk 100 Reserved 101 iomux.sai3_ipg_clk_sai_mclk 110 Reserved 111 Reserved
SAI1_MCLK1_SEL	SAI1 MCLK1 source select See the Audio subsystem clocking diagram in the Audio Overview Chapter for more information. 000 ccm.ssi1_clk_root 001 Reserved 010 ccm.ssi3_clk_root 011 iomux.sai1_ipg_clk_sai_mclk 100 Reserved 101 iomux.sai3_ipg_clk_sai_mclk 110 Reserved 111 Reserved

11.4.3 GPR2 General Purpose Register (IOMUXC_GPR_GPR2)

GPR Register

Address: 400A_C000h base + 8h offset = 400A_C008h



IOMUXC_GPR_GPR2 field descriptions

Field	Description
31–30 -	This field is reserved. Reserved
29 -	This field is reserved. Reserved
28–27 -	This field is reserved. Reserved
26 MQS_ OVERSAMPLE	Medium Quality Sound (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

Table continues on the next page...

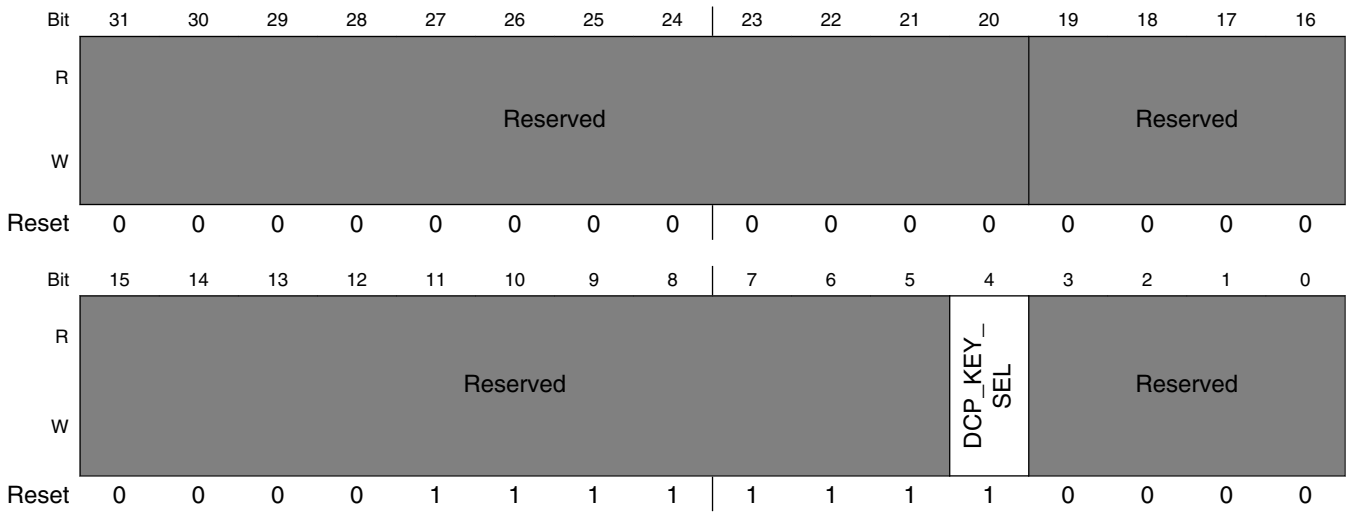
IOMUXC_GPR_GPR2 field descriptions (continued)

Field	Description
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, *n* is from 0 to 255. 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	This bit controls how memory (OCRAM) enters 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 (OCRAM in normal mode) 1 Force memory into deep sleep mode (OCRAM in power saving 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 This bit controls how OCRAM enters power saving mode. 0 Enters power saving mode only when chip is in SUSPEND mode 1 Controlled by L2_MEM_DEEPSLEEP bitfield
11–6 -	This field is reserved. Reserved
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
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
-	This field is reserved. Reserved

11.4.4 GPR3 General Purpose Register (IOMUXC_GPR_GPR3)

GPR Register

Address: 400A_C000h base + Ch offset = 400A_C00Ch



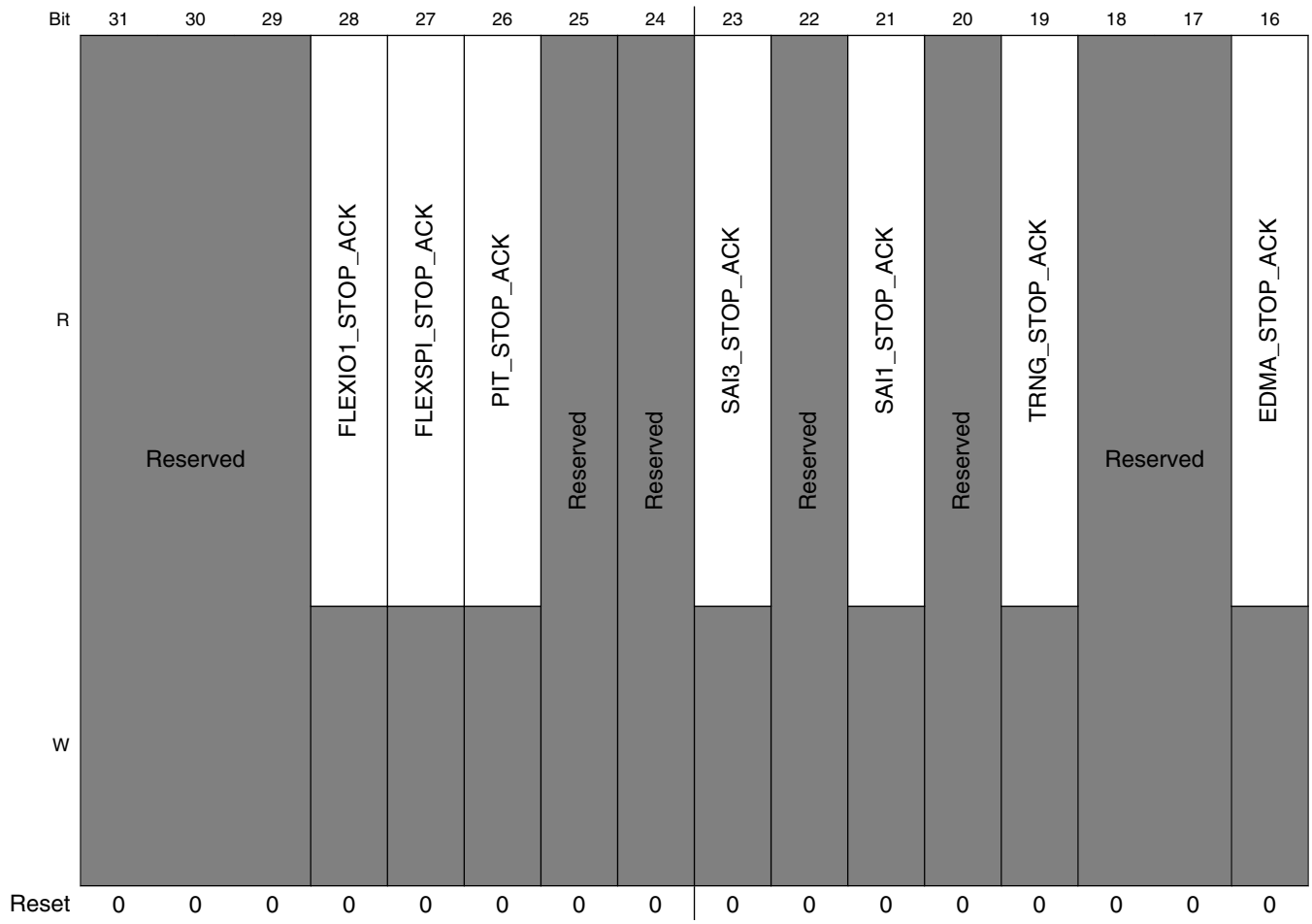
IOMUXC_GPR_GPR3 field descriptions

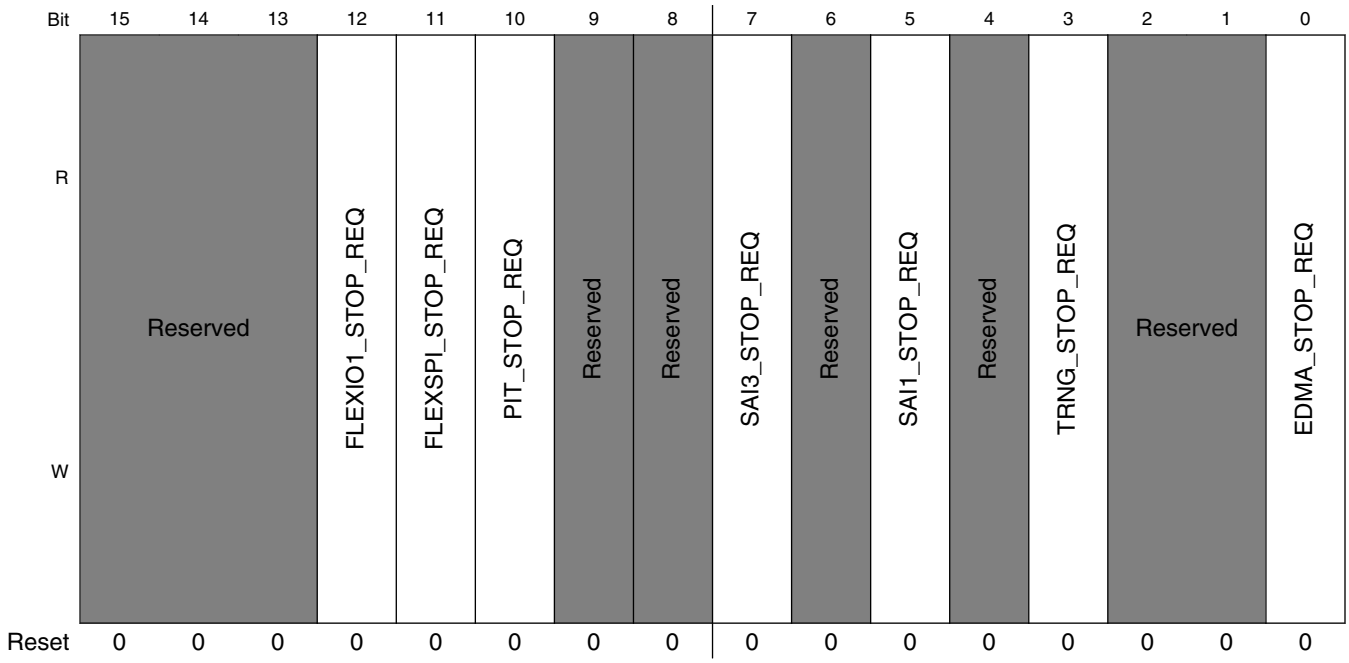
Field	Description
31–20 -	This field is reserved. Reserved
19–16 -	This field is reserved. Reserved
15–5 -	This field is reserved. Reserved
4 DCP_KEY_SEL	Select 128-bit DCP key from 256-bit key from SNVS Master Key It is necessary to reset the DCP after changing this bit for the change to take effect. Note that IOMUXC_GPR10[DCPKEY_OCOTP_OR_KEYMUX] also affects the key sent to the DCP. 0 Select [127:0] from SNVS Master Key as DCP key 1 Select [255:128] from SNVS Master Key as DCP key
-	This field is reserved. Reserved

11.4.5 GPR4 General Purpose Register (IOMUXC_GPR_GPR4)

GPR Register

Address: 400A_C000h base + 10h offset = 400A_C010h





IOMUXC_GPR_GPR4 field descriptions

Field	Description
31–29 -	This field is reserved. Reserved
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 -	This field is reserved. Reserved
24 -	This field is reserved. Reserved
23 SAI3_STOP_ACK	SAI3 stop acknowledge 0 SAI3 stop acknowledge is not asserted 1 SAI3 stop acknowledge is asserted
22 -	This field is reserved. Reserved
21 SAI1_STOP_ACK	SAI1 stop acknowledge

Table continues on the next page...

IOMUXC_GPR_GPR4 field descriptions (continued)

Field	Description
	0 SAI1 stop acknowledge is not asserted 1 SAI1 stop acknowledge is asserted
20 -	This field is reserved. Reserved
19 TRNG_STOP_ACK	TRNG stop acknowledge 0 TRNG stop acknowledge is not asserted 1 TRNG stop acknowledge is asserted
18–17 -	This field is reserved. Reserved
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).
15–13 -	This field is reserved. Reserved
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 -	This field is reserved. Reserved
8 -	This field is reserved. Reserved
7 SAI3_STOP_REQ	SAI3 stop request. 0 stop request off 1 stop request on
6 -	This field is reserved. Reserved
5 SAI1_STOP_REQ	SAI1 stop request. 0 stop request off 1 stop request on
4 -	This field is reserved. Reserved
3 TRNG_STOP_REQ	TRNG stop request. 0 stop request off 1 stop request on

Table continues on the next page...

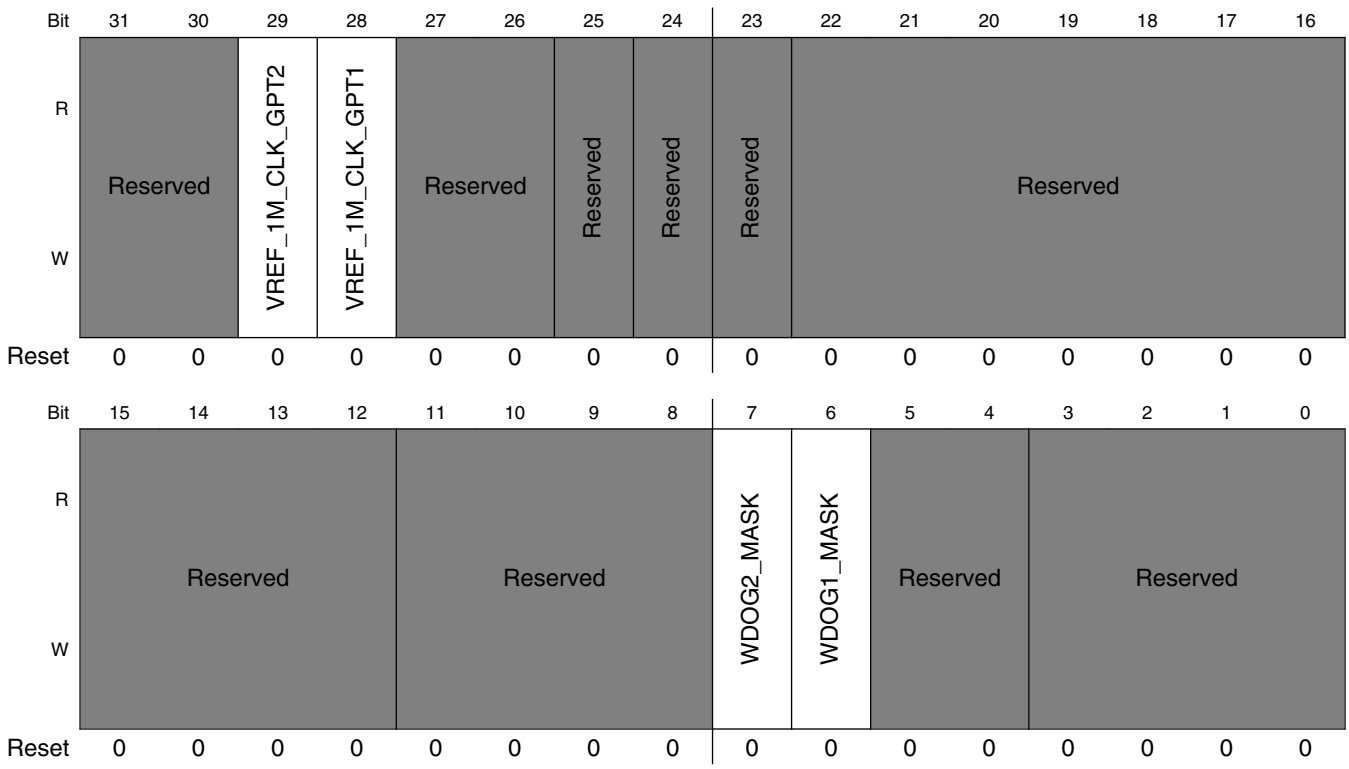
IOMUXC_GPR_GPR4 field descriptions (continued)

Field	Description
2–1 -	This field is reserved. Reserved
0 EDMA_STOP_REQ	EDMA stop request. 0 stop request off 1 stop request on

11.4.6 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

Table continues on the next page...

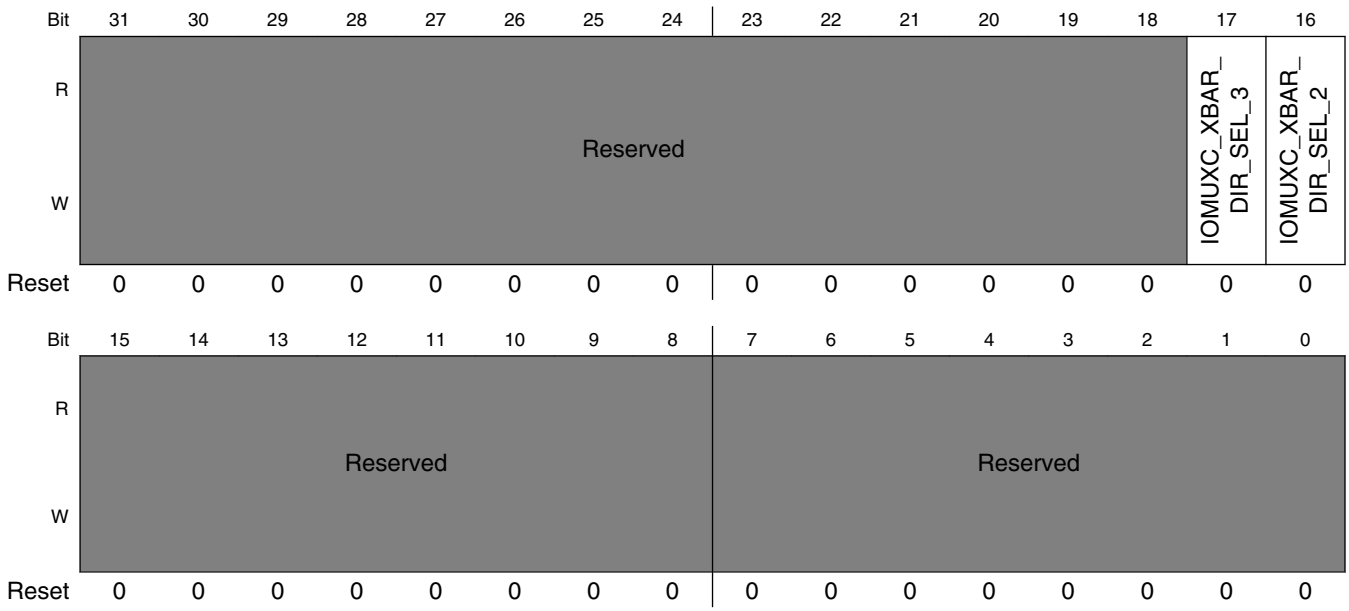
IOMUXC_GPR_GPR5 field descriptions (continued)

Field	Description
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
27–26 -	This field is reserved. Reserved
25 -	This field is reserved. Reserved
24 -	This field is reserved. Reserved
23 -	This field is reserved. Reserved
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

11.4.7 GPR6 General Purpose Register (IOMUXC_GPR_GPR6)

GPR Register

Address: 400A_C000h base + 18h offset = 400A_C018h



IOMUXC_GPR_GPR6 field descriptions

Field	Description
31–18 -	This field is reserved. Reserved
17 IOMUXC_XBAR_DIR_SEL_3	IOMUXC XBAR_INOUT3 function direction select 0 XBAR_INOUT as input 1 XBAR_INOUT as output
16 IOMUXC_XBAR_DIR_SEL_2	IOMUXC XBAR_INOUT2 function direction select 0 XBAR_INOUT as input 1 XBAR_INOUT as output
15–8 -	This field is reserved. Reserved
-	This field is reserved. Reserved

11.4.8 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	Reserved				LPUART4_STOP_ACK	LPUART3_STOP_ACK	LPUART2_STOP_ACK	LPUART1_STOP_ACK	Reserved		LPSPI2_STOP_ACK	LPSPI1_STOP_ACK	Reserved		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	Reserved				LPUART4_STOP_REQ	LPUART3_STOP_REQ	LPUART2_STOP_REQ	LPUART1_STOP_REQ	Reserved		LPSPI2_STOP_REQ	LPSPI1_STOP_REQ	Reserved		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
31–28 -	This field is reserved. Reserved
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
23–22 -	This field is reserved. Reserved
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
19–18 -	This field is reserved. Reserved
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)
15–12 -	This field is reserved. Reserved
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

Table continues on the next page...

IOMUXC_GPR_GPR7 field descriptions (continued)

Field	Description
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
7–6 -	This field is reserved. Reserved
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
3–2 -	This field is reserved. Reserved
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

11.4.9 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	Reserved								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	Reserved				LPSPI2_IPG_DOZE	LPSPI2_IPG_STOP_MODE	LPSPI1_IPG_DOZE	LPSPI1_IPG_STOP_MODE	Reserved				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
31–24 -	This field is reserved. Reserved
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

Table continues on the next page...

IOMUXC_GPR_GPR8 field descriptions (continued)

Field	Description
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
15–12 -	This field is reserved. Reserved
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
7–4 -	This field is reserved. Reserved
3 LPI2C2_IPG_DOZE	LPI2C2 ipg_doze mode 0 not in doze mode 1 in doze mode
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

Table continues on the next page...

IOMUXC_GPR_GPR8 field descriptions (continued)

Field	Description
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

11.4.10 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

11.4.11 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	Reserved		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	Reserved		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
31–30 -	This field is reserved. Reserved
29–25 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

Table continues on the next page...

IOMUXC_GPR_GPR10 field descriptions (continued)

Field	Description
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)
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)
15–14 -	This field is reserved. Reserved
13–9 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).

Table continues on the next page...

IOMUXC_GPR_GPR10 field descriptions (continued)

Field	Description
0 NIDEN	Arm non-secure (non-invasive) debug enable 0 Debug turned off. 1 Debug enabled (default).

11.4.12 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				Reserved				LOCK_M7_APC_AC_R3_CTRL		LOCK_M7_APC_AC_R2_CTRL		LOCK_M7_APC_AC_R1_CTRL		LOCK_M7_APC_AC_R0_CTRL	
Reset	0	0	0	0	0	0	0	0	0	0	0	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				Reserved				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
31–28 -	This field is reserved. Reserved
27–24 -	This field is reserved. Reserved
23–22 LOCK_M7_APC_AC_R3_CTRL	Lock M7_APC_AC_R3_CTRL 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)
21–20 LOCK_M7_APC_AC_R2_CTRL	Lock M7_APC_AC_R2_CTRL 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)
19–18 LOCK_M7_APC_AC_R1_CTRL	Lock M7_APC_AC_R1_CTRL 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)
17–16 LOCK_M7_APC_AC_R0_CTRL	Lock M7_APC_AC_R0_CTRL 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)

Table continues on the next page...

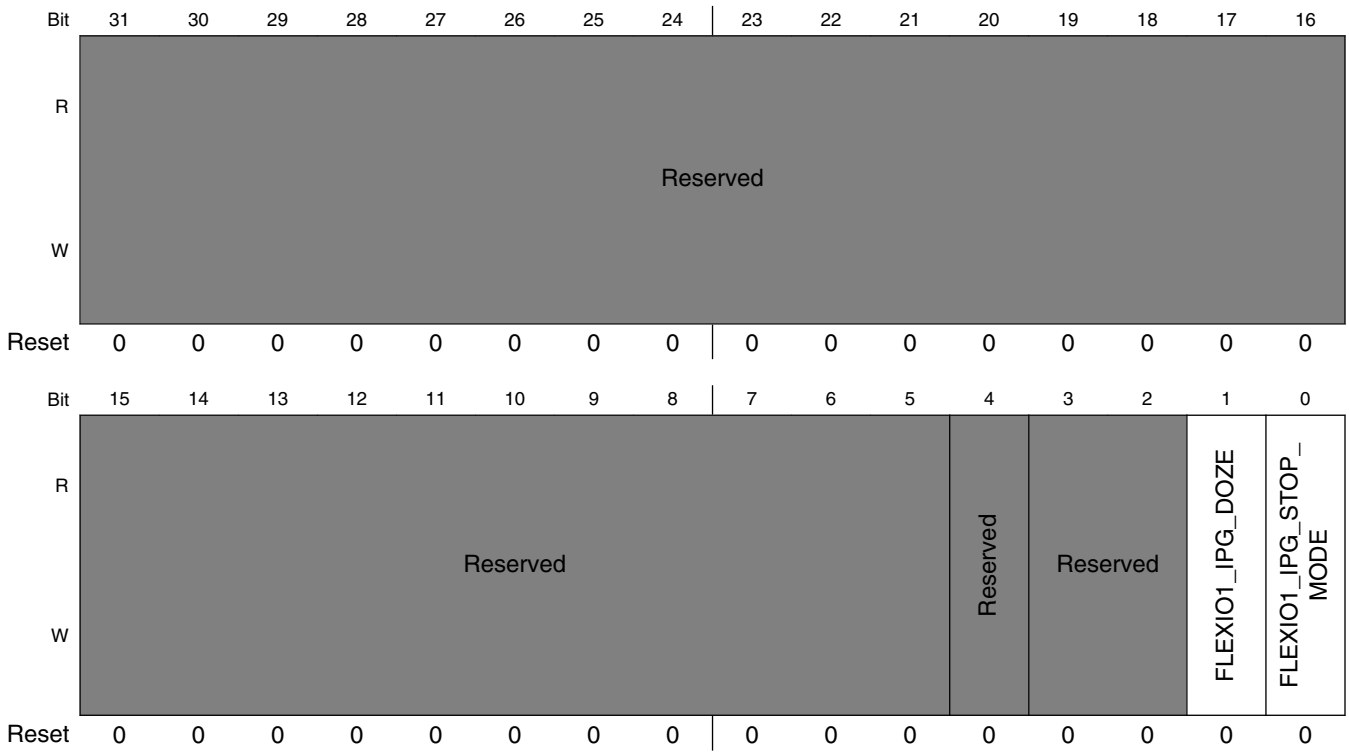
IOMUXC_GPR_GPR11 field descriptions (continued)

Field	Description
15–12 -	This field is reserved. Reserved
11–8 -	This field is reserved. Reserved
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 10 Reserved 11 Reserved

11.4.13 GPR12 General Purpose Register (IOMUXC_GPR_GPR12)

GPR Register

Address: 400A_C000h base + 30h offset = 400A_C030h



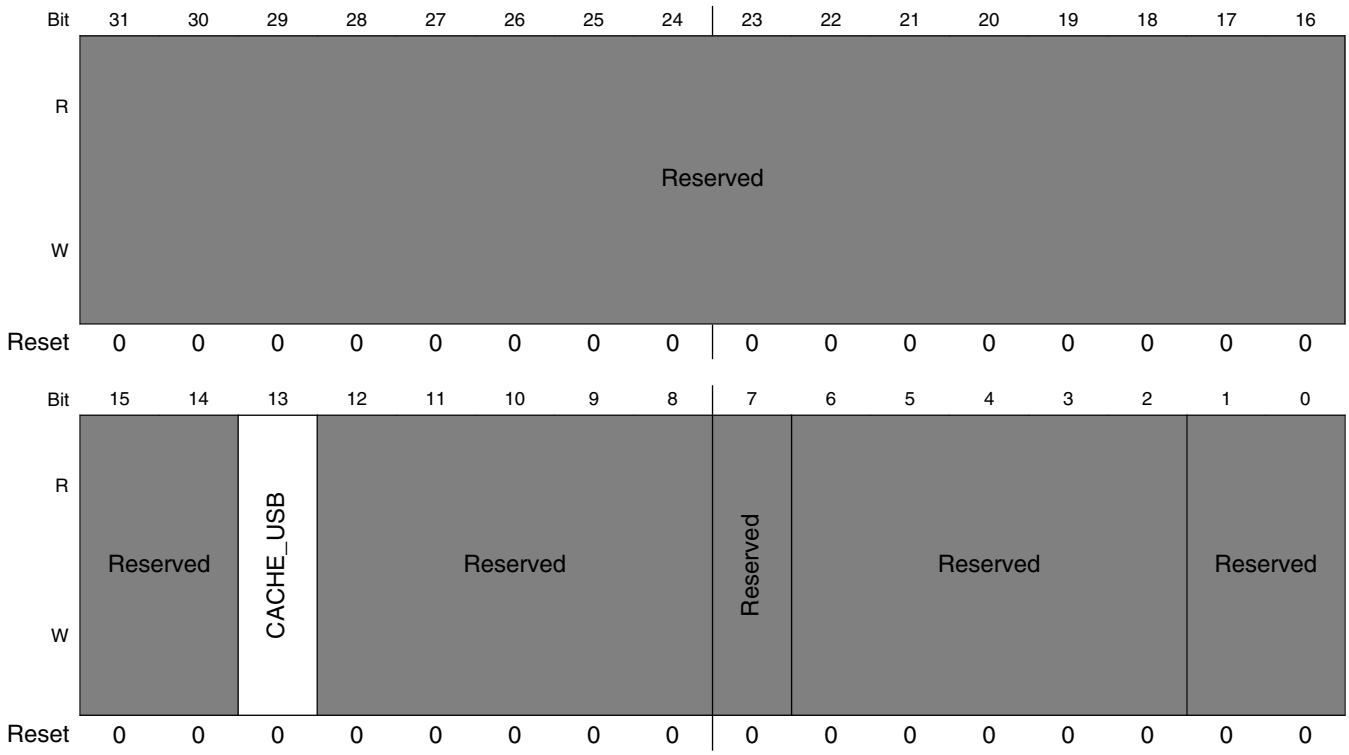
IOMUXC_GPR_GPR12 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 -	This field is reserved. Reserved
3–2 -	This field is reserved. Reserved
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.

11.4.14 GPR13 General Purpose Register (IOMUXC_GPR_GPR13)

GPR Register

Address: 400A_C000h base + 34h offset = 400A_C034h



IOMUXC_GPR_GPR13 field descriptions

Field	Description
31–14 -	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 -	This field is reserved. Reserved
6–2 -	This field is reserved. Reserved
-	This field is reserved. Reserved

11.4.15 GPR14 General Purpose Register (IOMUXC_GPR_GPR14)

GPR Register

Address: 400A_C000h base + 38h offset = 400A_C038h

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								CM7_ CFGDTCMSZ				CM7_ CFGITCMSZ				Reserved				Reserved											
Reset	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

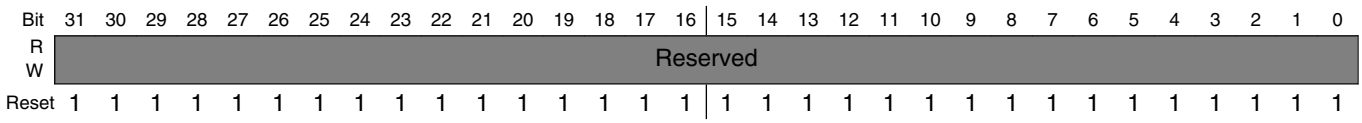
IOMUXC_GPR_GPR14 field descriptions

Field	Description
31–24 -	This field is reserved. Reserved
23–20 CM7_ CFGDTCMSZ	DTCM total size configuration 0000 0 KB (No DTCM) 0011 4 KB 0100 8 KB 0101 16 KB 0110 32 KB 0111 64 KB 1000 128 KB other reserved
19–16 CM7_ CFGITCMSZ	ITCM total size configuration 0000 0 KB (No ITCM) 0011 4 KB 0100 8 KB 0101 16 KB 0110 32 KB 0111 64 KB 1000 128 KB other reserved
15–12 -	This field is reserved. Reserved
-	This field is reserved. Reserved

11.4.16 GPR15 General Purpose Register (IOMUXC_GPR_GPR15)

GPR Register

Address: 400A_C000h base + 3Ch offset = 400A_C03Ch



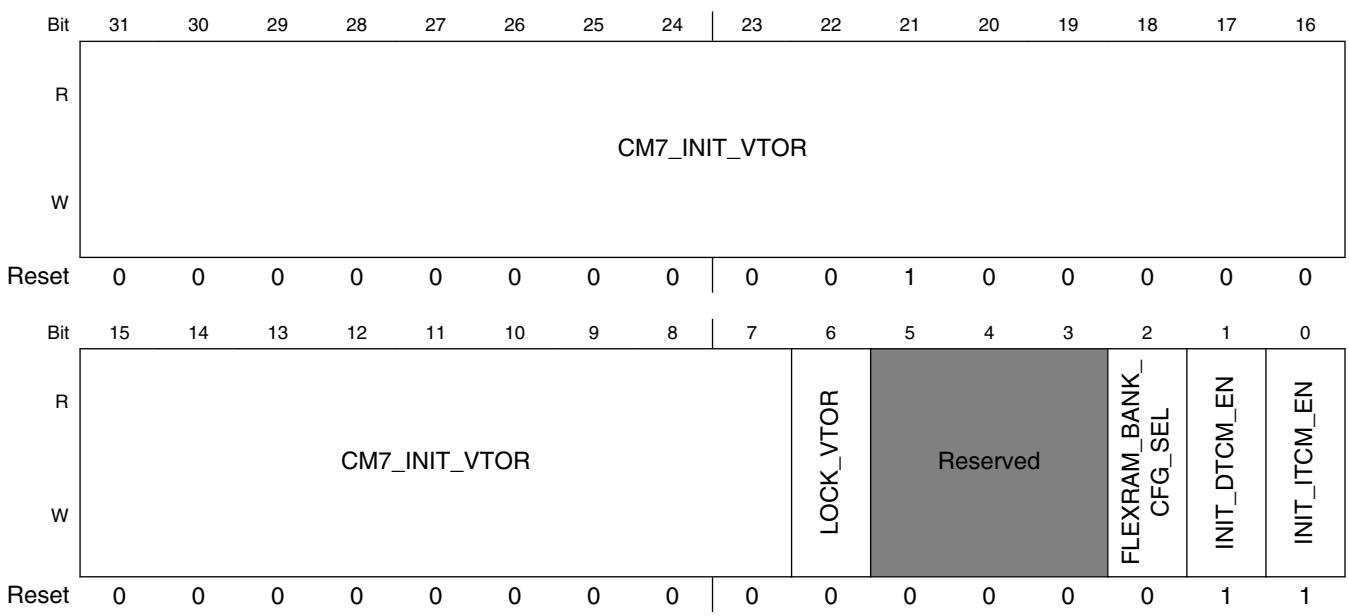
IOMUXC_GPR_GPR15 field descriptions

Field	Description
-	Reserved This field is reserved. General purpose bits to be used by SoC integration. Bit field type: DEFAULT

11.4.17 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 LOCK_VTOR	Lock CM7_INIT_VTOR field for changes. This is a sticky field: once set, it cannot be cleared (only by reset). 0 CM7_INIT_VTOR field is not locked. 1 CM7_INIT_VTOR field is locked (read access only).
5–3 -	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
1 INIT_DTCM_EN	DTCM enable initialization out of reset. **NOTE:** When a TCM is enabled, the corresponding CFG*TCMSZ register must NOT be set to 0'b0000 0 DTCM is disabled 1 DTCM is enabled
0 INIT_ITCM_EN	ITCM enable initialization out of reset. **NOTE:** When a TCM is enabled, the corresponding CFG*TCMSZ register must NOT be set to 0'b0000 0 ITCM is disabled 1 ITCM is enabled

11.4.18 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	Reserved																								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
31–8 -	This field is reserved. Reserved
FLEXRAM_ BANK_CFG	FlexRAM bank config value GPR_FLEXRAM_BANK_CFG[2*n*+1 : 2*n*], where n = 0, 1, ..., 3 • 00: RAM bank *n* is not used • 01: RAM bank *n* is OCRAM

Table continues on the next page...

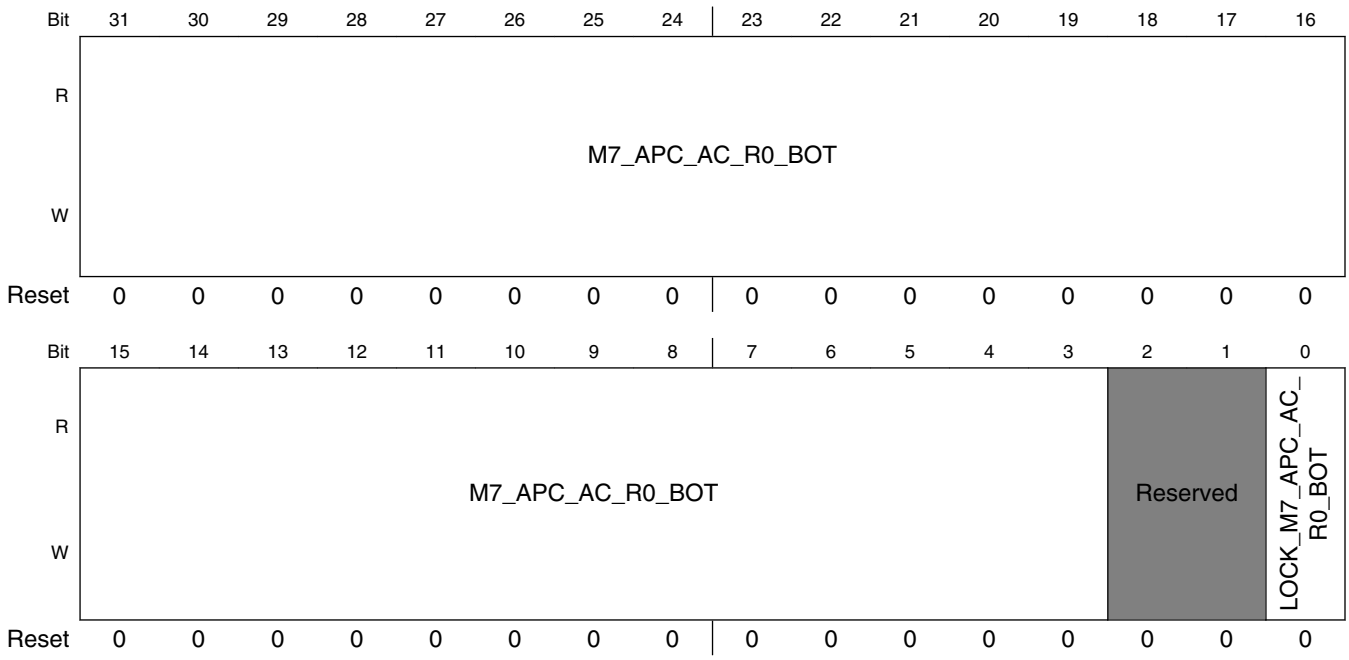
IOMUXC_GPR_GPR17 field descriptions (continued)

Field	Description
	• 10: RAM bank *n* is DTCM • 11: RAM bank *n* is ITCM

11.4.19 GPR18 General Purpose Register (IOMUXC_GPR_GPR18)

GPR Register

Address: 400A_C000h base + 48h offset = 400A_C048h



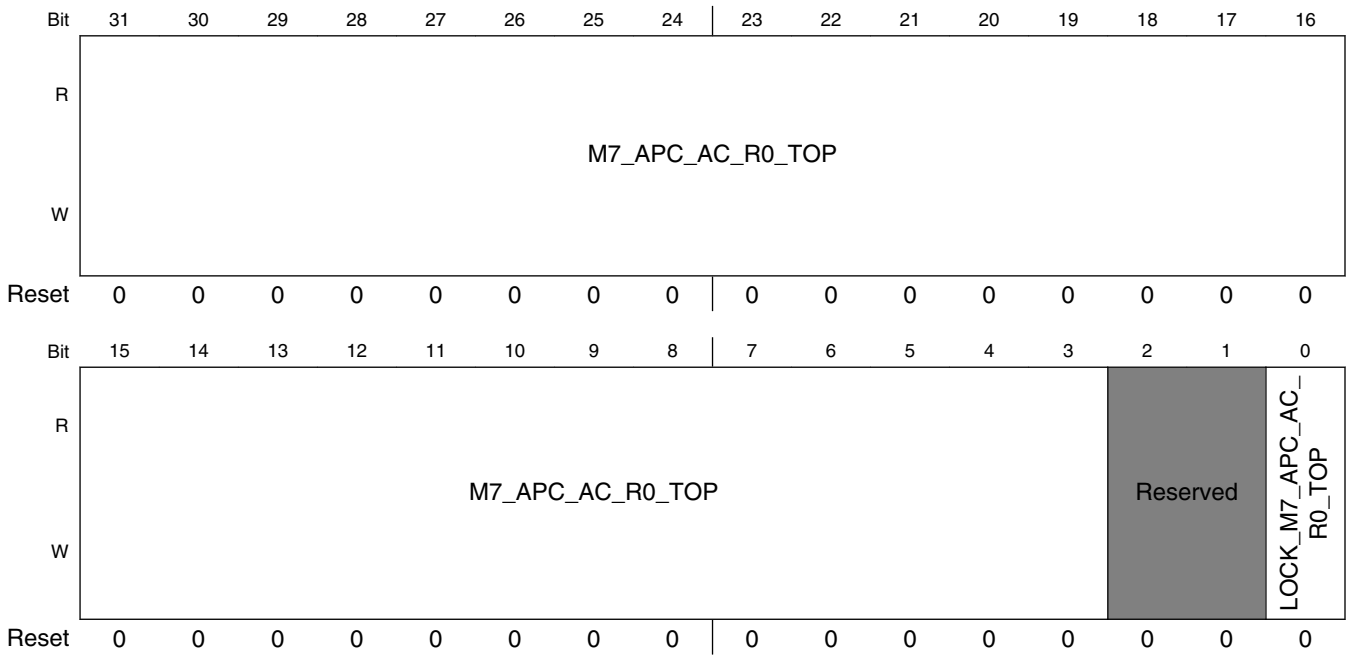
IOMUXC_GPR_GPR18 field descriptions

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)

11.4.20 GPR19 General Purpose Register (IOMUXC_GPR_GPR19)

GPR Register

Address: 400A_C000h base + 4Ch offset = 400A_C04Ch



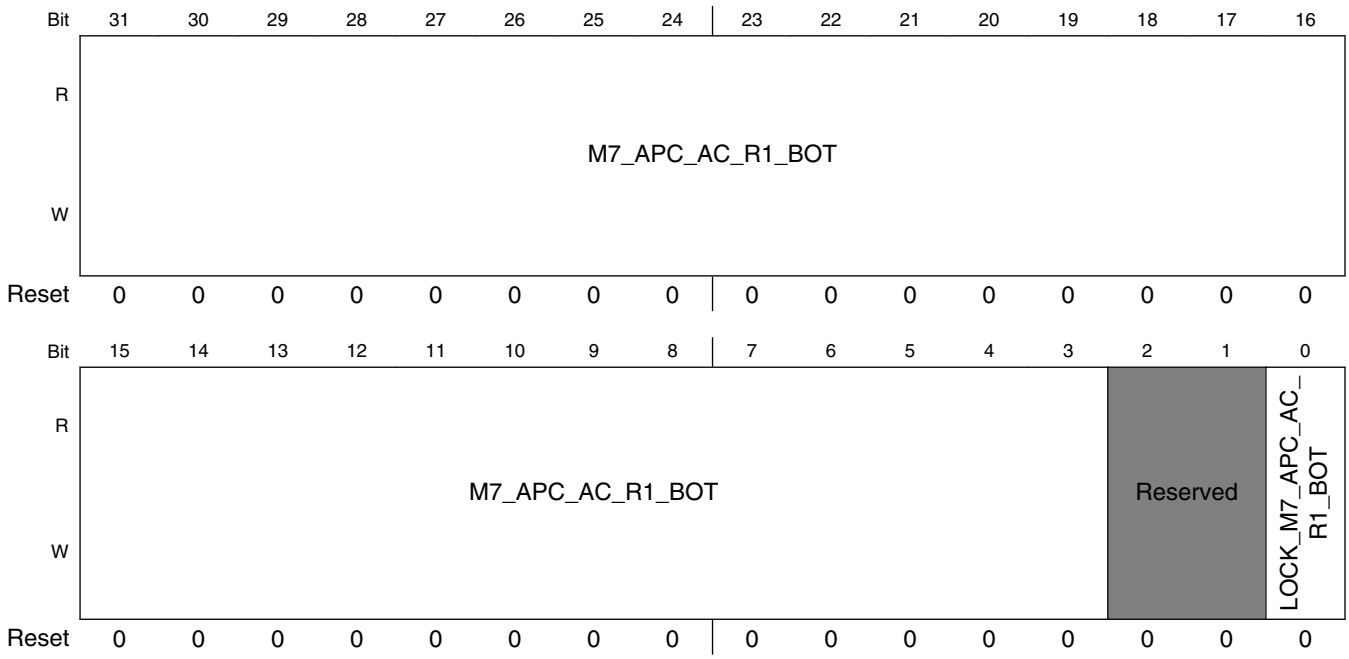
IOMUXC_GPR_GPR19 field descriptions

Field	Description
31–3 M7_APC_AC_R0_TOP	APC start address of memory region-0
2–1 -	This field is reserved. Reserved
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)

11.4.21 GPR20 General Purpose Register (IOMUXC_GPR_GPR20)

GPR Register

Address: 400A_C000h base + 50h offset = 400A_C050h



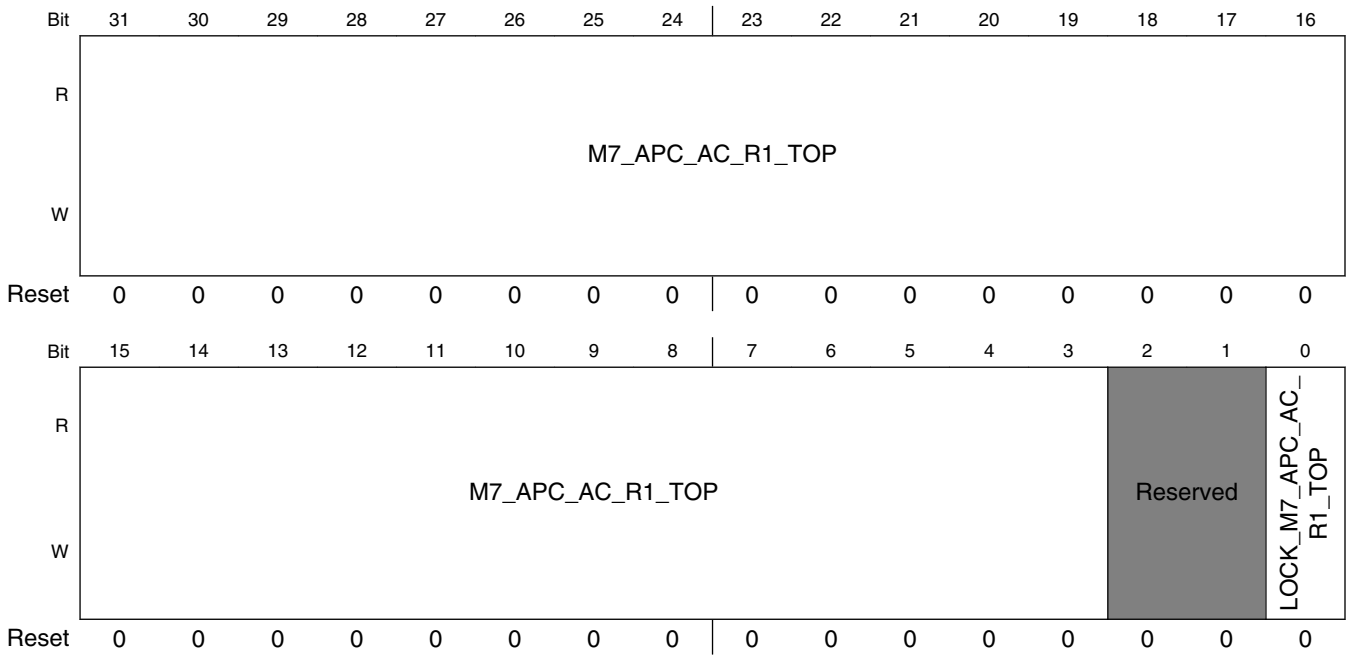
IOMUXC_GPR_GPR20 field descriptions

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)

11.4.22 GPR21 General Purpose Register (IOMUXC_GPR_GPR21)

GPR Register

Address: 400A_C000h base + 54h offset = 400A_C054h



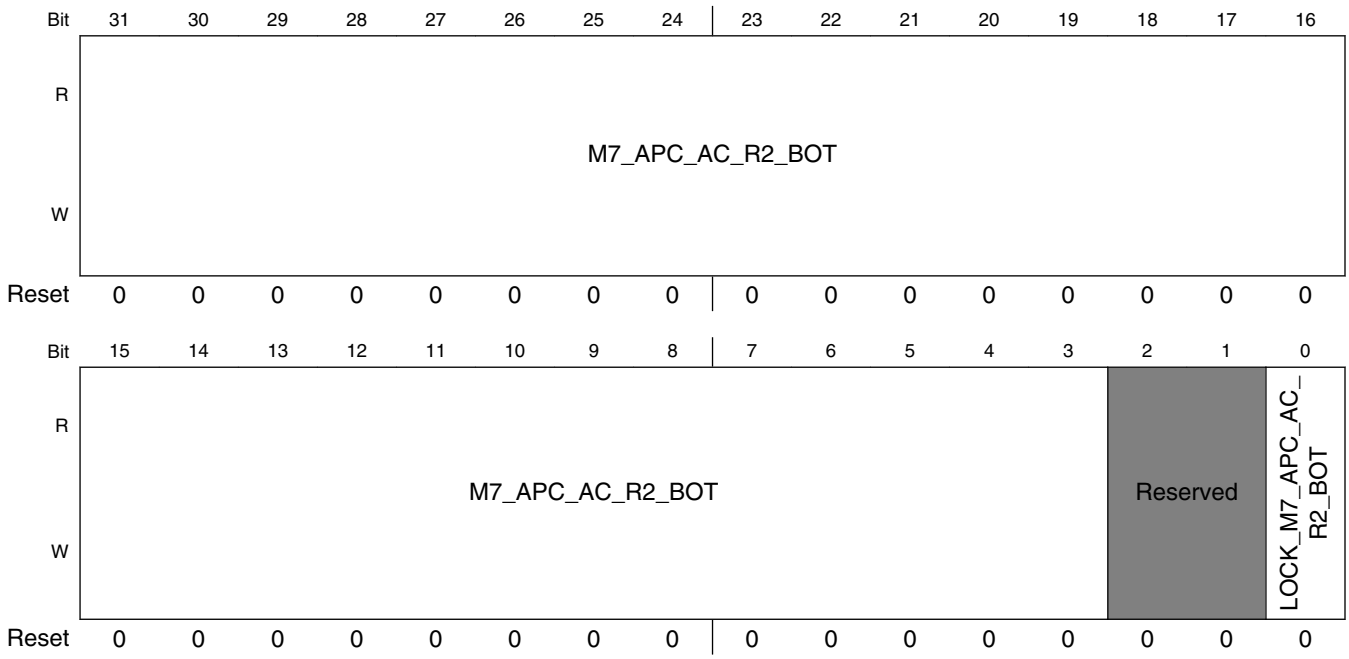
IOMUXC_GPR_GPR21 field descriptions

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)

11.4.23 GPR22 General Purpose Register (IOMUXC_GPR_GPR22)

GPR Register

Address: 400A_C000h base + 58h offset = 400A_C058h



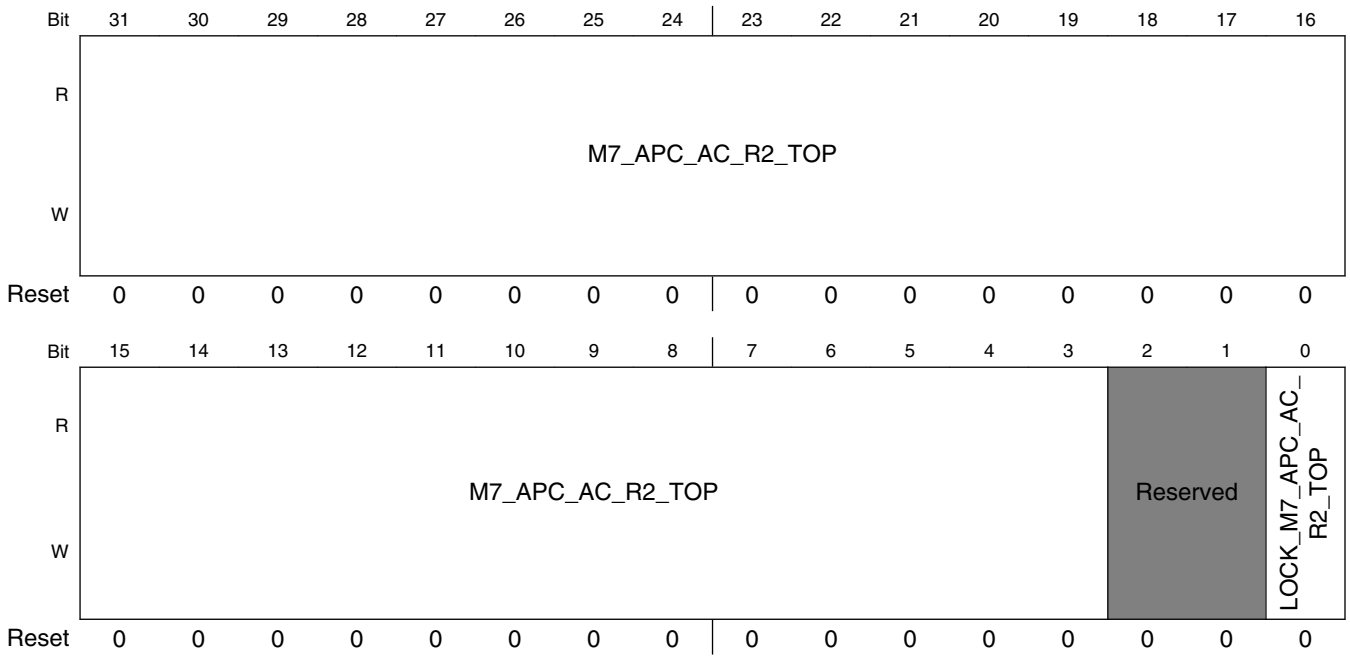
IOMUXC_GPR_GPR22 field descriptions

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)

11.4.24 GPR23 General Purpose Register (IOMUXC_GPR_GPR23)

GPR Register

Address: 400A_C000h base + 5Ch offset = 400A_C05Ch



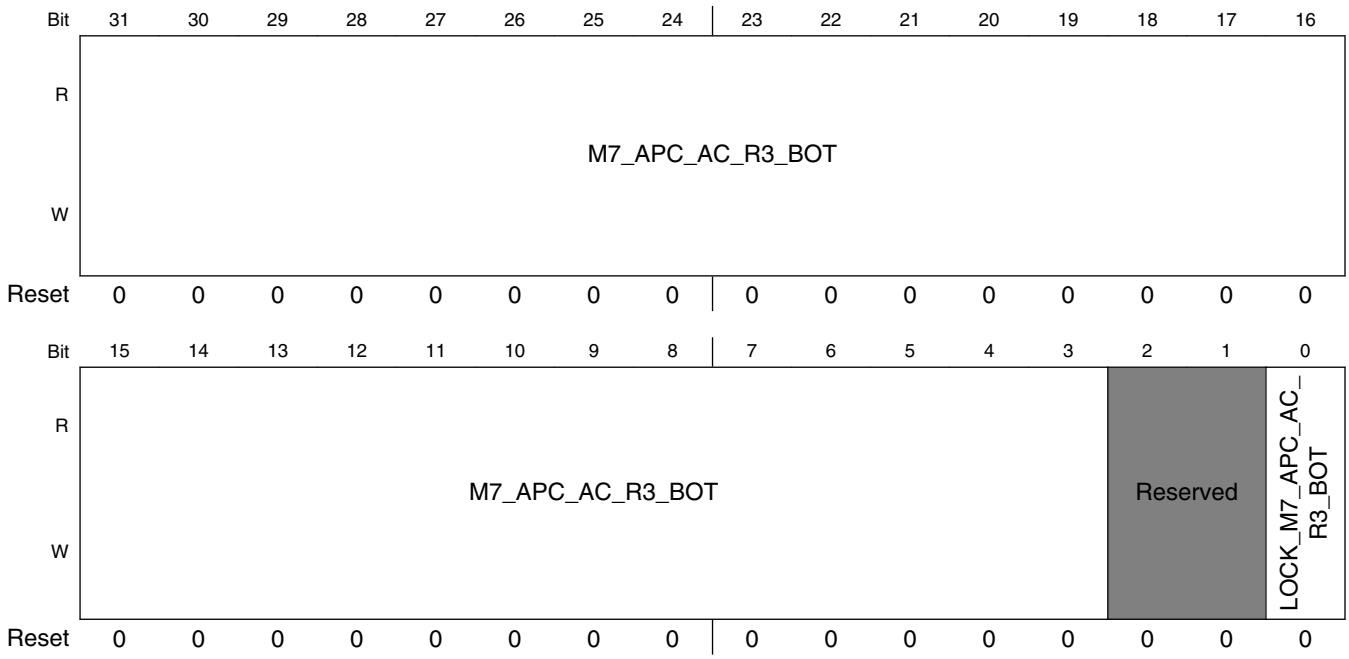
IOMUXC_GPR_GPR23 field descriptions

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)

11.4.25 GPR24 General Purpose Register (IOMUXC_GPR_GPR24)

GPR Register

Address: 400A_C000h base + 60h offset = 400A_C060h



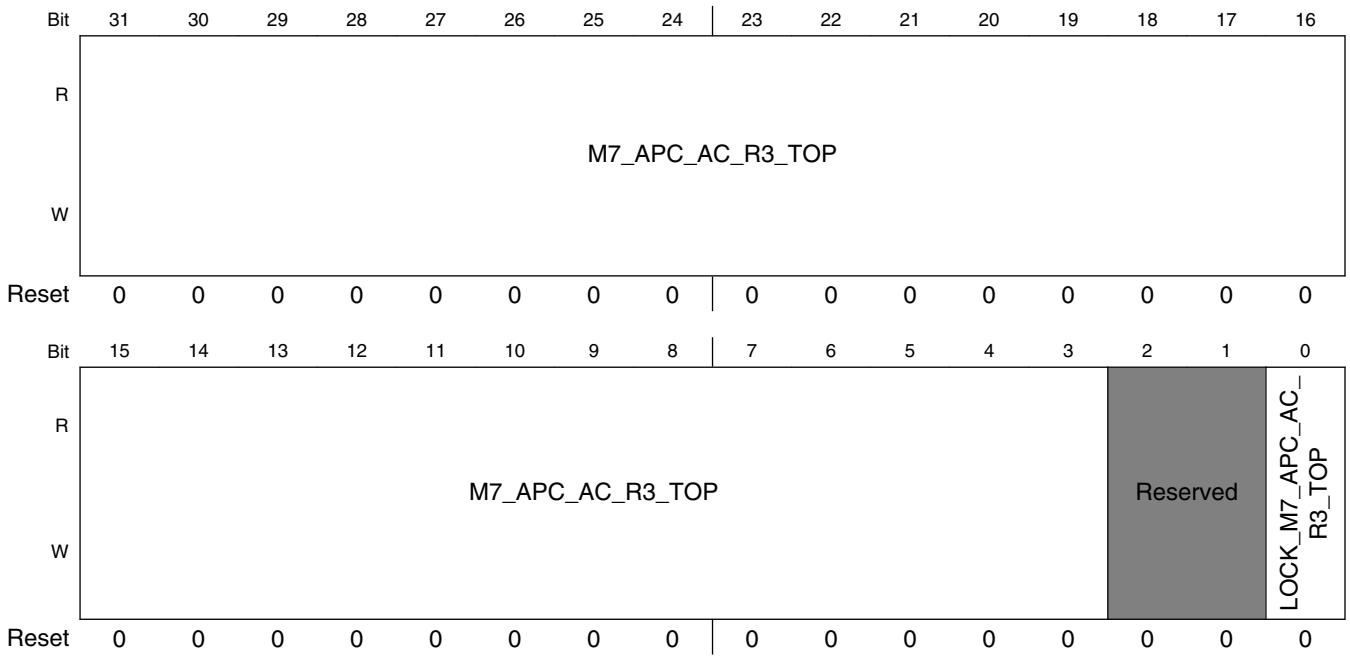
IOMUXC_GPR_GPR24 field descriptions

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)

11.4.26 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)

11.4.27 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_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_SEL	Select GPIO1 or GPIO2

11.4.28 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	FLEXSPI_REMAP_ADDR_START																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_GPR27 field descriptions

Field	Description
31–12 FLEXSPI_ REMAP_ADDR_ START	Start address of flexspi1
-	This field is reserved. Reserved

11.4.29 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	FLEXSPI_REMAP_ADDR_END																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_GPR28 field descriptions

Field	Description
31–12 FLEXSPI_ REMAP_ADDR_ END	End address of flexspi1
-	This field is reserved. Reserved

11.4.30 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	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_GPR29 field descriptions

Field	Description
31–12 FLEXSPI_ REMAP_ADDR_ OFFSET	Offset address of flexspi1.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

11.5 IOMUXC SNVS Memory Map/Register Definition

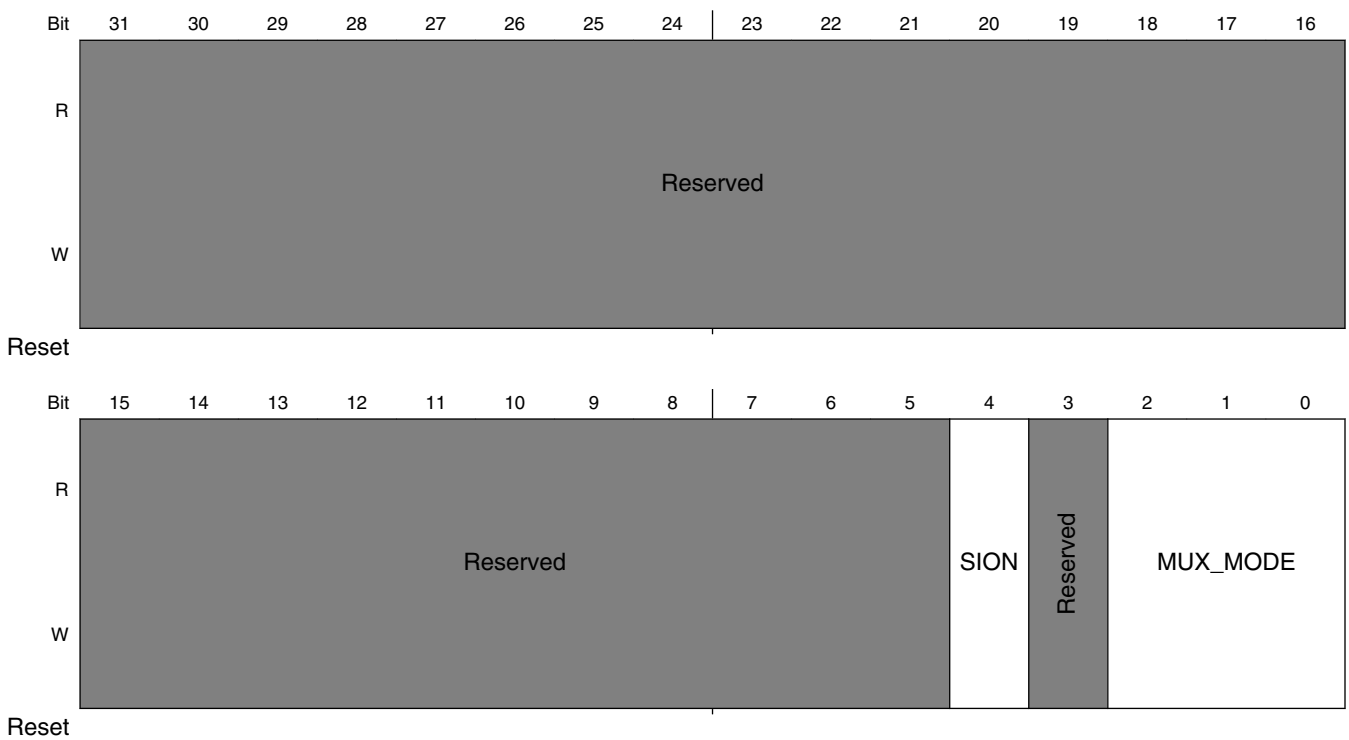
IOMUXC_SNVS memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400A_8000	SW_MUX_CTL_PAD_PMIC_ON_REQ SW MUX Control Register (IOMUXC_SNVS_SW_MUX_CTL_PAD_PMIC_ON_REQ)	32	R/W	0000_0000h	11.5.1/243
400A_8004	SW_PAD_CTL_PAD_TEST_MODE SW PAD Control Register (IOMUXC_SNVS_SW_PAD_CTL_PAD_TEST_MODE)	32	R/W	0000_30A0h	11.5.2/244
400A_8008	SW_PAD_CTL_PAD_POR_B SW PAD Control Register (IOMUXC_SNVS_SW_PAD_CTL_PAD_POR_B)	32	R/W	0001_B0A0h	11.5.3/246
400A_800C	SW_PAD_CTL_PAD_ONOFF SW PAD Control Register (IOMUXC_SNVS_SW_PAD_CTL_PAD_ONOFF)	32	R/W	0001_B0A0h	11.5.4/248
400A_8010	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	11.5.5/250

11.5.1 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 + 0h offset = 400A_8000h



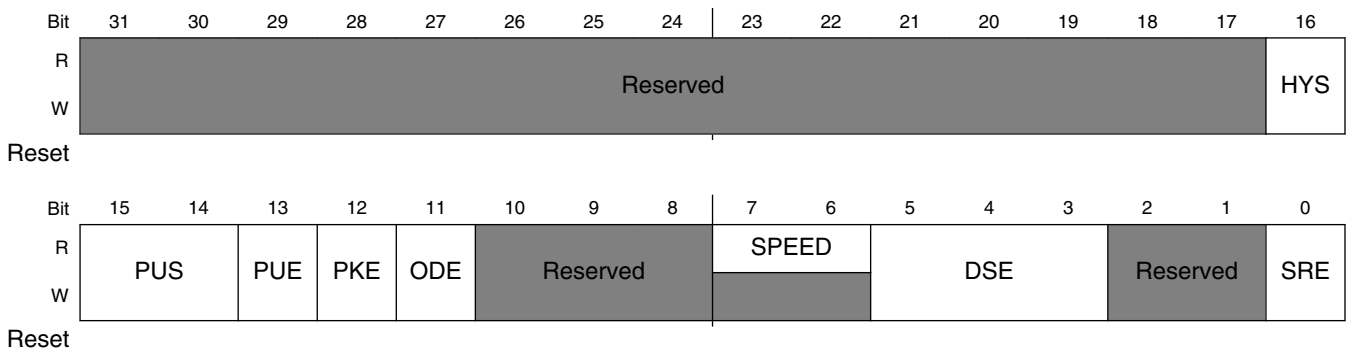
IOMUXC_SNVS_SW_MUX_CTL_PAD_PMIC_ON_REQ field descriptions

Field	Description
31–5 -	This field is reserved. Reserved
4 SION	Software Input On Field. Force the selected mux mode Input path no matter of MUX_MODE functionality. 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. 000 **ALT0** — Select mux mode: ALT0 mux port: SNVS_LP_PMIC_ON_REQ of instance: snvs_lp 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO5_IO00 of instance: gpio5

11.5.2 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 + 4h offset = 400A_8004h



IOMUXC_SNVS_SW_PAD_CTL_PAD_TEST_MODE field descriptions

Field	Description
31–17 -	This field is reserved. Reserved
16 HYS	Hyst. Enable Field Select one out of next values for pad: TEST_MODE

Table continues on the next page...

IOMUXC_SNVS_SW_PAD_CTL_PAD_TEST_MODE field descriptions (continued)

Field	Description
	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: 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 Select one out of next values for pad: TEST_MODE 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: TEST_MODE 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 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 Read Only Field 10 **SPEED** — medium(100MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: TEST_MODE 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

Table continues on the next page...

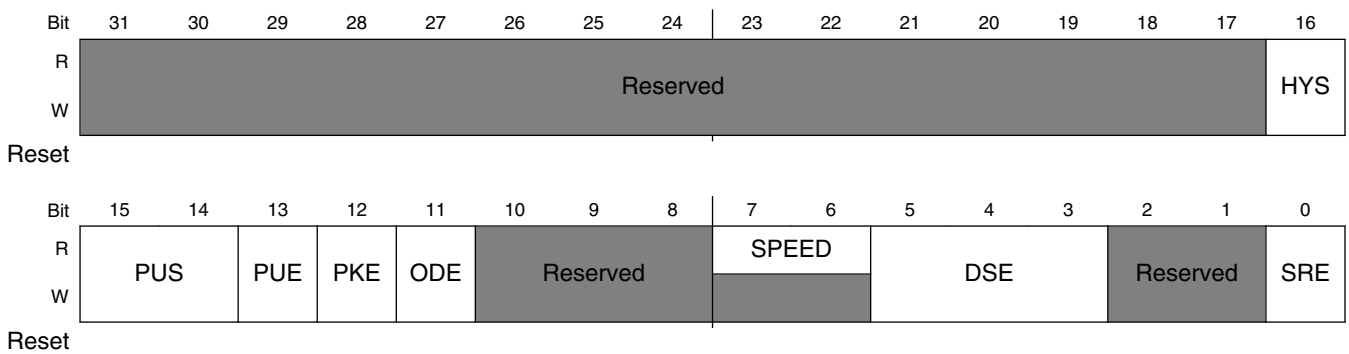
IOMUXC_SNVS_SW_PAD_CTL_PAD_TEST_MODE field descriptions (continued)

Field	Description
0 SRE	Slew Rate Field Select one out of next values for pad: TEST_MODE 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.5.3 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 + 8h offset = 400A_8008h



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 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: 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 Select one out of next values for pad: POR_B

Table continues on the next page...

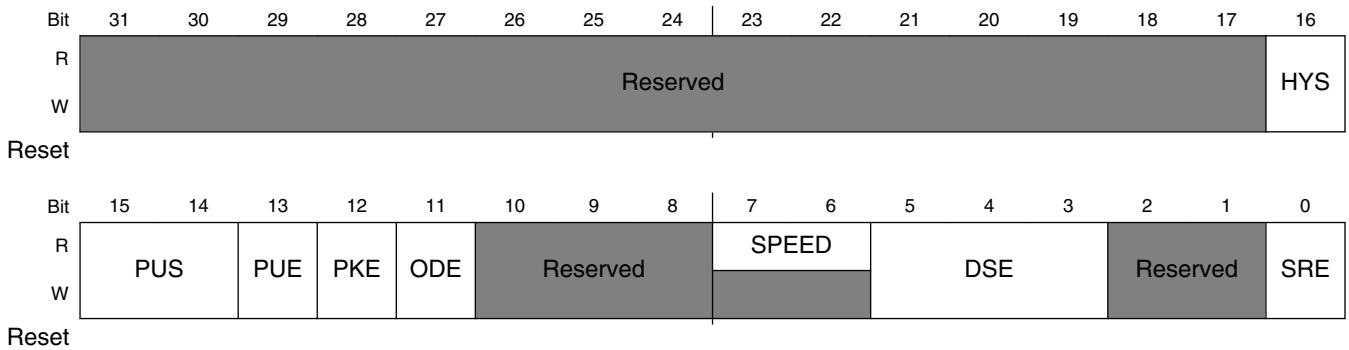
IOMUXC_SNVS_SW_PAD_CTL_PAD_POR_B field descriptions (continued)

Field	Description
	0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: POR_B 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 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 Read Only Field 10 **SPEED** — medium(100MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: POR_B 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: POR_B 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.5.4 SW_PAD_CTL_PAD_ONOFF SW PAD Control Register (IOMUXC_SNVS_SW_PAD_CTL_PAD_ONOFF)

SW_PAD_CTL Register

Address: 400A_8000h base + Ch offset = 400A_800Ch



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 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: ONOFF 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: ONOFF 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: ONOFF 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled

Table continues on the next page...

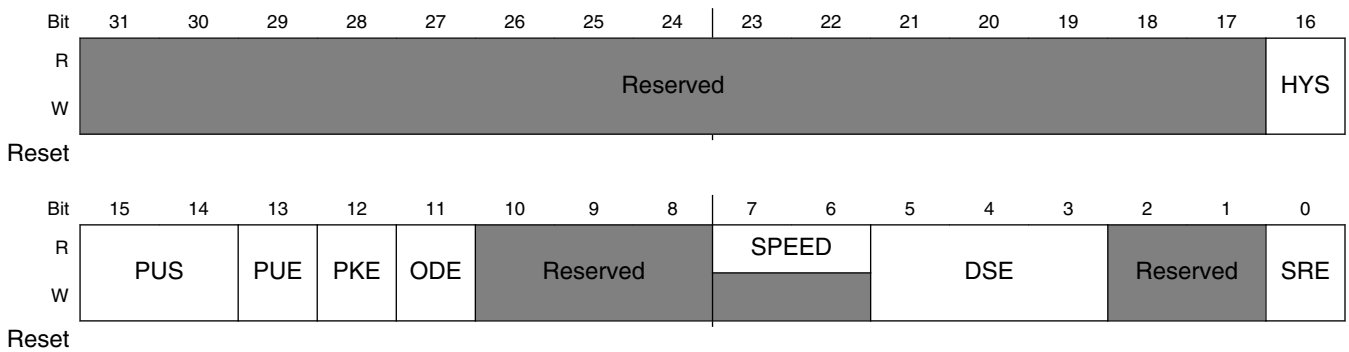
IOMUXC_SNVS_SW_PAD_CTL_PAD_ONOFF field descriptions (continued)

Field	Description
11 ODE	Open Drain Enable Field Select one out of next values for pad: ONOFF 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 Read Only Field 10 **SPEED** — medium(100MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: ONOFF 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: ONOFF 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.5.5 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 + 10h offset = 400A_8010h



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 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: 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 Select one out of next values for pad: PMIC_ON_REQ 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field Select one out of next values for pad: PMIC_ON_REQ 0 **PKE_0_Pull_Keeper_Disabled** — Pull/Keeper Disabled 1 **PKE_1_Pull_Keeper_Enabled** — Pull/Keeper Enabled

Table continues on the next page...

IOMUXC_SNVS_SW_PAD_CTL_PAD_PMIC_ON_REQ field descriptions (continued)

Field	Description
11 ODE	Open Drain Enable Field Select one out of next values for pad: PMIC_ON_REQ 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 Read Only Field 10 **SPEED** — medium(100MHz)
5–3 DSE	Drive Strength Field Select one out of next values for pad: PMIC_ON_REQ 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: PMIC_ON_REQ 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.6 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	11.6.1/252
400A_4004	GPR1 General Purpose Register (IOMUXC_SNVS_GPR_GPR1)	32	R/W	0000_0000h	11.6.2/252

Table continues on the next page...

IOMUXC_SNVS_GPR memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400A_4008	GPR2 General Purpose Register (IOMUXC_SNVS_GPR_GPR2)	32	R/W	0000_0000h	11.6.3/253
400A_400C	GPR3 General Purpose Register (IOMUXC_SNVS_GPR_GPR3)	32	R/W	0000_0000h	11.6.4/254

11.6.1 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

11.6.2 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

11.6.3 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

11.6.4 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
18 DCDC_OVER_VOL	DCDC output over voltage alert
17 DCDC_OVER_CUR	DCDC output over current alert
16 DCDC_IN_LOW_VOL	DCDC_IN low voltage detect.
15–4 -	This field is reserved. Reserved
3–2 POR_PULL_TYPE	POR_B pad control
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.
0 LPSR_MODE_ENABLE	Set to enable LPSR mode.

11.7 IOMUXC Memory Map/Register Definition

IOMUXC memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
401F_8010	SW_MUX_CTL_PAD_GPIO_AD_14 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_14)	32	R/W	0000_0005h	11.7.1/262
401F_8014	SW_MUX_CTL_PAD_GPIO_AD_13 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_13)	32	R/W	0000_0007h	11.7.2/263
401F_8018	SW_MUX_CTL_PAD_GPIO_AD_12 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_12)	32	R/W	0000_0007h	11.7.3/264
401F_801C	SW_MUX_CTL_PAD_GPIO_AD_11 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_11)	32	R/W	0000_0007h	11.7.4/266
401F_8020	SW_MUX_CTL_PAD_GPIO_AD_10 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_10)	32	R/W	0000_0007h	11.7.5/267

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
401F_8024	SW_MUX_CTL_PAD_GPIO_AD_09 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_09)	32	R/W	0000_0007h	11.7.6/268
401F_8028	SW_MUX_CTL_PAD_GPIO_AD_08 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_08)	32	R/W	0000_0007h	11.7.7/270
401F_802C	SW_MUX_CTL_PAD_GPIO_AD_07 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_07)	32	R/W	0000_0005h	11.7.8/271
401F_8030	SW_MUX_CTL_PAD_GPIO_AD_06 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_06)	32	R/W	0000_0005h	11.7.9/272
401F_8034	SW_MUX_CTL_PAD_GPIO_AD_05 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_05)	32	R/W	0000_0005h	11.7.10/274
401F_8038	SW_MUX_CTL_PAD_GPIO_AD_04 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_04)	32	R/W	0000_0005h	11.7.11/275
401F_803C	SW_MUX_CTL_PAD_GPIO_AD_03 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_03)	32	R/W	0000_0005h	11.7.12/276
401F_8040	SW_MUX_CTL_PAD_GPIO_AD_02 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_02)	32	R/W	0000_0005h	11.7.13/278
401F_8044	SW_MUX_CTL_PAD_GPIO_AD_01 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_01)	32	R/W	0000_0005h	11.7.14/279
401F_8048	SW_MUX_CTL_PAD_GPIO_AD_00 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_00)	32	R/W	0000_0005h	11.7.15/280
401F_804C	SW_MUX_CTL_PAD_GPIO_SD_14 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_14)	32	R/W	0000_0000h	11.7.16/281
401F_8050	SW_MUX_CTL_PAD_GPIO_SD_13 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_13)	32	R/W	0000_0005h	11.7.17/282
401F_8054	SW_MUX_CTL_PAD_GPIO_SD_12 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_12)	32	R/W	0000_0005h	11.7.18/284
401F_8058	SW_MUX_CTL_PAD_GPIO_SD_11 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_11)	32	R/W	0000_0005h	11.7.19/285
401F_805C	SW_MUX_CTL_PAD_GPIO_SD_10 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_10)	32	R/W	0000_0005h	11.7.20/286
401F_8060	SW_MUX_CTL_PAD_GPIO_SD_09 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_09)	32	R/W	0000_0005h	11.7.21/288
401F_8064	SW_MUX_CTL_PAD_GPIO_SD_08 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_08)	32	R/W	0000_0005h	11.7.22/289
401F_8068	SW_MUX_CTL_PAD_GPIO_SD_07 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_07)	32	R/W	0000_0005h	11.7.23/290
401F_806C	SW_MUX_CTL_PAD_GPIO_SD_06 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_06)	32	R/W	0000_0005h	11.7.24/291
401F_8070	SW_MUX_CTL_PAD_GPIO_SD_05 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_05)	32	R/W	0000_0005h	11.7.25/293
401F_8074	SW_MUX_CTL_PAD_GPIO_SD_04 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_04)	32	R/W	0000_0006h	11.7.26/294
401F_8078	SW_MUX_CTL_PAD_GPIO_SD_03 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_03)	32	R/W	0000_0006h	11.7.27/295

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
401F_807C	SW_MUX_CTL_PAD_GPIO_SD_02 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_02)	32	R/W	0000_0005h	11.7.28/297
401F_8080	SW_MUX_CTL_PAD_GPIO_SD_01 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_01)	32	R/W	0000_0005h	11.7.29/298
401F_8084	SW_MUX_CTL_PAD_GPIO_SD_00 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_00)	32	R/W	0000_0005h	11.7.30/299
401F_8088	SW_MUX_CTL_PAD_GPIO_13 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_13)	32	R/W	0000_0005h	11.7.31/301
401F_808C	SW_MUX_CTL_PAD_GPIO_12 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_12)	32	R/W	0000_0005h	11.7.32/302
401F_8090	SW_MUX_CTL_PAD_GPIO_11 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_11)	32	R/W	0000_0005h	11.7.33/303
401F_8094	SW_MUX_CTL_PAD_GPIO_10 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_10)	32	R/W	0000_0005h	11.7.34/305
401F_8098	SW_MUX_CTL_PAD_GPIO_09 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_09)	32	R/W	0000_0005h	11.7.35/306
401F_809C	SW_MUX_CTL_PAD_GPIO_08 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_08)	32	R/W	0000_0005h	11.7.36/307
401F_80A0	SW_MUX_CTL_PAD_GPIO_07 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_07)	32	R/W	0000_0005h	11.7.37/309
401F_80A4	SW_MUX_CTL_PAD_GPIO_06 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_06)	32	R/W	0000_0005h	11.7.38/310
401F_80A8	SW_MUX_CTL_PAD_GPIO_05 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_05)	32	R/W	0000_0005h	11.7.39/311
401F_80AC	SW_MUX_CTL_PAD_GPIO_04 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_04)	32	R/W	0000_0005h	11.7.40/313
401F_80B0	SW_MUX_CTL_PAD_GPIO_03 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_03)	32	R/W	0000_0005h	11.7.41/314
401F_80B4	SW_MUX_CTL_PAD_GPIO_02 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_02)	32	R/W	0000_0005h	11.7.42/315
401F_80B8	SW_MUX_CTL_PAD_GPIO_01 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_01)	32	R/W	0000_0005h	11.7.43/316
401F_80BC	SW_MUX_CTL_PAD_GPIO_00 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_00)	32	R/W	0000_0005h	11.7.44/318
401F_80C0	SW_PAD_CTL_PAD_GPIO_AD_14 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_14)	32	R/W	0000_10A0h	11.7.45/319
401F_80C4	SW_PAD_CTL_PAD_GPIO_AD_13 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_13)	32	R/W	0000_70A0h	11.7.46/321
401F_80C8	SW_PAD_CTL_PAD_GPIO_AD_12 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_12)	32	R/W	0000_30A0h	11.7.47/323
401F_80CC	SW_PAD_CTL_PAD_GPIO_AD_11 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_11)	32	R/W	0000_30A0h	11.7.48/325
401F_80D0	SW_PAD_CTL_PAD_GPIO_AD_10 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_10)	32	R/W	0000_70A0h	11.7.49/327

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
401F_80D4	SW_PAD_CTL_PAD_GPIO_AD_09 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_09)	32	R/W	0000_90B1h	11.7.50/329
401F_80D8	SW_PAD_CTL_PAD_GPIO_AD_08 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_08)	32	R/W	0000_70A0h	11.7.51/331
401F_80DC	SW_PAD_CTL_PAD_GPIO_AD_07 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_07)	32	R/W	0000_10A0h	11.7.52/333
401F_80E0	SW_PAD_CTL_PAD_GPIO_AD_06 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_06)	32	R/W	0000_10A0h	11.7.53/335
401F_80E4	SW_PAD_CTL_PAD_GPIO_AD_05 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_05)	32	R/W	0000_10A0h	11.7.54/337
401F_80E8	SW_PAD_CTL_PAD_GPIO_AD_04 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_04)	32	R/W	0000_10A0h	11.7.55/339
401F_80EC	SW_PAD_CTL_PAD_GPIO_AD_03 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_03)	32	R/W	0000_10A0h	11.7.56/341
401F_80F0	SW_PAD_CTL_PAD_GPIO_AD_02 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_02)	32	R/W	0000_10A0h	11.7.57/343
401F_80F4	SW_PAD_CTL_PAD_GPIO_AD_01 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_01)	32	R/W	0000_10A0h	11.7.58/345
401F_80F8	SW_PAD_CTL_PAD_GPIO_AD_00 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_00)	32	R/W	0000_10A0h	11.7.59/347
401F_80FC	SW_PAD_CTL_PAD_GPIO_SD_14 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_14)	32	R/W	0000_30A0h	11.7.60/349
401F_8100	SW_PAD_CTL_PAD_GPIO_SD_13 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_13)	32	R/W	0000_10A0h	11.7.61/351
401F_8104	SW_PAD_CTL_PAD_GPIO_SD_12 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_12)	32	R/W	0000_10A0h	11.7.62/353
401F_8108	SW_PAD_CTL_PAD_GPIO_SD_11 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_11)	32	R/W	0000_10A0h	11.7.63/355
401F_810C	SW_PAD_CTL_PAD_GPIO_SD_10 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_10)	32	R/W	0000_10A0h	11.7.64/357
401F_8110	SW_PAD_CTL_PAD_GPIO_SD_09 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_09)	32	R/W	0000_10A0h	11.7.65/359
401F_8114	SW_PAD_CTL_PAD_GPIO_SD_08 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_08)	32	R/W	0000_10A0h	11.7.66/361
401F_8118	SW_PAD_CTL_PAD_GPIO_SD_07 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_07)	32	R/W	0000_10A0h	11.7.67/363
401F_811C	SW_PAD_CTL_PAD_GPIO_SD_06 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_06)	32	R/W	0000_10A0h	11.7.68/365
401F_8120	SW_PAD_CTL_PAD_GPIO_SD_05 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_05)	32	R/W	0000_10A0h	11.7.69/367
401F_8124	SW_PAD_CTL_PAD_GPIO_SD_04 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_04)	32	R/W	0000_30A0h	11.7.70/369
401F_8128	SW_PAD_CTL_PAD_GPIO_SD_03 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_03)	32	R/W	0000_30A0h	11.7.71/371

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
401F_812C	SW_PAD_CTL_PAD_GPIO_SD_02 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_02)	32	R/W	0000_10A0h	11.7.72/373
401F_8130	SW_PAD_CTL_PAD_GPIO_SD_01 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_01)	32	R/W	0000_10A0h	11.7.73/375
401F_8134	SW_PAD_CTL_PAD_GPIO_SD_00 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_00)	32	R/W	0000_10A0h	11.7.74/377
401F_8138	SW_PAD_CTL_PAD_GPIO_13 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_13)	32	R/W	0000_10A0h	11.7.75/379
401F_813C	SW_PAD_CTL_PAD_GPIO_12 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_12)	32	R/W	0000_10A0h	11.7.76/381
401F_8140	SW_PAD_CTL_PAD_GPIO_11 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_11)	32	R/W	0000_10A0h	11.7.77/383
401F_8144	SW_PAD_CTL_PAD_GPIO_10 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_10)	32	R/W	0000_10A0h	11.7.78/385
401F_8148	SW_PAD_CTL_PAD_GPIO_09 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_09)	32	R/W	0000_10A0h	11.7.79/387
401F_814C	SW_PAD_CTL_PAD_GPIO_08 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_08)	32	R/W	0000_10A0h	11.7.80/389
401F_8150	SW_PAD_CTL_PAD_GPIO_07 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_07)	32	R/W	0000_10A0h	11.7.81/391
401F_8154	SW_PAD_CTL_PAD_GPIO_06 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_06)	32	R/W	0000_10A0h	11.7.82/393
401F_8158	SW_PAD_CTL_PAD_GPIO_05 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_05)	32	R/W	0000_10A0h	11.7.83/395
401F_815C	SW_PAD_CTL_PAD_GPIO_04 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_04)	32	R/W	0000_10A0h	11.7.84/397
401F_8160	SW_PAD_CTL_PAD_GPIO_03 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_03)	32	R/W	0000_10A0h	11.7.85/399
401F_8164	SW_PAD_CTL_PAD_GPIO_02 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_02)	32	R/W	0000_10A0h	11.7.86/401
401F_8168	SW_PAD_CTL_PAD_GPIO_01 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_01)	32	R/W	0000_10A0h	11.7.87/403
401F_816C	SW_PAD_CTL_PAD_GPIO_00 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_00)	32	R/W	0000_10A0h	11.7.88/405
401F_8170	USB_OTG_ID_SELECT_INPUT DAISY Register (IOMUXC_USB_OTG_ID_SELECT_INPUT)	32	R/W	0000_0000h	11.7.89/407
401F_8174	FLEXPWM1_PWMA_SELECT_INPUT_0 DAISY Register (IOMUXC_FLEXPWM1_PWMA_SELECT_INPUT_0)	32	R/W	0000_0000h	11.7.90/408
401F_8178	FLEXPWM1_PWMA_SELECT_INPUT_1 DAISY Register (IOMUXC_FLEXPWM1_PWMA_SELECT_INPUT_1)	32	R/W	0000_0000h	11.7.91/409
401F_817C	FLEXPWM1_PWMA_SELECT_INPUT_2 DAISY Register (IOMUXC_FLEXPWM1_PWMA_SELECT_INPUT_2)	32	R/W	0000_0000h	11.7.92/410
401F_8180	FLEXPWM1_PWMA_SELECT_INPUT_3 DAISY Register (IOMUXC_FLEXPWM1_PWMA_SELECT_INPUT_3)	32	R/W	0000_0000h	11.7.93/411

Table continues on the next page...

IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
401F_8184	FLEXPWM1_PWMB_SELECT_INPUT_0 DAISY Register (IOMUXC_FLEXPWM1_PWMB_SELECT_INPUT_0)	32	R/W	0000_0000h	11.7.94/412
401F_8188	FLEXPWM1_PWMB_SELECT_INPUT_1 DAISY Register (IOMUXC_FLEXPWM1_PWMB_SELECT_INPUT_1)	32	R/W	0000_0000h	11.7.95/413
401F_818C	FLEXPWM1_PWMB_SELECT_INPUT_2 DAISY Register (IOMUXC_FLEXPWM1_PWMB_SELECT_INPUT_2)	32	R/W	0000_0000h	11.7.96/414
401F_8190	FLEXPWM1_PWMB_SELECT_INPUT_3 DAISY Register (IOMUXC_FLEXPWM1_PWMB_SELECT_INPUT_3)	32	R/W	0000_0000h	11.7.97/415
401F_8194	FLEXSPI_DQS_FA_SELECT_INPUT DAISY Register (IOMUXC_FLEXSPI_DQS_FA_SELECT_INPUT)	32	R/W	0000_0000h	11.7.98/416
401F_8198	FLEXSPI_DQS_FB_SELECT_INPUT DAISY Register (IOMUXC_FLEXSPI_DQS_FB_SELECT_INPUT)	32	R/W	0000_0000h	11.7.99/417
401F_819C	KPP_COL_SELECT_INPUT_0 DAISY Register (IOMUXC_KPP_COL_SELECT_INPUT_0)	32	R/W	0000_0000h	11.7.100/ 418
401F_81A0	KPP_COL_SELECT_INPUT_1 DAISY Register (IOMUXC_KPP_COL_SELECT_INPUT_1)	32	R/W	0000_0000h	11.7.101/ 419
401F_81A4	KPP_COL_SELECT_INPUT_2 DAISY Register (IOMUXC_KPP_COL_SELECT_INPUT_2)	32	R/W	0000_0000h	11.7.102/ 420
401F_81A8	KPP_COL_SELECT_INPUT_3 DAISY Register (IOMUXC_KPP_COL_SELECT_INPUT_3)	32	R/W	0000_0000h	11.7.103/ 421
401F_81AC	KPP_ROW_SELECT_INPUT_0 DAISY Register (IOMUXC_KPP_ROW_SELECT_INPUT_0)	32	R/W	0000_0000h	11.7.104/ 422
401F_81B0	KPP_ROW_SELECT_INPUT_1 DAISY Register (IOMUXC_KPP_ROW_SELECT_INPUT_1)	32	R/W	0000_0000h	11.7.105/ 423
401F_81B4	KPP_ROW_SELECT_INPUT_2 DAISY Register (IOMUXC_KPP_ROW_SELECT_INPUT_2)	32	R/W	0000_0000h	11.7.106/ 424
401F_81B8	KPP_ROW_SELECT_INPUT_3 DAISY Register (IOMUXC_KPP_ROW_SELECT_INPUT_3)	32	R/W	0000_0000h	11.7.107/ 425
401F_81BC	LPI2C1_HREQ_SELECT_INPUT DAISY Register (IOMUXC_LPI2C1_HREQ_SELECT_INPUT)	32	R/W	0000_0000h	11.7.108/ 426
401F_81C0	LPI2C1_SCL_SELECT_INPUT DAISY Register (IOMUXC_LPI2C1_SCL_SELECT_INPUT)	32	R/W	0000_0000h	11.7.109/ 426
401F_81C4	LPI2C1_SDA_SELECT_INPUT DAISY Register (IOMUXC_LPI2C1_SDA_SELECT_INPUT)	32	R/W	0000_0000h	11.7.110/ 427
401F_81C8	LPI2C2_SCL_SELECT_INPUT DAISY Register (IOMUXC_LPI2C2_SCL_SELECT_INPUT)	32	R/W	0000_0000h	11.7.111/ 428
401F_81CC	LPI2C2_SDA_SELECT_INPUT DAISY Register (IOMUXC_LPI2C2_SDA_SELECT_INPUT)	32	R/W	0000_0000h	11.7.112/ 428
401F_81D0	LPSPI1_PCS_SELECT_INPUT_0 DAISY Register (IOMUXC_LPSPI1_PCS_SELECT_INPUT_0)	32	R/W	0000_0000h	11.7.113/ 429
401F_81D4	LPSPI1_SCK_SELECT_INPUT DAISY Register (IOMUXC_LPSPI1_SCK_SELECT_INPUT)	32	R/W	0000_0000h	11.7.114/ 430
401F_81D8	LPSPI1_SDI_SELECT_INPUT DAISY Register (IOMUXC_LPSPI1_SDI_SELECT_INPUT)	32	R/W	0000_0000h	11.7.115/ 431

Table continues on the next page...

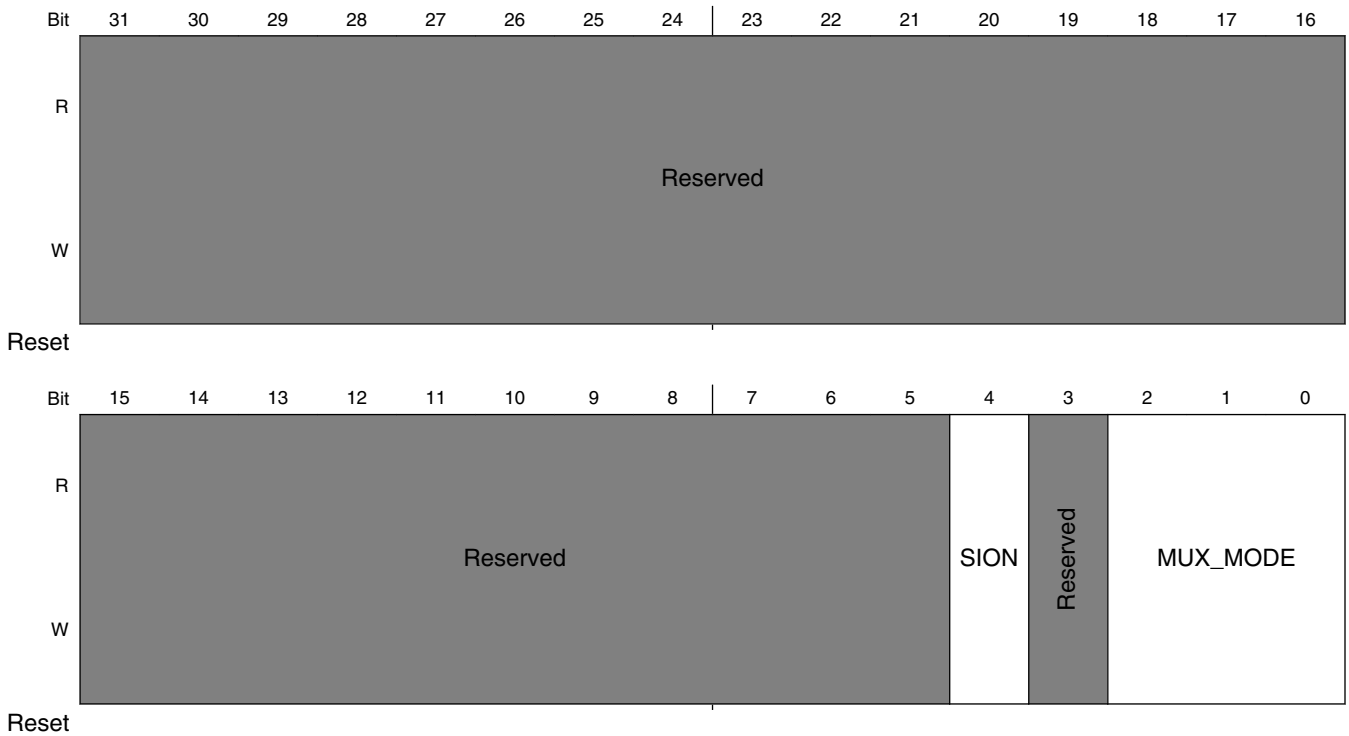
IOMUXC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
401F_81DC	LPSPI1_SDO_SELECT_INPUT DAISY Register (IOMUXC_LPSPI1_SDO_SELECT_INPUT)	32	R/W	0000_0000h	11.7.116/ 432
401F_81E0	LPSPI2_PCS_SELECT_INPUT_0 DAISY Register (IOMUXC_LPSPI2_PCS_SELECT_INPUT_0)	32	R/W	0000_0000h	11.7.117/ 433
401F_81E4	LPSPI2_SCK_SELECT_INPUT DAISY Register (IOMUXC_LPSPI2_SCK_SELECT_INPUT)	32	R/W	0000_0000h	11.7.118/ 434
401F_81E8	LPSPI2_SDI_SELECT_INPUT DAISY Register (IOMUXC_LPSPI2_SDI_SELECT_INPUT)	32	R/W	0000_0000h	11.7.119/ 435
401F_81EC	LPSPI2_SDO_SELECT_INPUT DAISY Register (IOMUXC_LPSPI2_SDO_SELECT_INPUT)	32	R/W	0000_0000h	11.7.120/ 436
401F_81F0	LPUART1_RXD_SELECT_INPUT DAISY Register (IOMUXC_LPUART1_RXD_SELECT_INPUT)	32	R/W	0000_0000h	11.7.121/ 437
401F_81F4	LPUART1_TXD_SELECT_INPUT DAISY Register (IOMUXC_LPUART1_TXD_SELECT_INPUT)	32	R/W	0000_0000h	11.7.122/ 438
401F_81F8	LPUART2_RXD_SELECT_INPUT DAISY Register (IOMUXC_LPUART2_RXD_SELECT_INPUT)	32	R/W	0000_0000h	11.7.123/ 439
401F_81FC	LPUART2_TXD_SELECT_INPUT DAISY Register (IOMUXC_LPUART2_TXD_SELECT_INPUT)	32	R/W	0000_0000h	11.7.124/ 440
401F_8200	LPUART3_RXD_SELECT_INPUT DAISY Register (IOMUXC_LPUART3_RXD_SELECT_INPUT)	32	R/W	0000_0000h	11.7.125/ 440
401F_8204	LPUART3_TXD_SELECT_INPUT DAISY Register (IOMUXC_LPUART3_TXD_SELECT_INPUT)	32	R/W	0000_0000h	11.7.126/ 441
401F_8208	LPUART4_RXD_SELECT_INPUT DAISY Register (IOMUXC_LPUART4_RXD_SELECT_INPUT)	32	R/W	0000_0000h	11.7.127/ 442
401F_820C	LPUART4_TXD_SELECT_INPUT DAISY Register (IOMUXC_LPUART4_TXD_SELECT_INPUT)	32	R/W	0000_0000h	11.7.128/ 443
401F_8210	NMI_GLUE_NMI_SELECT_INPUT DAISY Register (IOMUXC_NMI_GLUE_NMI_SELECT_INPUT)	32	R/W	0000_0000h	11.7.129/ 444
401F_8214	SPDIF_IN1_SELECT_INPUT DAISY Register (IOMUXC_SPDIF_IN1_SELECT_INPUT)	32	R/W	0000_0000h	11.7.130/ 445
401F_8218	SPDIF_TX_CLK2_SELECT_INPUT DAISY Register (IOMUXC_SPDIF_TX_CLK2_SELECT_INPUT)	32	R/W	0000_0000h	11.7.131/ 446
401F_821C	USB_OTG_OC_SELECT_INPUT DAISY Register (IOMUXC_USB_OTG_OC_SELECT_INPUT)	32	R/W	0000_0000h	11.7.132/ 447
401F_8220	XEV_GLUE_RXEV_SELECT_INPUT DAISY Register (IOMUXC_XEV_GLUE_RXEV_SELECT_INPUT)	32	R/W	0000_0000h	11.7.133/ 448

11.7.1 SW_MUX_CTL_PAD_GPIO_AD_14 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_14)

SW_MUX_CTL Register

Address: 401F_8000h base + 10h offset = 401F_8010h



IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_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_14 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO_AD_14. 000 **ALT0** — Select mux mode: ALT0 mux port: LPI2C1_SCL of instance: LPI2C1 001 **ALT1** — Select mux mode: ALT1 mux port: LPUART3_CTS_B of instance: LPUART3 010 **ALT2** — Select mux mode: ALT2 mux port: KPP_COL00 of instance: KPP

Table continues on the next page...

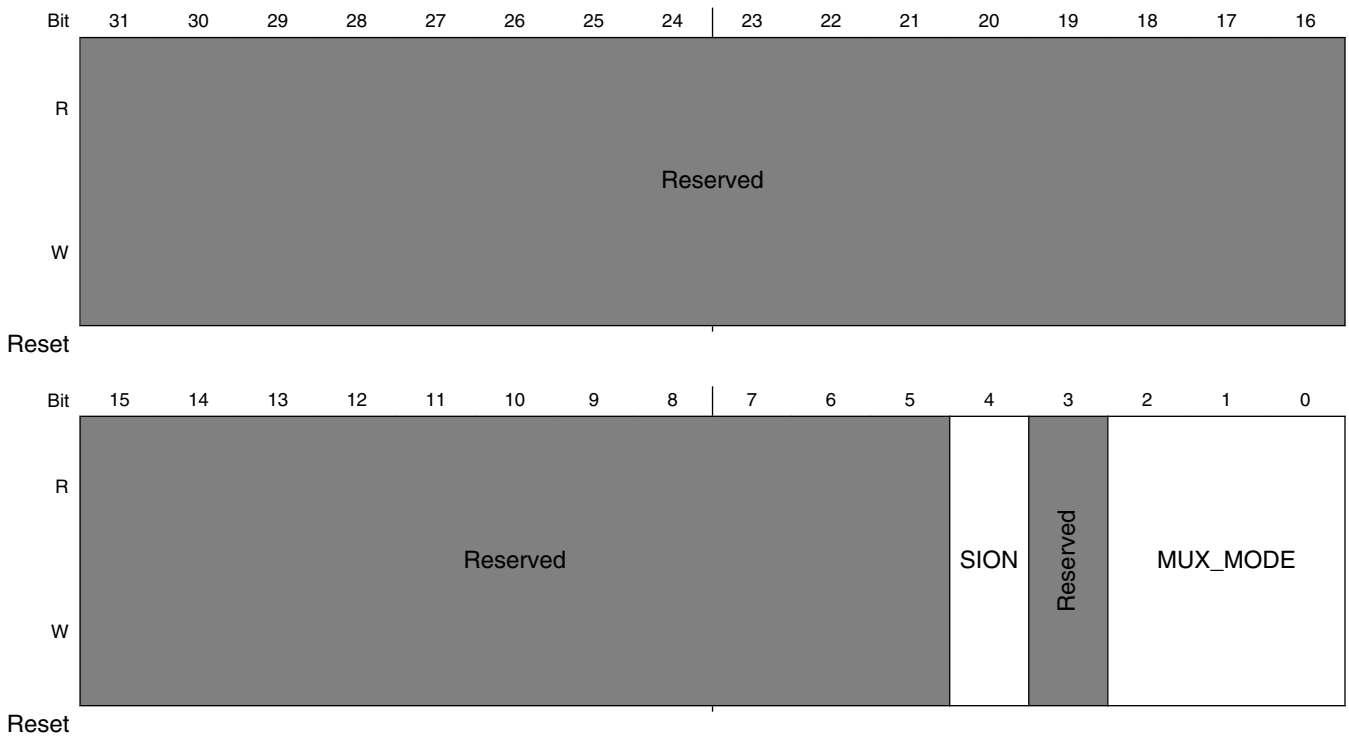
IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_14 field descriptions (continued)

Field	Description
	011 **ALT3** — Select mux mode: ALT3 mux port: LPUART4_CTS_B of instance: LPUART4 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO1_IO26 of instance: FLEXIO1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIOMUX_IO28 of instance: GPIOMUX 110 **ALT6** — Select mux mode: ALT6 mux port: REF_CLK_24M of instance: XTAL OSC 111 **ALT7** — Select mux mode: ALT7 mux port: XBAR1_INOUT02 of instance: XBAR1

11.7.2 SW_MUX_CTL_PAD_GPIO_AD_13 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_13)

SW_MUX_CTL Register

Address: 401F_8000h base + 14h offset = 401F_8014h



IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_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.

Table continues on the next page...

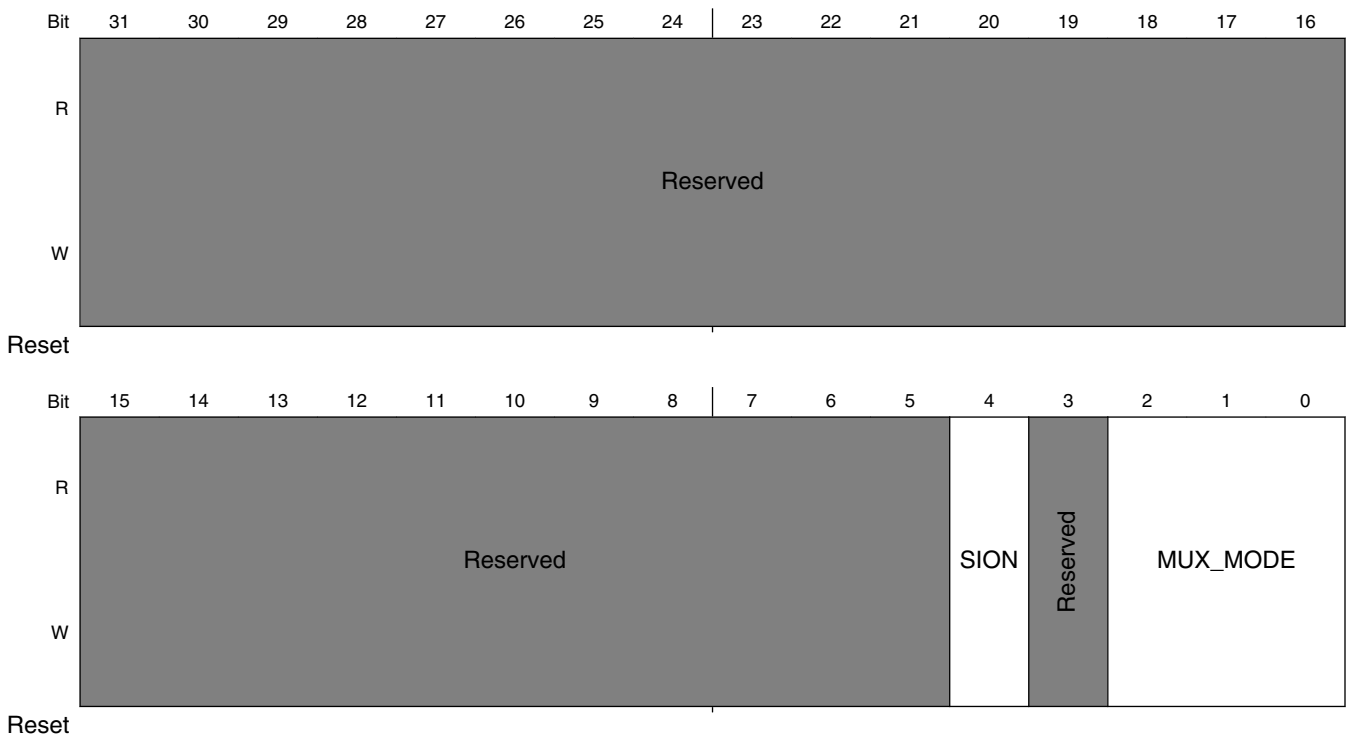
IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_13 field descriptions (continued)

Field	Description
	1 **ENABLED** — Force input path of pad GPIO_AD_13 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO_AD_13. 000 **ALT0** — Select mux mode: ALT0 mux port: LPI2C1_SDA of instance: LPI2C1 001 **ALT1** — Select mux mode: ALT1 mux port: LPUART3_RTS_B of instance: LPUART3 010 **ALT2** — Select mux mode: ALT2 mux port: KPP_ROW00 of instance: KPP 011 **ALT3** — Select mux mode: ALT3 mux port: LPUART4_RTS_B of instance: LPUART4 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO1_IO25 of instance: FLEXIO1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIOMUX_IO27 of instance: GPIOMUX 110 **ALT6** — Select mux mode: ALT6 mux port: NMI_GLUE_NMI of instance: NMI_GLUE 111 **ALT7** — Select mux mode: ALT7 mux port: JTAG_TMS of instance: JTAG

11.7.3 SW_MUX_CTL_PAD_GPIO_AD_12 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_12)

SW_MUX_CTL Register

Address: 401F_8000h base + 18h offset = 401F_8018h



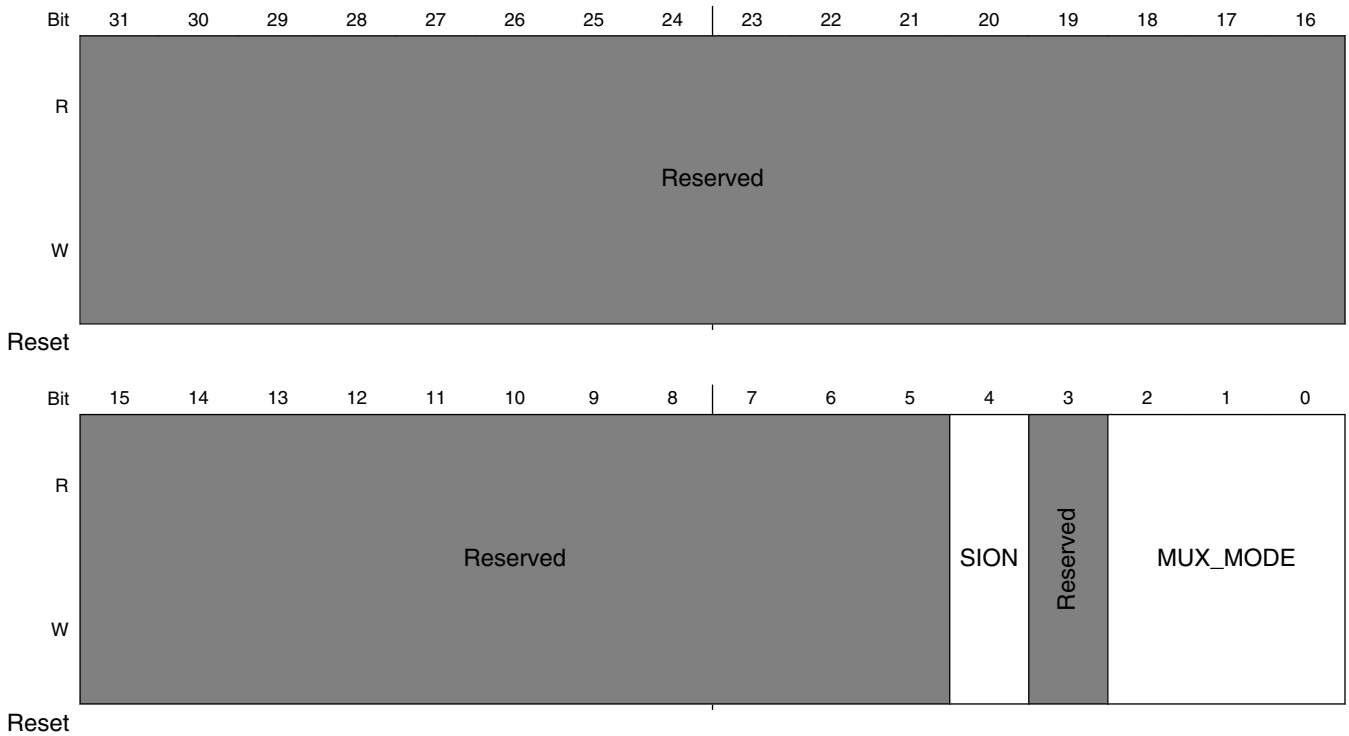
IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_12 field descriptions

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_12 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO_AD_12. 000 **ALT0** — Select mux mode: ALT0 mux port: LPSPI2_SCK of instance: LPSPI2 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXPWM1_PWM0_X of instance: FLEXPWM1 010 **ALT2** — Select mux mode: ALT2 mux port: KPP_COL01 of instance: KPP 011 **ALT3** — Select mux mode: ALT3 mux port: PIT_TRIGGER01 of instance: PIT 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO1_IO24 of instance: FLEXIO1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIOMUX_IO26 of instance: GPIOMUX 110 **ALT6** — Select mux mode: ALT6 mux port: USB_OTG1_PWR of instance: USB 111 **ALT7** — Select mux mode: ALT7 mux port: JTAG_TCK of instance: JTAG

11.7.4 SW_MUX_CTL_PAD_GPIO_AD_11 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_11)

SW_MUX_CTL Register

Address: 401F_8000h base + 1Ch offset = 401F_801Ch



IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_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_11 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO_AD_11. 000 **ALT0** — Select mux mode: ALT0 mux port: LPSPI2_PCS0 of instance: LPSPI2 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXPWM1_PWM1_X of instance: FLEXPWM1 010 **ALT2** — Select mux mode: ALT2 mux port: KPP_ROW01 of instance: KPP

Table continues on the next page...

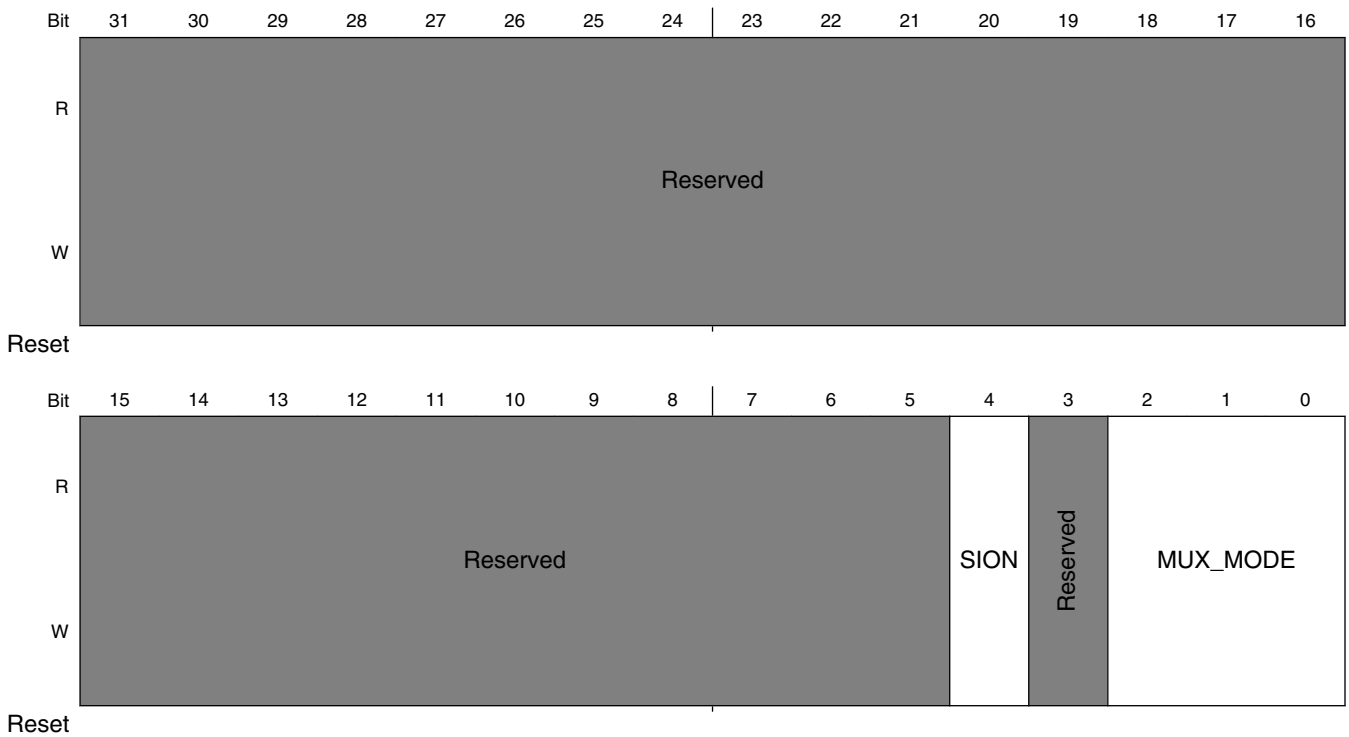
IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_11 field descriptions (continued)

Field	Description
	011 **ALT3** — Select mux mode: ALT3 mux port: PIT_TRIGGER02 of instance: PIT 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO1_IO23 of instance: FLEXIO1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIOMUX_IO25 of instance: GPIOMUX 110 **ALT6** — Select mux mode: ALT6 mux port: WDOG1_B of instance: WDOG1 111 **ALT7** — Select mux mode: ALT7 mux port: JTAG_MOD of instance: JTAG

11.7.5 SW_MUX_CTL_PAD_GPIO_AD_10 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_10)

SW_MUX_CTL Register

Address: 401F_8000h base + 20h offset = 401F_8020h



IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_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.

Table continues on the next page...

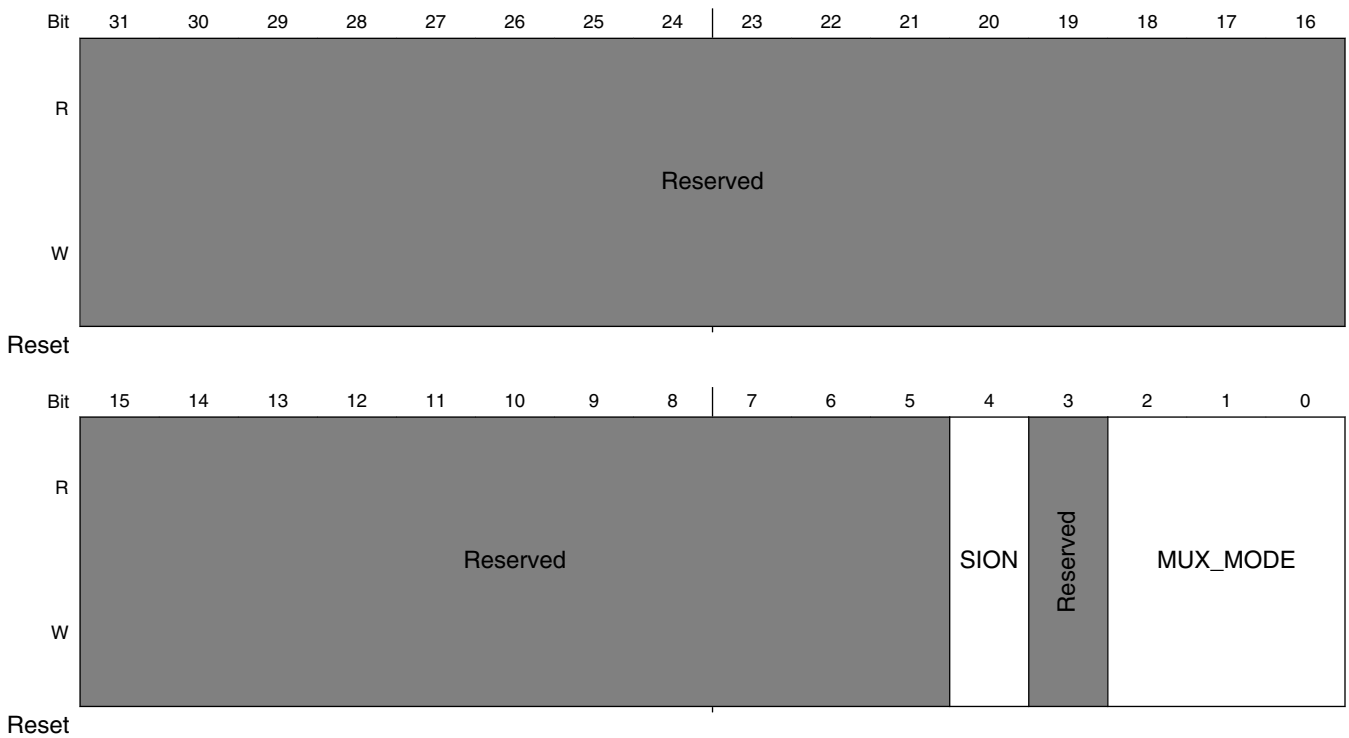
IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_10 field descriptions (continued)

Field	Description
	1 **ENABLED** — Force input path of pad GPIO_AD_10 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO_AD_10. 000 **ALT0** — Select mux mode: ALT0 mux port: LPSPI2_SDO of instance: LPSPI2 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXPWM1_PWM2_X of instance: FLEXPWM1 010 **ALT2** — Select mux mode: ALT2 mux port: KPP_COL02 of instance: KPP 011 **ALT3** — Select mux mode: ALT3 mux port: PIT_TRIGGER03 of instance: PIT 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO1_IO22 of instance: FLEXIO1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIOMUX_IO24 of instance: GPIOMUX 110 **ALT6** — Select mux mode: ALT6 mux port: OTG1_ID of instance: anatop 111 **ALT7** — Select mux mode: ALT7 mux port: JTAG_TDI of instance: JTAG

11.7.6 SW_MUX_CTL_PAD_GPIO_AD_09 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_09)

SW_MUX_CTL Register

Address: 401F_8000h base + 24h offset = 401F_8024h



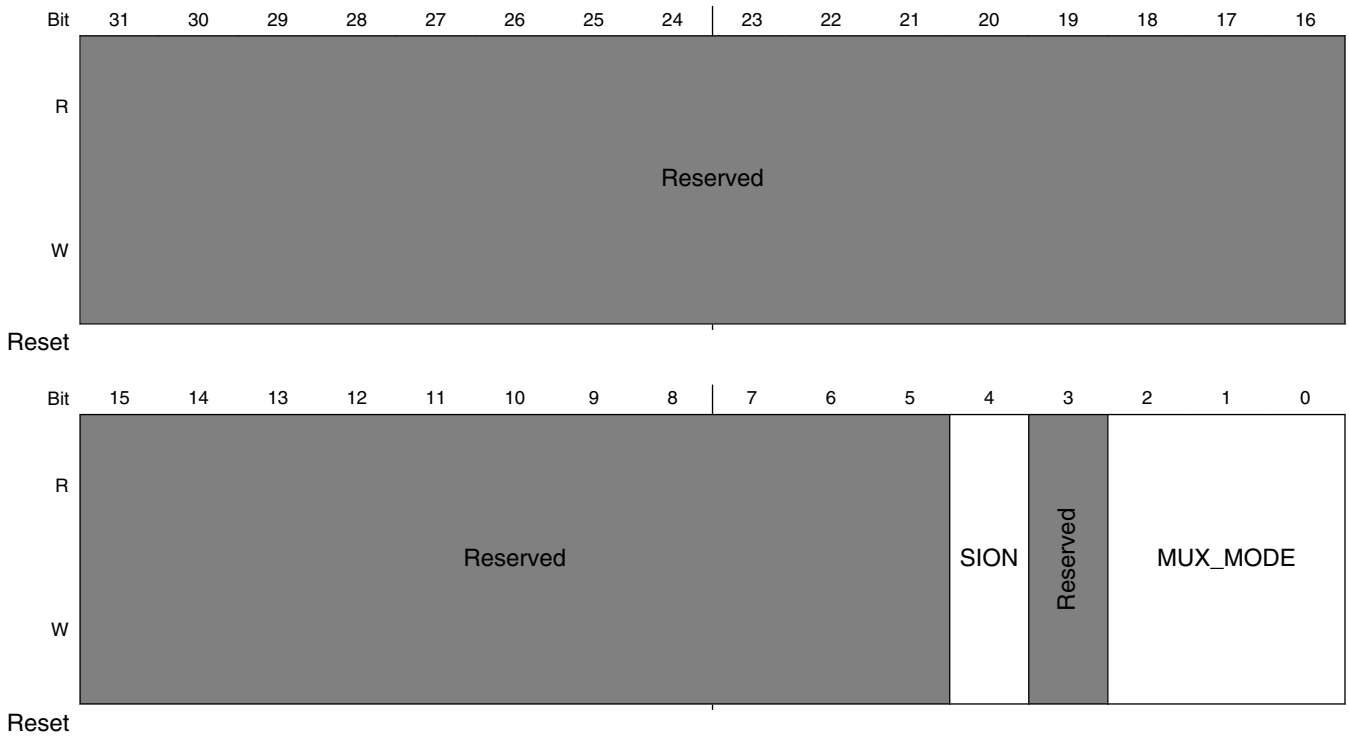
IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_09 field descriptions

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_09 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO_AD_09. 000 **ALT0** — Select mux mode: ALT0 mux port: LPSPI2_SDI of instance: LPSPI2 001 **ALT1** — Select mux mode: ALT1 mux port: FLEXPWM1_PWM3_X of instance: FLEXPWM1 010 **ALT2** — Select mux mode: ALT2 mux port: KPP_ROW02 of instance: KPP 011 **ALT3** — Select mux mode: ALT3 mux port: ARM_TRACE_SWO of instance: cm7_mxrt 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO1_IO21 of instance: FLEXIO1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIOMUX_IO23 of instance: GPIOMUX 110 **ALT6** — Select mux mode: ALT6 mux port: REF_32K_OUT of instance: anatop 111 **ALT7** — Select mux mode: ALT7 mux port: JTAG_TDO of instance: JTAG

11.7.7 SW_MUX_CTL_PAD_GPIO_AD_08 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_08)

SW_MUX_CTL Register

Address: 401F_8000h base + 28h offset = 401F_8028h



IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_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_08 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO_AD_08. 000 **ALT0** — Select mux mode: ALT0 mux port: LPI2C2_SCL of instance: LPI2C2 001 **ALT1** — Select mux mode: ALT1 mux port: LPUART3_TXD of instance: LPUART3 010 **ALT2** — Select mux mode: ALT2 mux port: ARM_CM7_TXEV of instance: cm7_mxrt

Table continues on the next page...

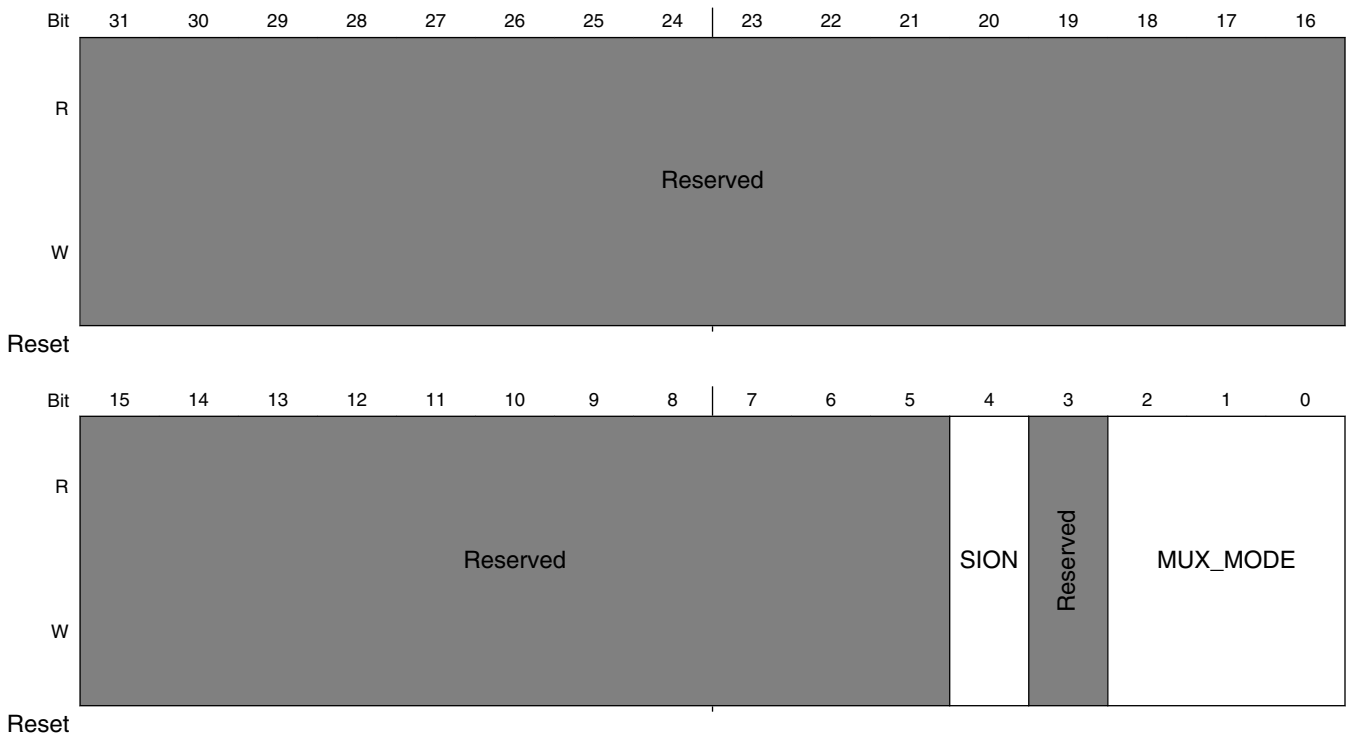
IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_08 field descriptions (continued)

Field	Description
	011 **ALT3** — Select mux mode: ALT3 mux port: LPUART2_CTS_B of instance: LPUART2 100 **ALT4** — Select mux mode: ALT4 mux port: GPT2_COMPARE3 of instance: GPT2 101 **ALT5** — Select mux mode: ALT5 mux port: GPIOMUX_IO22 of instance: GPIOMUX 110 **ALT6** — Select mux mode: ALT6 mux port:EWM_OUT_B of instance: EWM 111 **ALT7** — Select mux mode: ALT7 mux port: JTAG_TRSTB of instance: JTAG

11.7.8 SW_MUX_CTL_PAD_GPIO_AD_07 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_07)

SW_MUX_CTL Register

Address: 401F_8000h base + 2Ch offset = 401F_802Ch



IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_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.

Table continues on the next page...

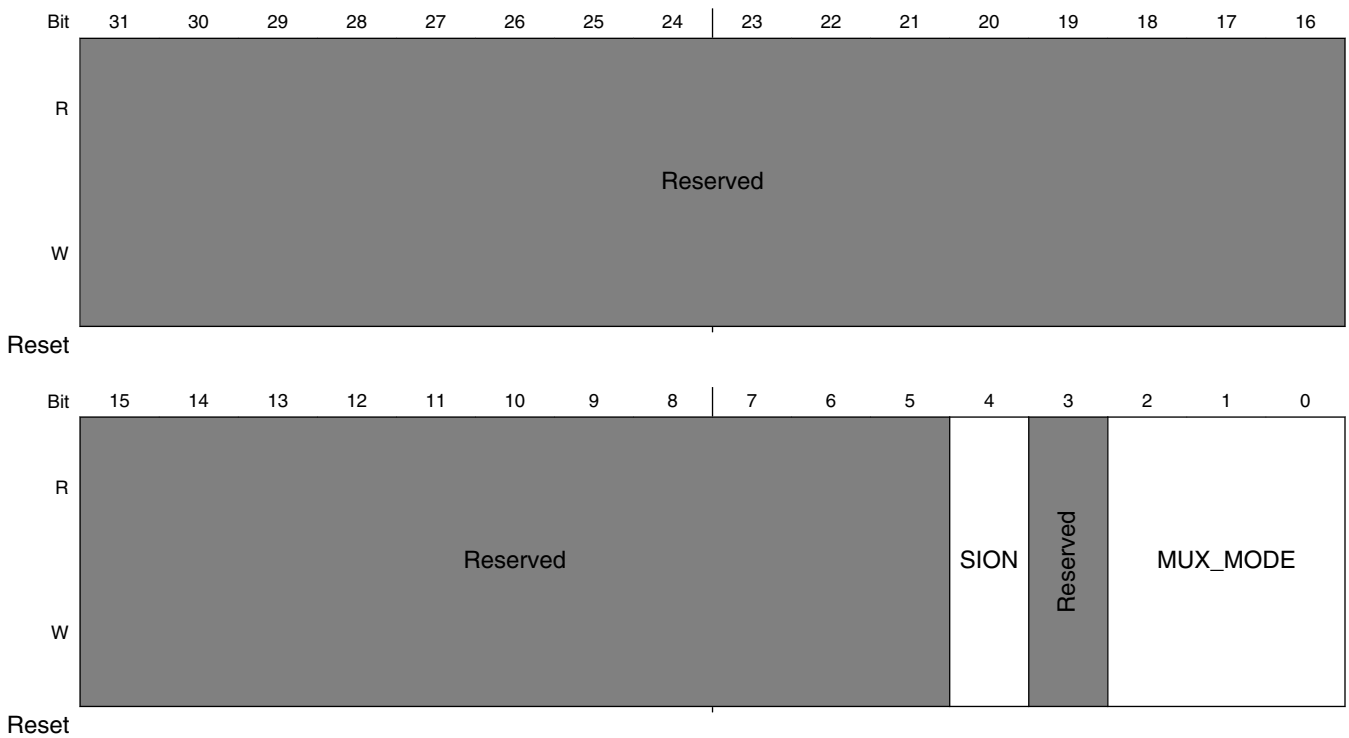
IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_07 field descriptions (continued)

Field	Description
	1 **ENABLED** — Force input path of pad GPIO_AD_07 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO_AD_07. 000 **ALT0** — Select mux mode: ALT0 mux port: LPI2C2_SDA of instance: LPI2C2 001 **ALT1** — Select mux mode: ALT1 mux port: LPUART3_RXD of instance: LPUART3 010 **ALT2** — Select mux mode: ALT2 mux port: ARM_CM7_RXEV of instance: cm7_mxrt 011 **ALT3** — Select mux mode: ALT3 mux port: LPUART2_RTS_B of instance: LPUART2 100 **ALT4** — Select mux mode: ALT4 mux port: GPT2_CAPTURE2 of instance: GPT2 101 **ALT5** — Select mux mode: ALT5 mux port: GPIOMUX_IO21 of instance: GPIOMUX 110 **ALT6** — Select mux mode: ALT6 mux port: OCOTP_FUSE_LATCHED of instance: OCOTP 111 **ALT7** — Select mux mode: ALT7 mux port: XBAR1_INOUT03 of instance: XBAR1

11.7.9 SW_MUX_CTL_PAD_GPIO_AD_06 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_06)

SW_MUX_CTL Register

Address: 401F_8000h base + 30h offset = 401F_8030h



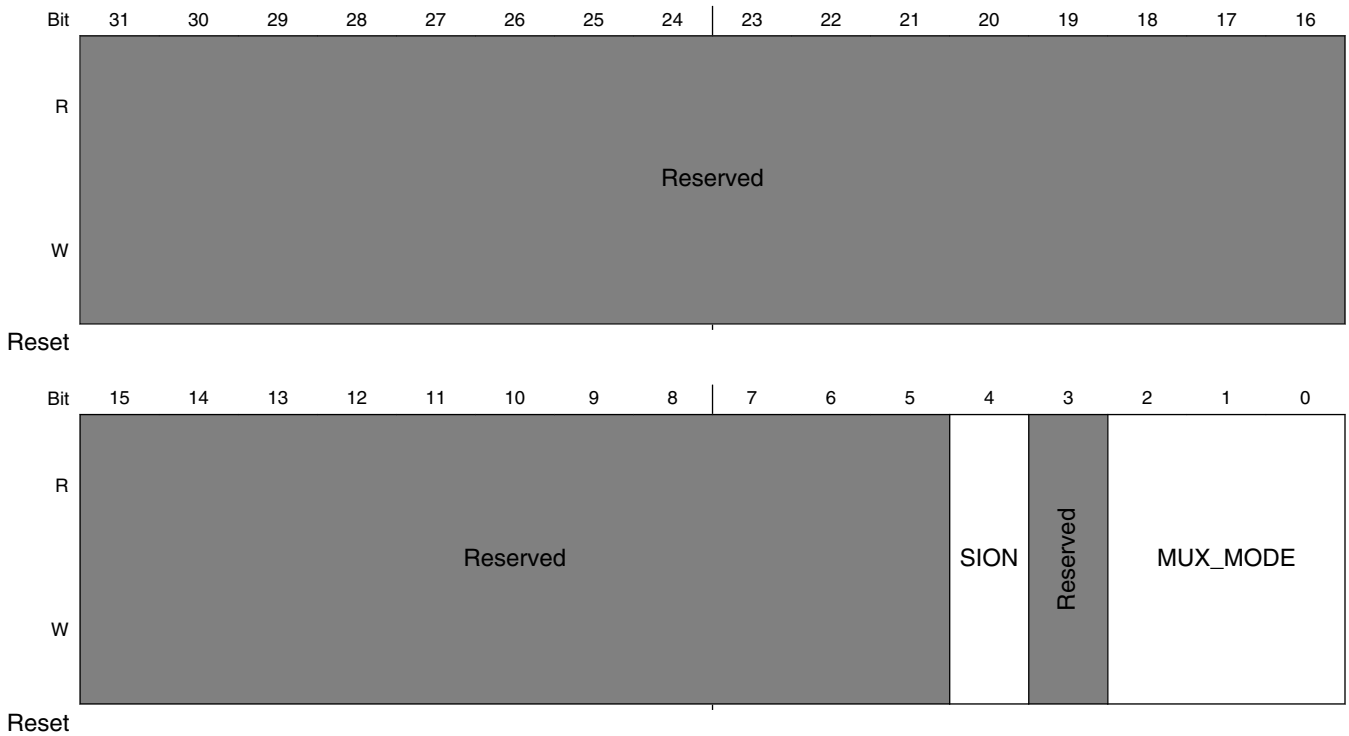
IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_06 field descriptions

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_06 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO_AD_06. 000 **ALT0** — Select mux mode: ALT0 mux port: LPSPI1_SCK of instance: LPSPI1 001 **ALT1** — Select mux mode: ALT1 mux port: PIT_TRIGGER00 of instance: PIT 010 **ALT2** — Select mux mode: ALT2 mux port: FLEXPWM1_PWM3_A of instance: FLEXPWM1 011 **ALT3** — Select mux mode: ALT3 mux port: KPP_COL01 of instance: KPP 100 **ALT4** — Select mux mode: ALT4 mux port: GPT2_COMPARE2 of instance: GPT2 101 **ALT5** — Select mux mode: ALT5 mux port: GPIOMUX_IO20 of instance: GPIOMUX 110 **ALT6** — Select mux mode: ALT6 mux port: LPI2C1_HREQ of instance: LPI2C1

11.7.10 SW_MUX_CTL_PAD_GPIO_AD_05 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_05)

SW_MUX_CTL Register

Address: 401F_8000h base + 34h offset = 401F_8034h



IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_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_05 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO_AD_05. 000 **ALT0** — Select mux mode: ALT0 mux port: LPSPI1_PCS0 of instance: LPSPI1 001 **ALT1** — Select mux mode: ALT1 mux port: PIT_TRIGGER01 of instance: PIT 010 **ALT2** — Select mux mode: ALT2 mux port: FLEXPWM1_PWM3_B of instance: FLEXPWM1

Table continues on the next page...

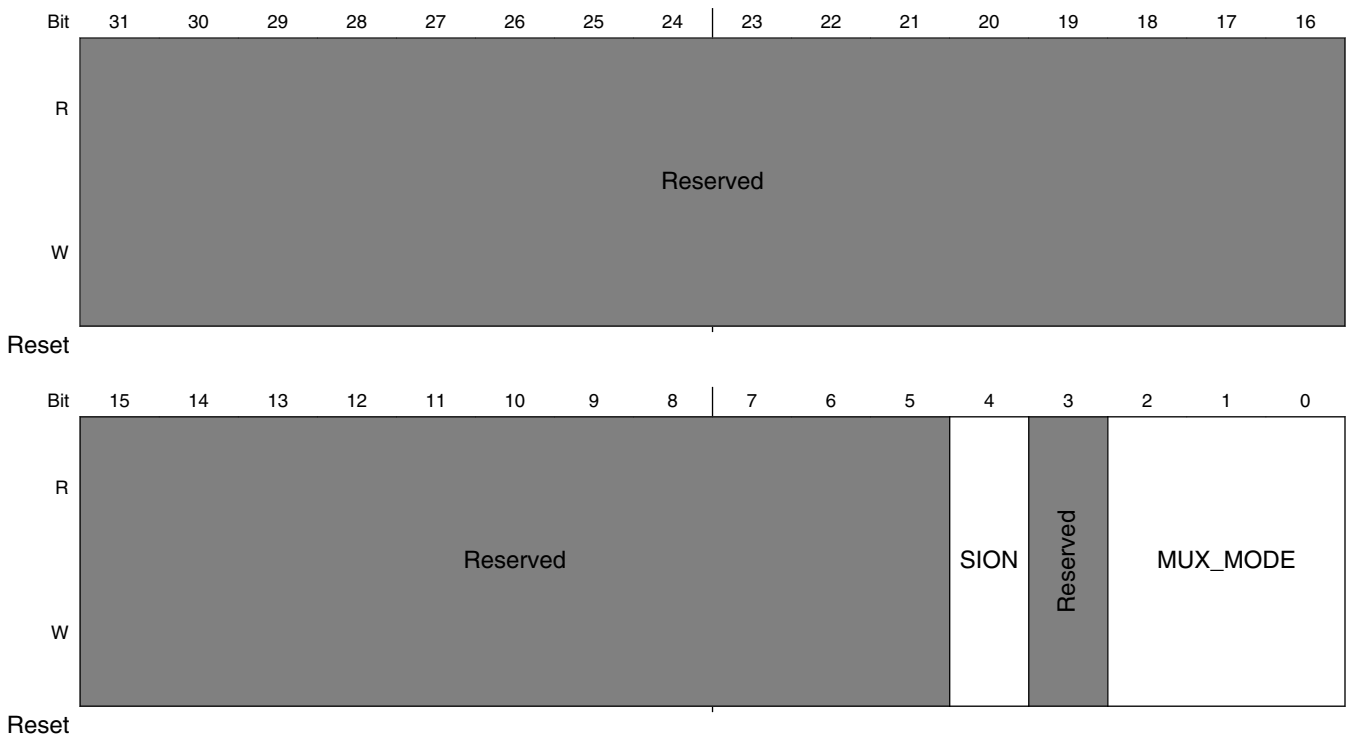
IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_05 field descriptions (continued)

Field	Description
	011 **ALT3** — Select mux mode: ALT3 mux port: KPP_ROW01 of instance: KPP 100 **ALT4** — Select mux mode: ALT4 mux port: GPT2_CAPTURE1 of instance: GPT2 101 **ALT5** — Select mux mode: ALT5 mux port: GPIOMUX_IO19 of instance: GPIOMUX

11.7.11 SW_MUX_CTL_PAD_GPIO_AD_04 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_04)

SW_MUX_CTL Register

Address: 401F_8000h base + 38h offset = 401F_8038h



IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_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_04 0 **DISABLED** — Input Path is determined by functionality

Table continues on the next page...

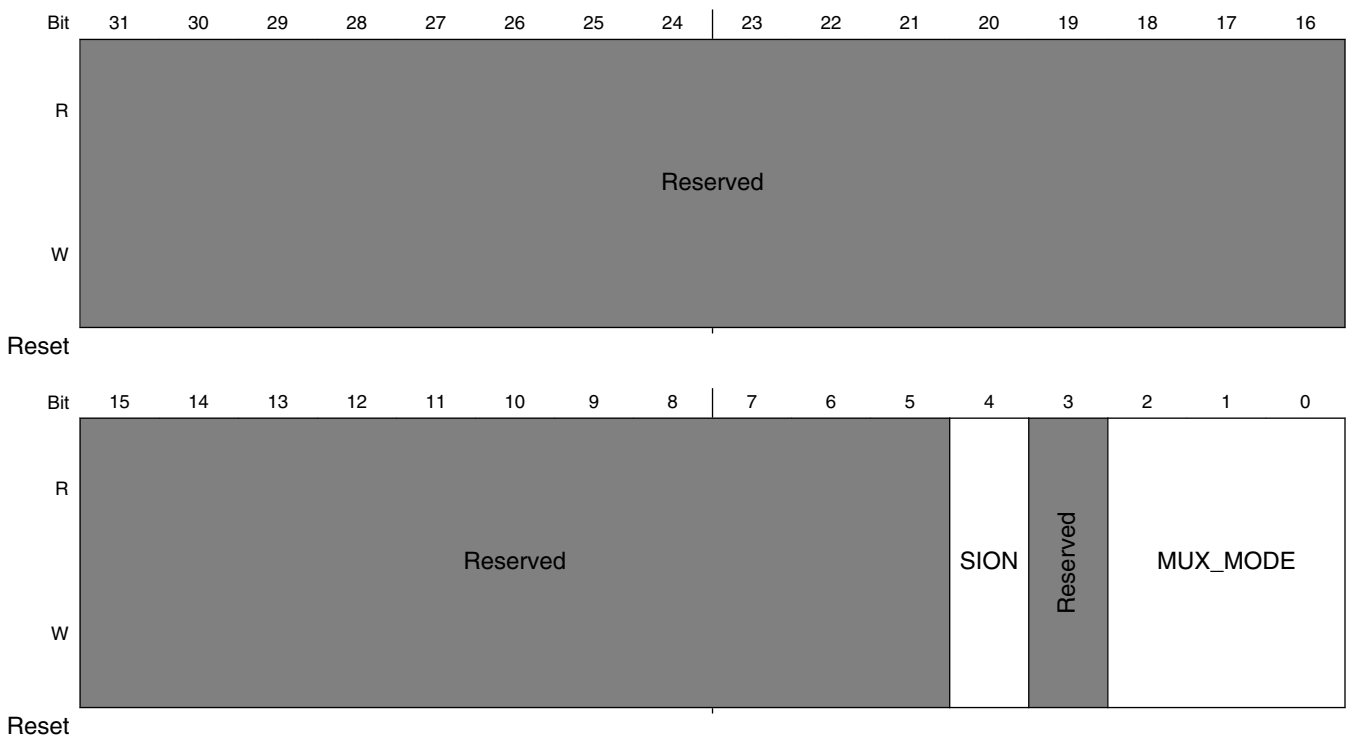
IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_04 field descriptions (continued)

Field	Description
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO_AD_04. 000 **ALT0** — Select mux mode: ALT0 mux port: LPSPI1_SDO of instance: LPSPI1 001 **ALT1** — Select mux mode: ALT1 mux port: PIT_TRIGGER02 of instance: PIT 010 **ALT2** — Select mux mode: ALT2 mux port: FLEXPWM1_PWM2_A of instance: FLEXPWM1 011 **ALT3** — Select mux mode: ALT3 mux port: KPP_COL02 of instance: KPP 100 **ALT4** — Select mux mode: ALT4 mux port: GPT2_COMPARE1 of instance: GPT2 101 **ALT5** — Select mux mode: ALT5 mux port: GPIOMUX_IO18 of instance: GPIOMUX 110 **ALT6** — Select mux mode: ALT6 mux port: SNVS_HP_VIO_5_CTL of instance: snvs_hp

11.7.12 SW_MUX_CTL_PAD_GPIO_AD_03 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_03)

SW_MUX_CTL Register

Address: 401F_8000h base + 3Ch offset = 401F_803Ch



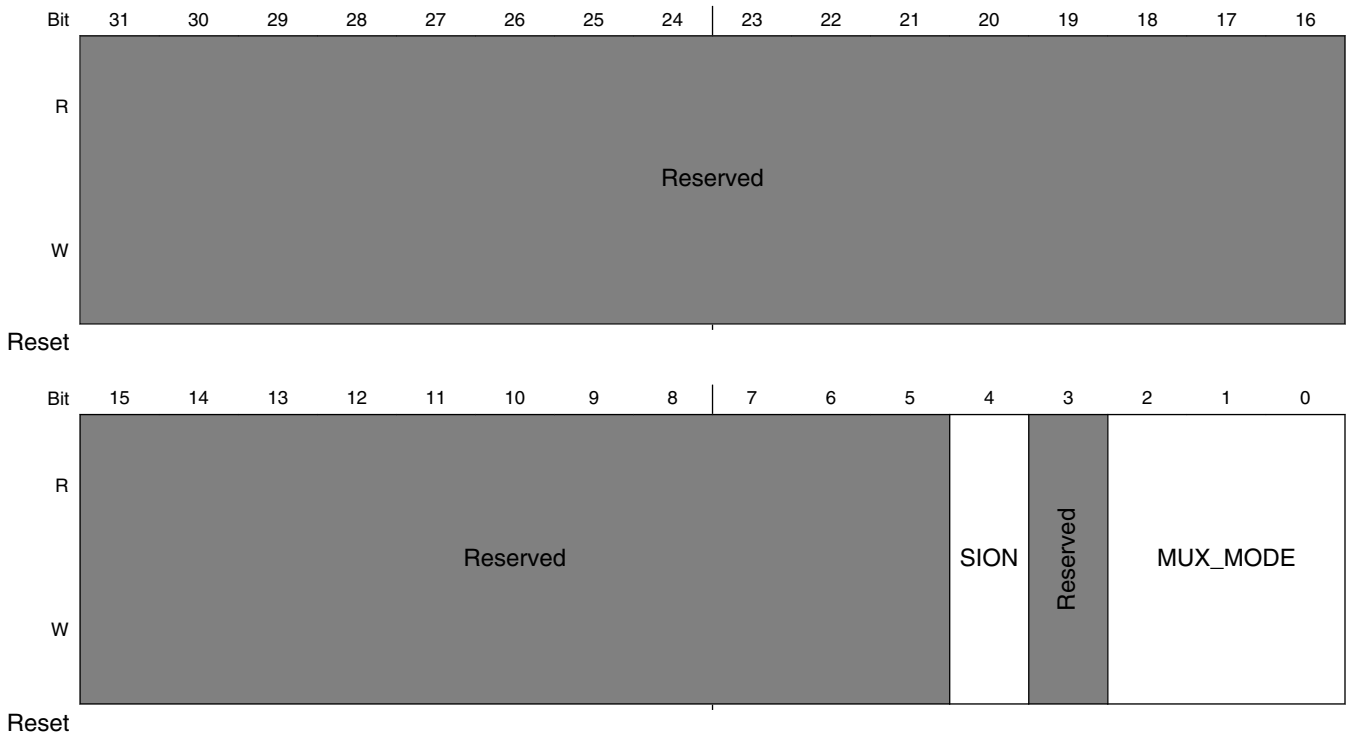
IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_03 field descriptions

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_03 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO_AD_03. 000 **ALT0** — Select mux mode: ALT0 mux port: LPSPI1_SDI of instance: LPSPI1 001 **ALT1** — Select mux mode: ALT1 mux port: PIT_TRIGGER03 of instance: PIT 010 **ALT2** — Select mux mode: ALT2 mux port: FLEXPWM1_PWM2_B of instance: FLEXPWM1 011 **ALT3** — Select mux mode: ALT3 mux port: KPP_ROW02 of instance: KPP 100 **ALT4** — Select mux mode: ALT4 mux port: GPT2_CLK of instance: GPT2 101 **ALT5** — Select mux mode: ALT5 mux port: GPIOMUX_IO17 of instance: GPIOMUX 110 **ALT6** — Select mux mode: ALT6 mux port: SNVS_HP_VIO_5_B of instance: snvs_hp 111 **ALT7** — Select mux mode: ALT7 mux port: JTAG_DE_B of instance: JTAG

11.7.13 SW_MUX_CTL_PAD_GPIO_AD_02 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_02)

SW_MUX_CTL Register

Address: 401F_8000h base + 40h offset = 401F_8040h



IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_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_02 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO_AD_02. 000 **ALT0** — Select mux mode: ALT0 mux port: LPUART4_TXD of instance: LPUART4 001 **ALT1** — Select mux mode: ALT1 mux port: LPSPI1_PCS1 of instance: LPSPI1 010 **ALT2** — Select mux mode: ALT2 mux port: WDOG2_B of instance: WDOG2

Table continues on the next page...

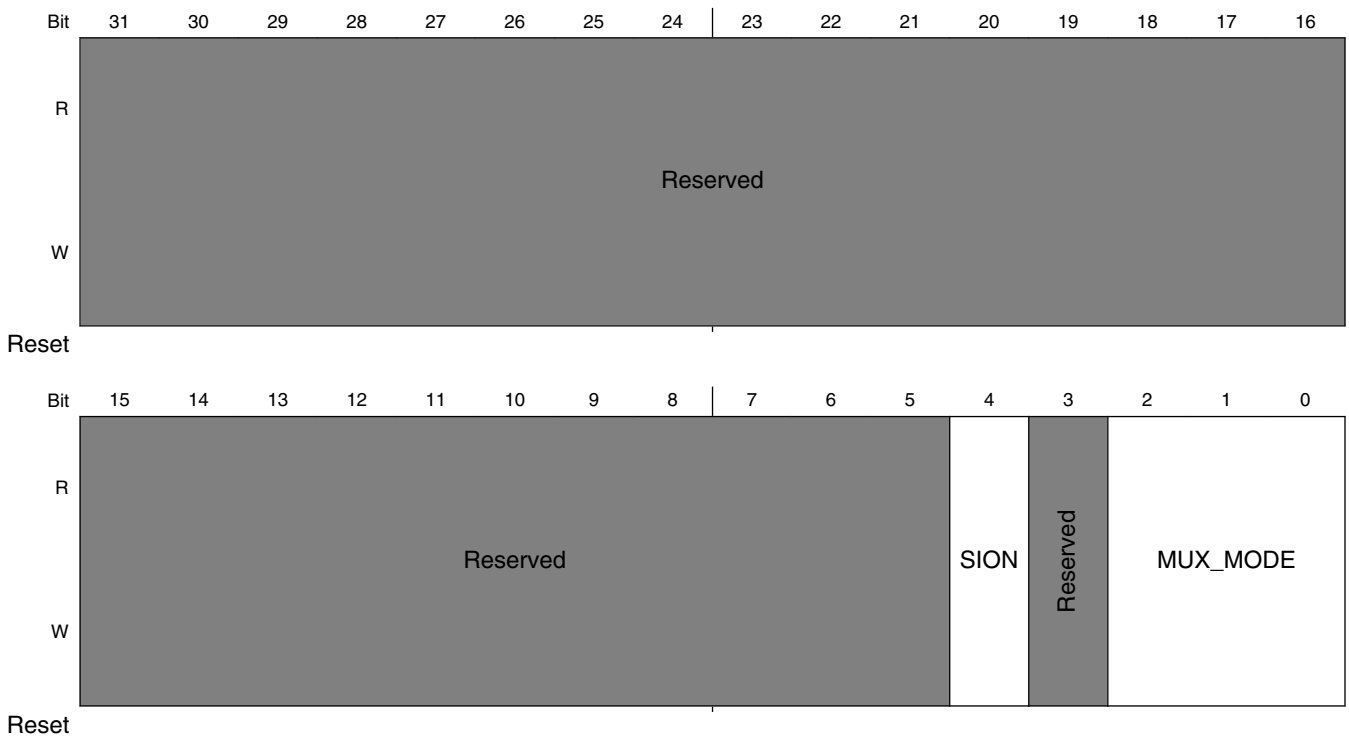
IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_02 field descriptions (continued)

Field	Description
	011 **ALT3** — Select mux mode: ALT3 mux port: LPI2C2_SCL of instance: LPI2C2 100 **ALT4** — Select mux mode: ALT4 mux port: MQS_RIGHT of instance: MQS 101 **ALT5** — Select mux mode: ALT5 mux port: GPIOMUX_IO16 of instance: GPIOMUX 111 **ALT7** — Select mux mode: ALT7 mux port: ARM_CM7_TRACE_CLK of instance: cm7_mxrt

11.7.14 SW_MUX_CTL_PAD_GPIO_AD_01 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_01)

SW_MUX_CTL Register

Address: 401F_8000h base + 44h offset = 401F_8044h



IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_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_01 0 **DISABLED** — Input Path is determined by functionality

Table continues on the next page...

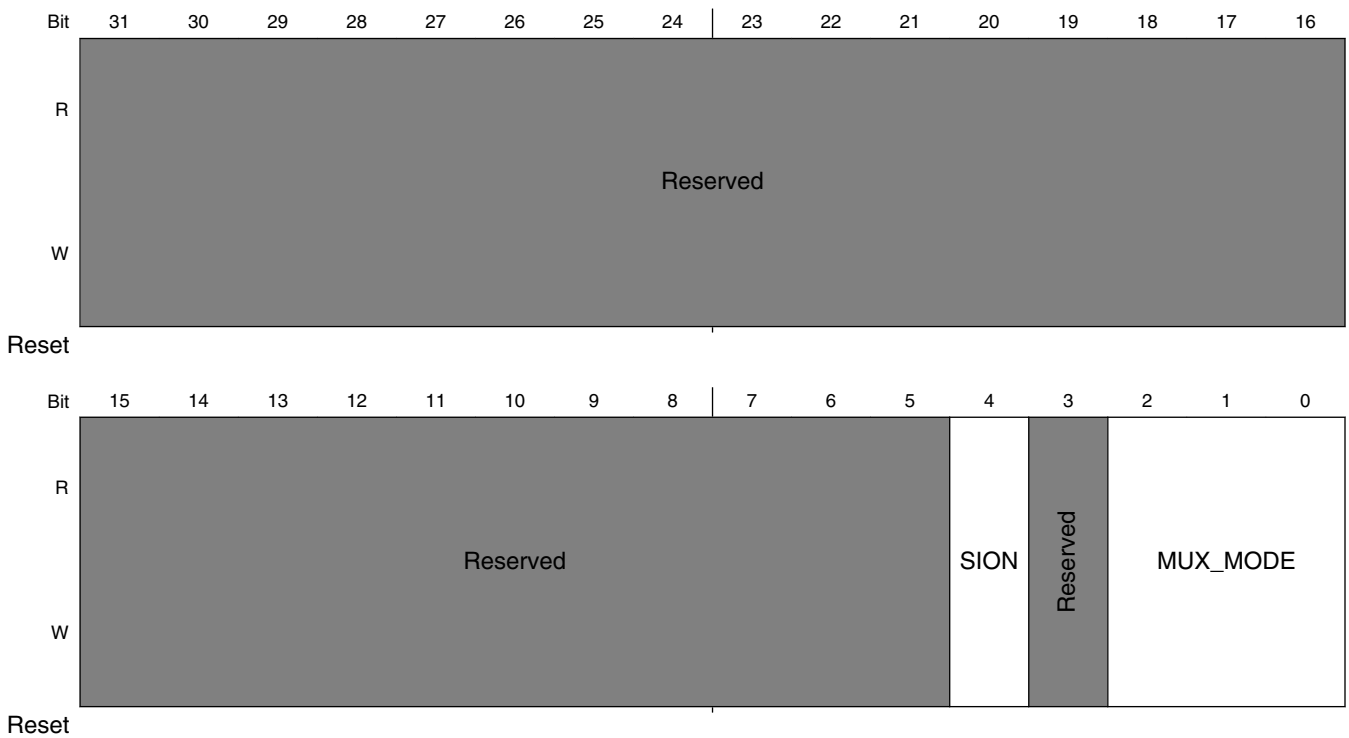
IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_01 field descriptions (continued)

Field	Description
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO_AD_01. 000 **ALT0** — Select mux mode: ALT0 mux port: LPUART4_RXD of instance: LPUART4 001 **ALT1** — Select mux mode: ALT1 mux port: LPSPI2_PCS1 of instance: LPSPI2 010 **ALT2** — Select mux mode: ALT2 mux port: WDOG1_ANY of instance: WDOG1 011 **ALT3** — Select mux mode: ALT3 mux port: LPI2C2_SDA of instance: LPI2C2 100 **ALT4** — Select mux mode: ALT4 mux port: MQS_LEFT of instance: MQS 101 **ALT5** — Select mux mode: ALT5 mux port: GPIOMUX_IO15 of instance: GPIOMUX 110 **ALT6** — Select mux mode: ALT6 mux port: USB_OTG1_OC of instance: USB 111 **ALT7** — Select mux mode: ALT7 mux port: ARM_CM7_TRACE_SWO of instance: cm7_mxrt

11.7.15 SW_MUX_CTL_PAD_GPIO_AD_00 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_00)

SW_MUX_CTL Register

Address: 401F_8000h base + 48h offset = 401F_8048h



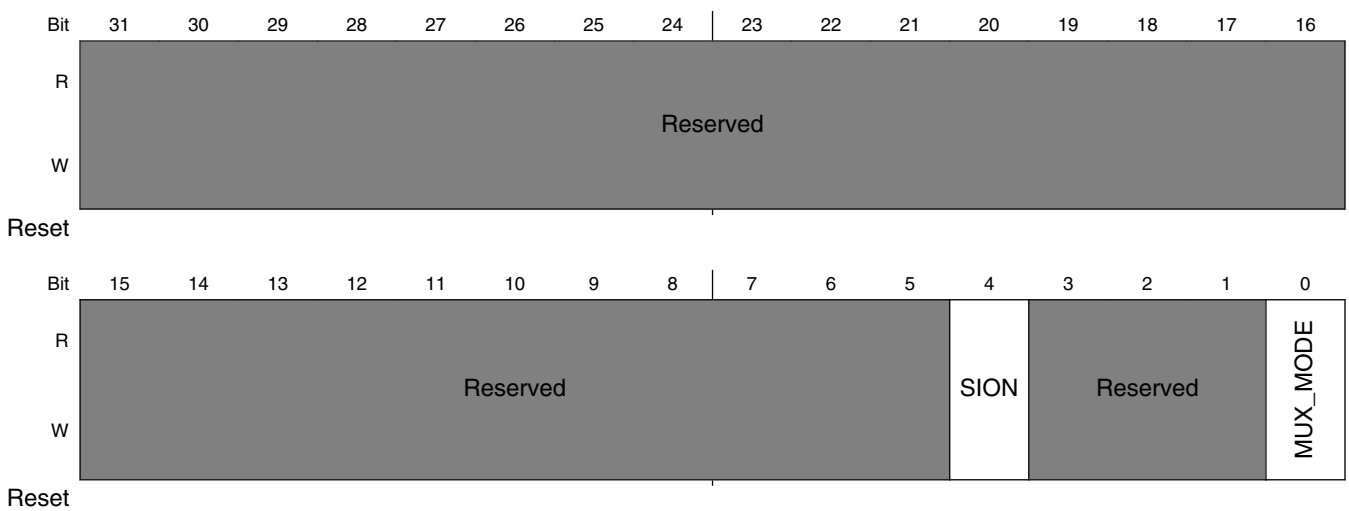
IOMUXC_SW_MUX_CTL_PAD_GPIO_AD_00 field descriptions

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_00 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO_AD_00. 000 **ALT0** — Select mux mode: ALT0 mux port: LPUART2_TXD of instance: LPUART2 001 **ALT1** — Select mux mode: ALT1 mux port: LPSPI1_PCS2 of instance: LPSPI1 010 **ALT2** — Select mux mode: ALT2 mux port: KPP_COL03 of instance: KPP 011 **ALT3** — Select mux mode: ALT3 mux port: USB_OTG1_PWR of instance: USB 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO1_IO20 of instance: FLEXIO1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIOMUX_IO14 of instance: GPIOMUX 110 **ALT6** — Select mux mode: ALT6 mux port: NMI_GLUE_NMI of instance: NMI_GLUE 111 **ALT7** — Select mux mode: ALT7 mux port: ARM_CM7_TRACE00 of instance: cm7_mxrt

11.7.16 SW_MUX_CTL_PAD_GPIO_SD_14 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_14)

SW_MUX_CTL Register

Address: 401F_8000h base + 4Ch offset = 401F_804Ch



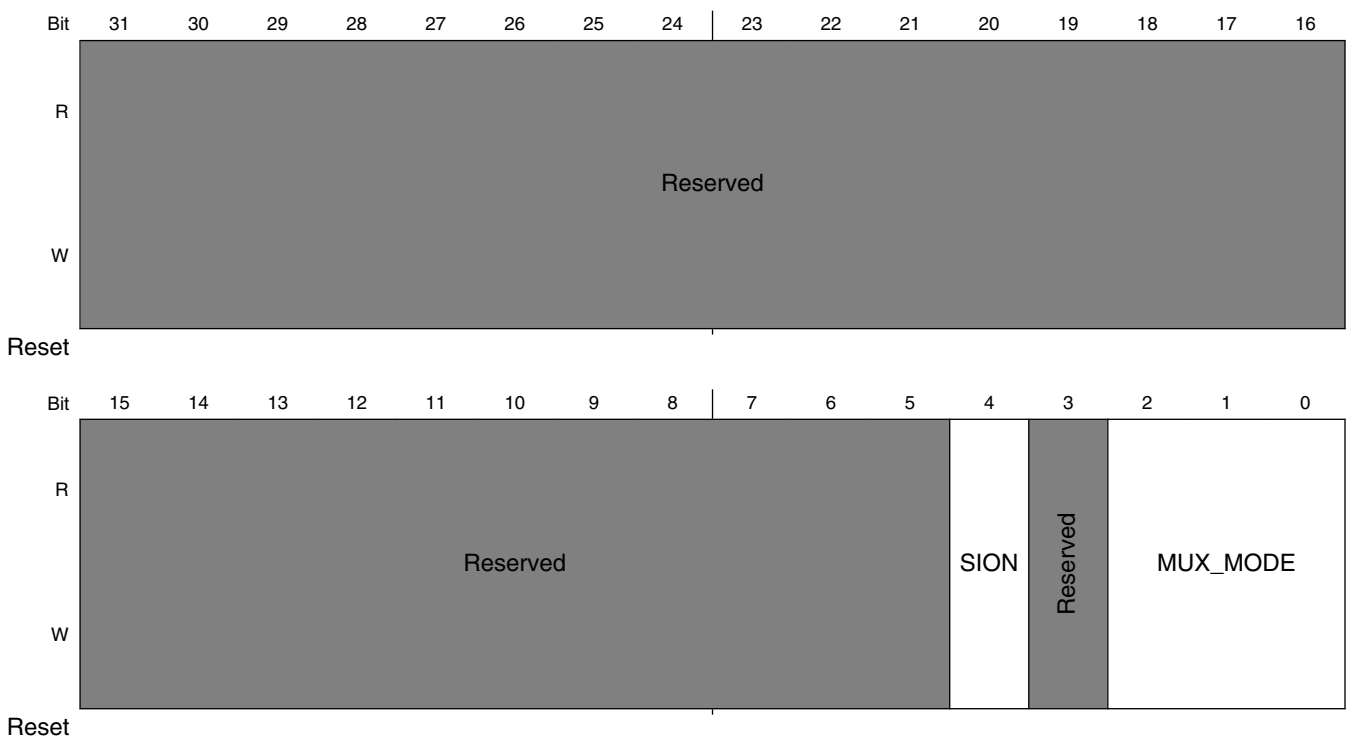
IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_14 field descriptions

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_SD_14 0 **DISABLED** — Input Path is determined by functionality
3–1 -	This field is reserved. Reserved
0 MUX_MODE	MUX Mode Select Field. Select 1 of 2 iomux modes to be used for pad: GPIO_SD_14. 00 **ALT0** — Select mux mode: ALT0 mux port: FLEXSPI_A_DQS of instance: FLEXSPI 01 **ALT1** — Select mux mode: ALT1 mux port: FLEXSPI_B_DQS of instance: FLEXSPI

11.7.17 SW_MUX_CTL_PAD_GPIO_SD_13 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_13)

SW_MUX_CTL Register

Address: 401F_8000h base + 50h offset = 401F_8050h



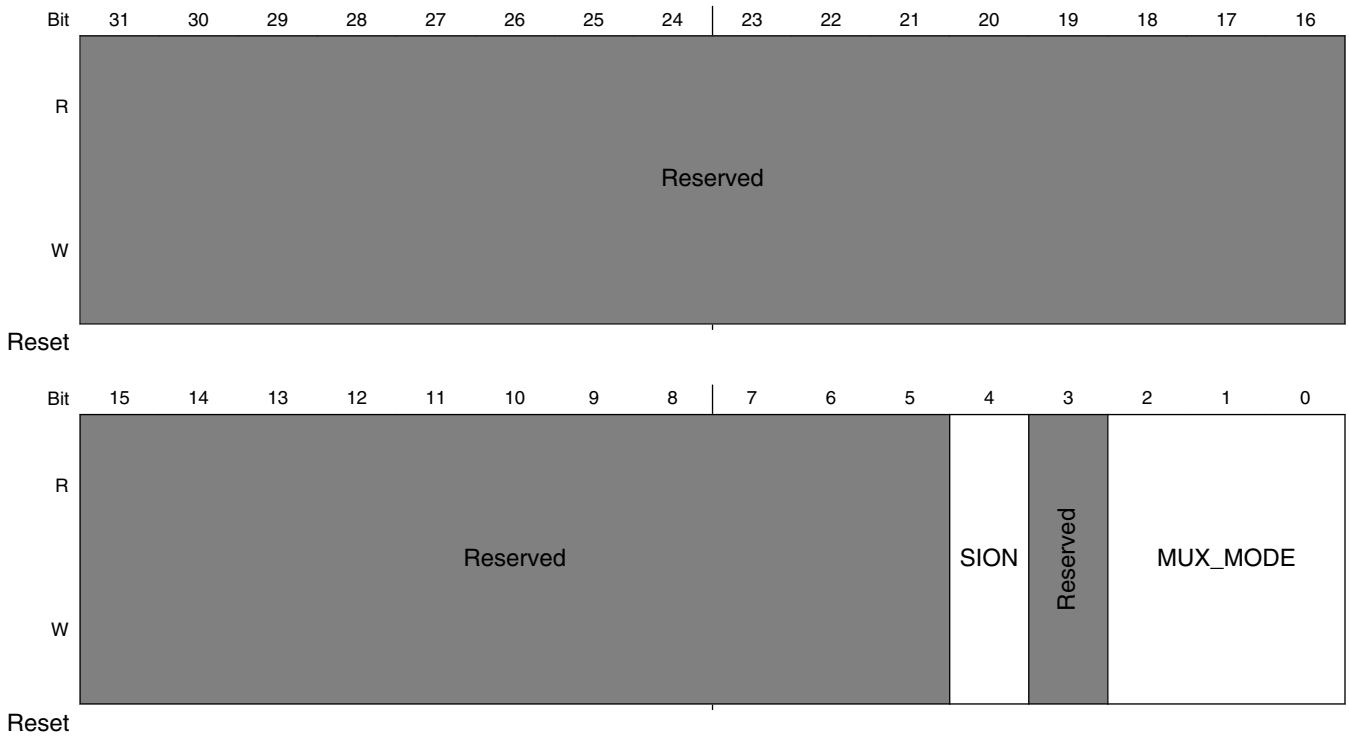
IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_13 field descriptions

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_SD_13 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO_SD_13. 000 **ALT0** — Select mux mode: ALT0 mux port: FLEXSPI_B_SCLK of instance: FLEXSPI 001 **ALT1** — Select mux mode: ALT1 mux port: SAI3_RX_BCLK of instance: SAI3 010 **ALT2** — Select mux mode: ALT2 mux port: ARM_CM7_TXEV of instance: cm7_mxrt 011 **ALT3** — Select mux mode: ALT3 mux port: CCM_PMIC_RDY of instance: CCM 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO1_IO19 of instance: FLEXIO1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO13 of instance: GPIO2 110 **ALT6** — Select mux mode: ALT6 mux port: SRC_BT_CFG03 of instance: SRC

11.7.18 SW_MUX_CTL_PAD_GPIO_SD_12 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_12)

SW_MUX_CTL Register

Address: 401F_8000h base + 54h offset = 401F_8054h



IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_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_SD_12 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO_SD_12. 000 **ALT0** — Select mux mode: ALT0 mux port: FLEXSPI_A_DQS of instance: FLEXSPI 001 **ALT1** — Select mux mode: ALT1 mux port: LPSPI2_PCS0 of instance: LPSPI2 010 **ALT2** — Select mux mode: ALT2 mux port: LPUART1_TXD of instance: LPUART1

Table continues on the next page...

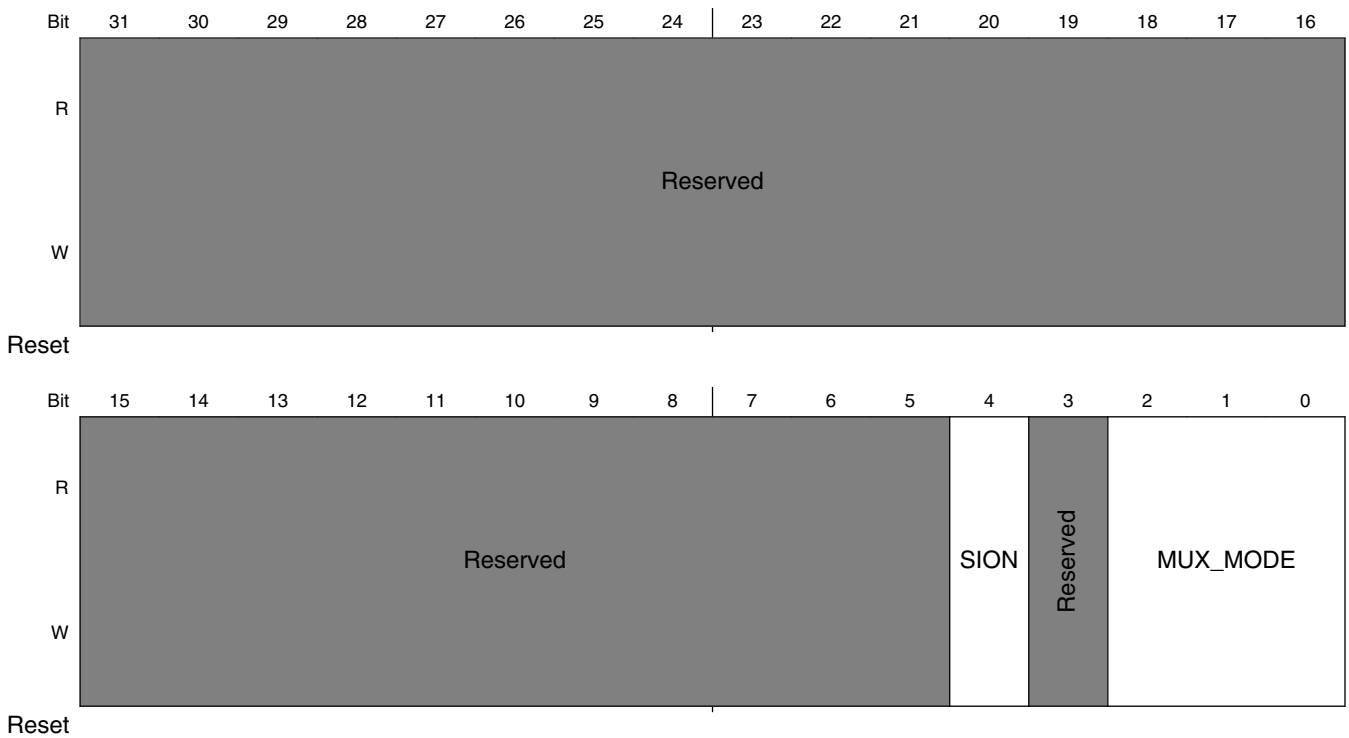
IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_12 field descriptions (continued)

Field	Description
	100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO1_IO18 of instance: FLEXIO1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO12 of instance: GPIO2 110 **ALT6** — Select mux mode: ALT6 mux port: WDOG2_RST_B_DEB of instance: WDOG2

11.7.19 SW_MUX_CTL_PAD_GPIO_SD_11 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_11)

SW_MUX_CTL Register

Address: 401F_8000h base + 58h offset = 401F_8058h



IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_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_SD_11 0 **DISABLED** — Input Path is determined by functionality

Table continues on the next page...

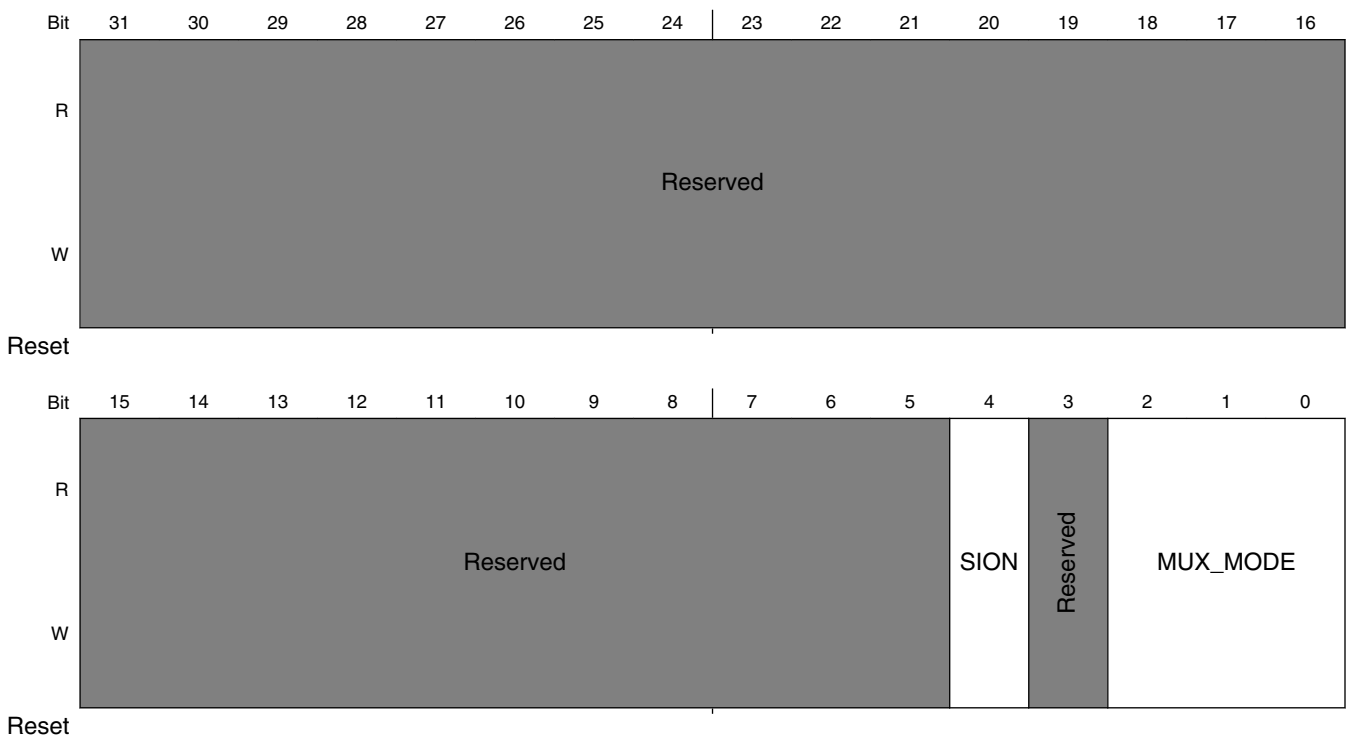
IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_11 field descriptions (continued)

Field	Description
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO_SD_11. 000 **ALT0** — Select mux mode: ALT0 mux port: FLEXSPI_A_DATA3 of instance: FLEXSPI 001 **ALT1** — Select mux mode: ALT1 mux port: LPSPI2_SCK of instance: LPSPI2 010 **ALT2** — Select mux mode: ALT2 mux port: LPUART1_RXD of instance: LPUART1 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO1_IO17 of instance: FLEXIO1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO11 of instance: GPIO2 110 **ALT6** — Select mux mode: ALT6 mux port: WDOG1_RST_B_DEB of instance: WDOG1

11.7.20 SW_MUX_CTL_PAD_GPIO_SD_10 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_10)

SW_MUX_CTL Register

Address: 401F_8000h base + 5Ch offset = 401F_805Ch



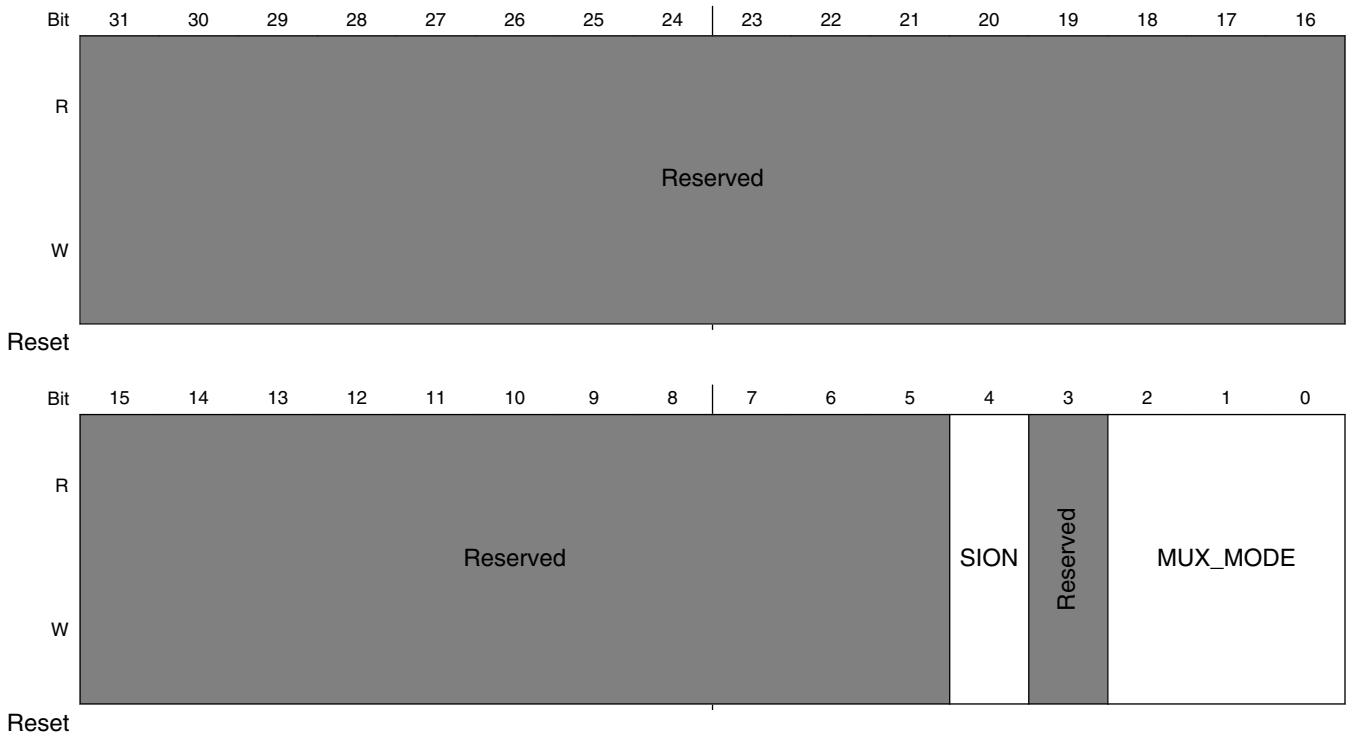
IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_10 field descriptions

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_10 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO_SD_10. 000 **ALT0** — Select mux mode: ALT0 mux port: FLEXSPI_A_SCLK of instance: FLEXSPI 001 **ALT1** — Select mux mode: ALT1 mux port: LPSPI2_SDO of instance: LPSPI2 010 **ALT2** — Select mux mode: ALT2 mux port: LPUART2_TXD of instance: LPUART2 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO1_IO16 of instance: FLEXIO1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO10 of instance: GPIO2

11.7.21 SW_MUX_CTL_PAD_GPIO_SD_09 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_09)

SW_MUX_CTL Register

Address: 401F_8000h base + 60h offset = 401F_8060h



IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_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_09 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO_SD_09. 000 **ALT0** — Select mux mode: ALT0 mux port: FLEXSPI_A_DATA0 of instance: FLEXSPI 001 **ALT1** — Select mux mode: ALT1 mux port: LPSPI2_SDI of instance: LPSPI2 010 **ALT2** — Select mux mode: ALT2 mux port: LPUART2_RXD of instance: LPUART2

Table continues on the next page...

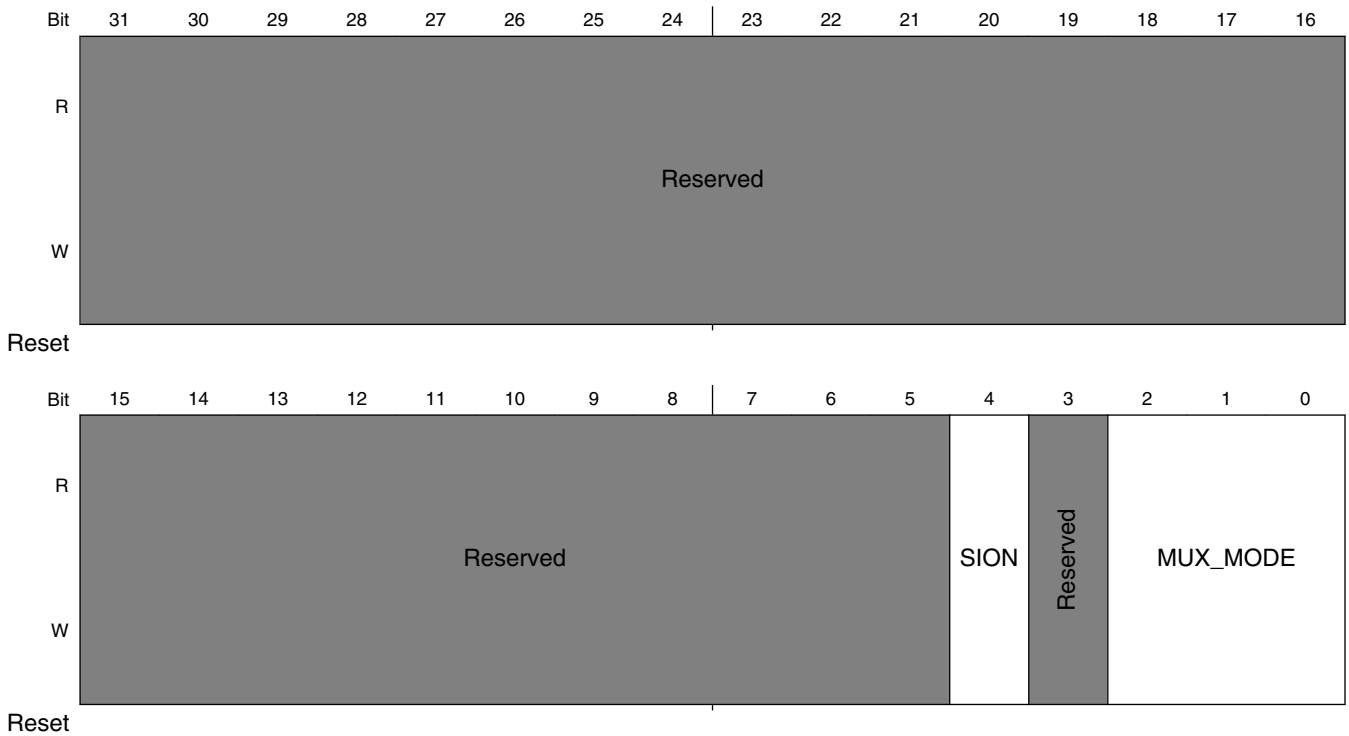
IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_09 field descriptions (continued)

Field	Description
	100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO1_IO15 of instance: FLEXIO1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO09 of instance: GPIO2

11.7.22 SW_MUX_CTL_PAD_GPIO_SD_08 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_08)

SW_MUX_CTL Register

Address: 401F_8000h base + 64h offset = 401F_8064h



IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_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_SD_08 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved

Table continues on the next page...

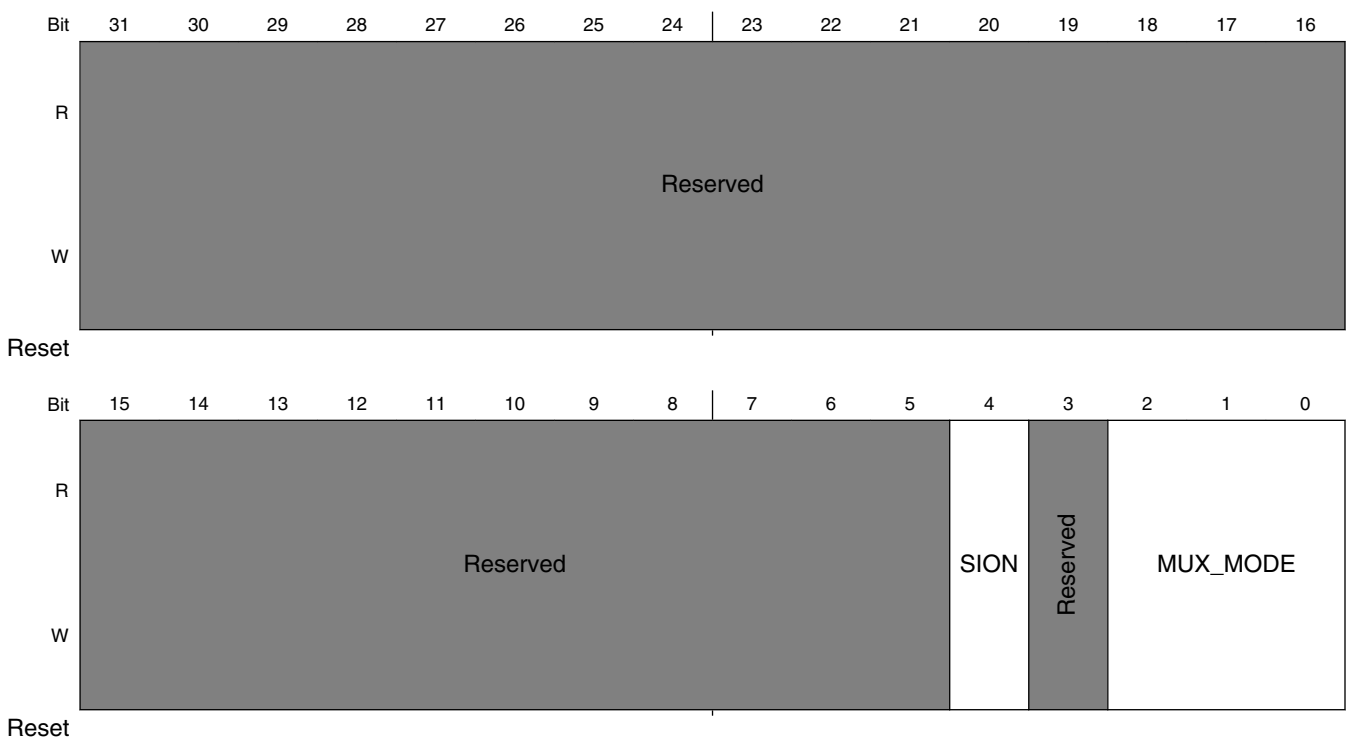
IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_08 field descriptions (continued)

Field	Description
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO_SD_08. 000 **ALT0** — Select mux mode: ALT0 mux port: FLEXSPI_A_DATA2 of instance: FLEXSPI 001 **ALT1** — Select mux mode: ALT1 mux port: LPI2C2_SCL of instance: LPI2C2 010 **ALT2** — Select mux mode: ALT2 mux port: LPSPI1_SCK of instance: LPSPI1 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO1_IO14 of instance: FLEXIO1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO08 of instance: GPIO2

11.7.23 SW_MUX_CTL_PAD_GPIO_SD_07 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_07)

SW_MUX_CTL Register

Address: 401F_8000h base + 68h offset = 401F_8068h



IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_07 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved

Table continues on the next page...

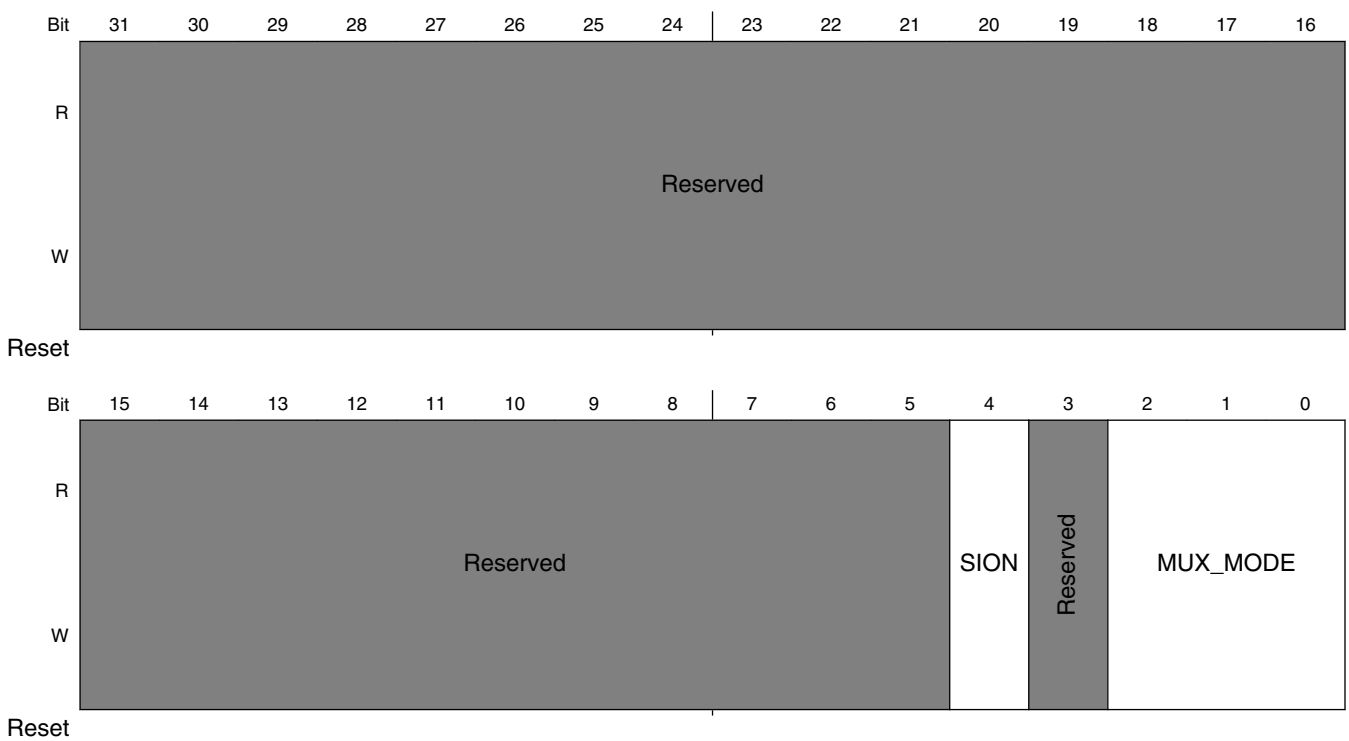
IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_07 field descriptions (continued)

Field	Description
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_07 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO_SD_07. 000 **ALT0** — Select mux mode: ALT0 mux port: FLEXSPI_A_DATA1 of instance: FLEXSPI 001 **ALT1** — Select mux mode: ALT1 mux port: LPI2C2_SDA of instance: LPI2C2 010 **ALT2** — Select mux mode: ALT2 mux port: LPSPI1_PCS0 of instance: LPSPI1 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO1_IO13 of instance: FLEXIO1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO07 of instance: GPIO2

11.7.24 SW_MUX_CTL_PAD_GPIO_SD_06 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_06)

SW_MUX_CTL Register

Address: 401F_8000h base + 6Ch offset = 401F_806Ch



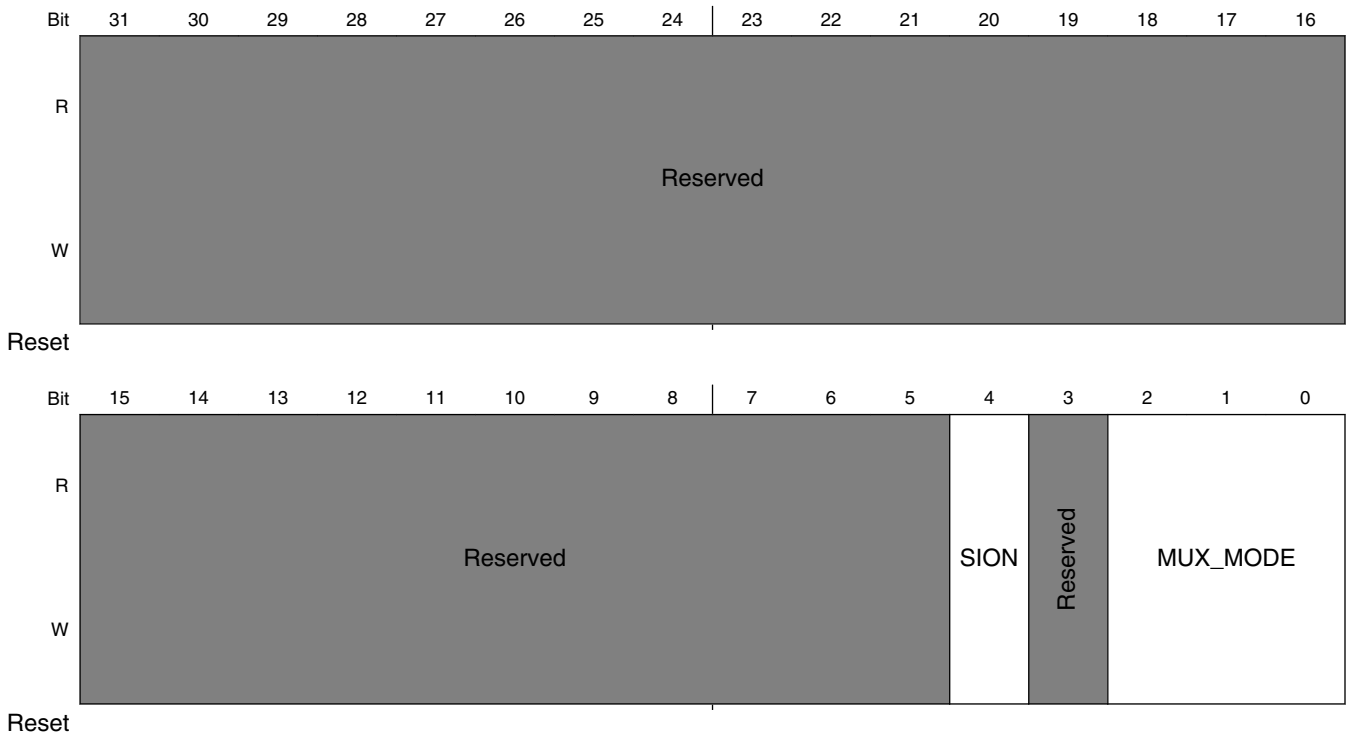
IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_06 field descriptions

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_06 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO_SD_06. 000 **ALT0** — Select mux mode: ALT0 mux port: FLEXSPI_A_SS0_B of instance: FLEXSPI 001 **ALT1** — Select mux mode: ALT1 mux port: LPI2C1_SCL of instance: LPI2C1 010 **ALT2** — Select mux mode: ALT2 mux port: LPSPI1_SDO of instance: LPSPI1 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO1_IO12 of instance: FLEXIO1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO06 of instance: GPIO2

11.7.25 SW_MUX_CTL_PAD_GPIO_SD_05 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_05)

SW_MUX_CTL Register

Address: 401F_8000h base + 70h offset = 401F_8070h



IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_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_05 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO_SD_05. 000 **ALT0** — Select mux mode: ALT0 mux port: FLEXSPI_A_SS1_B of instance: FLEXSPI 001 **ALT1** — Select mux mode: ALT1 mux port: LPI2C1_SDA of instance: LPI2C1 010 **ALT2** — Select mux mode: ALT2 mux port: LPSPI1_SDI of instance: LPSPI1

Table continues on the next page...

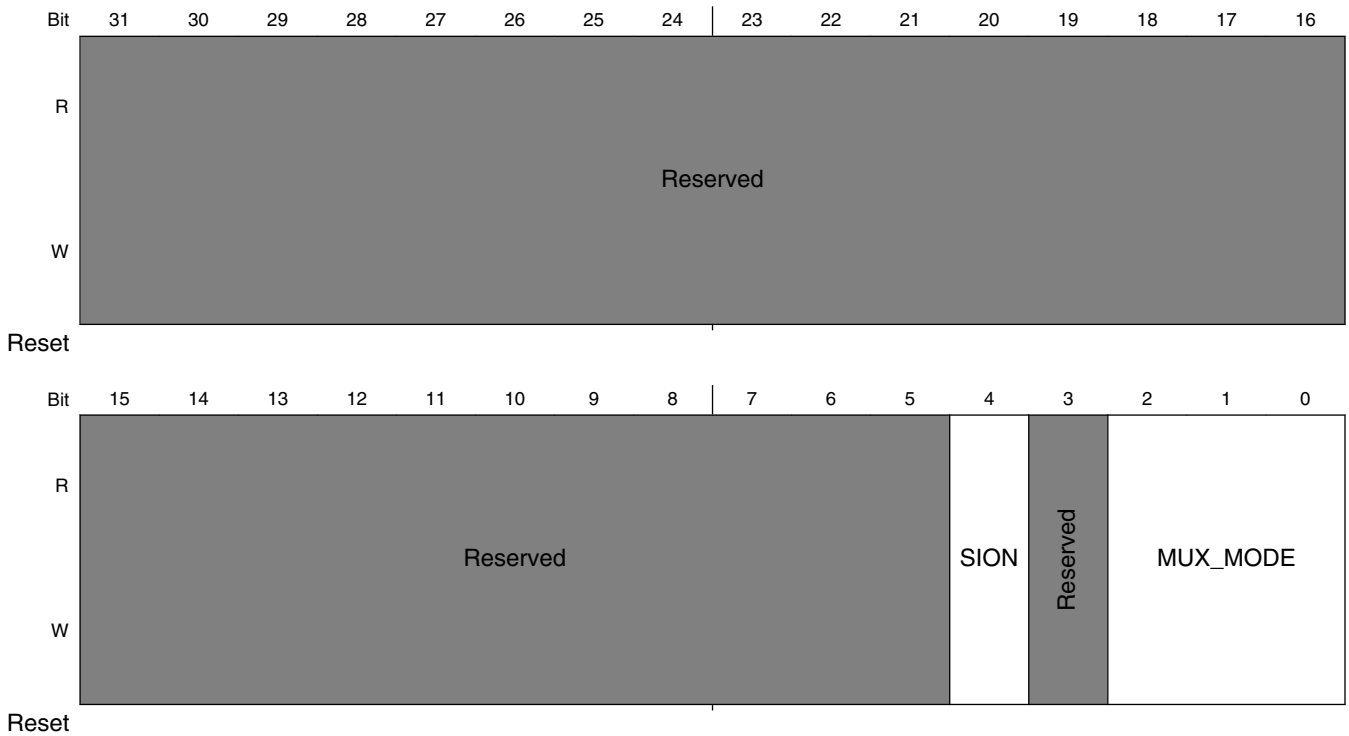
IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_05 field descriptions (continued)

Field	Description
	100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO1_IO11 of instance: FLEXIO1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO05 of instance: GPIO2

11.7.26 SW_MUX_CTL_PAD_GPIO_SD_04 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_04)

SW_MUX_CTL Register

Address: 401F_8000h base + 74h offset = 401F_8074h



IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_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_04 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved

Table continues on the next page...

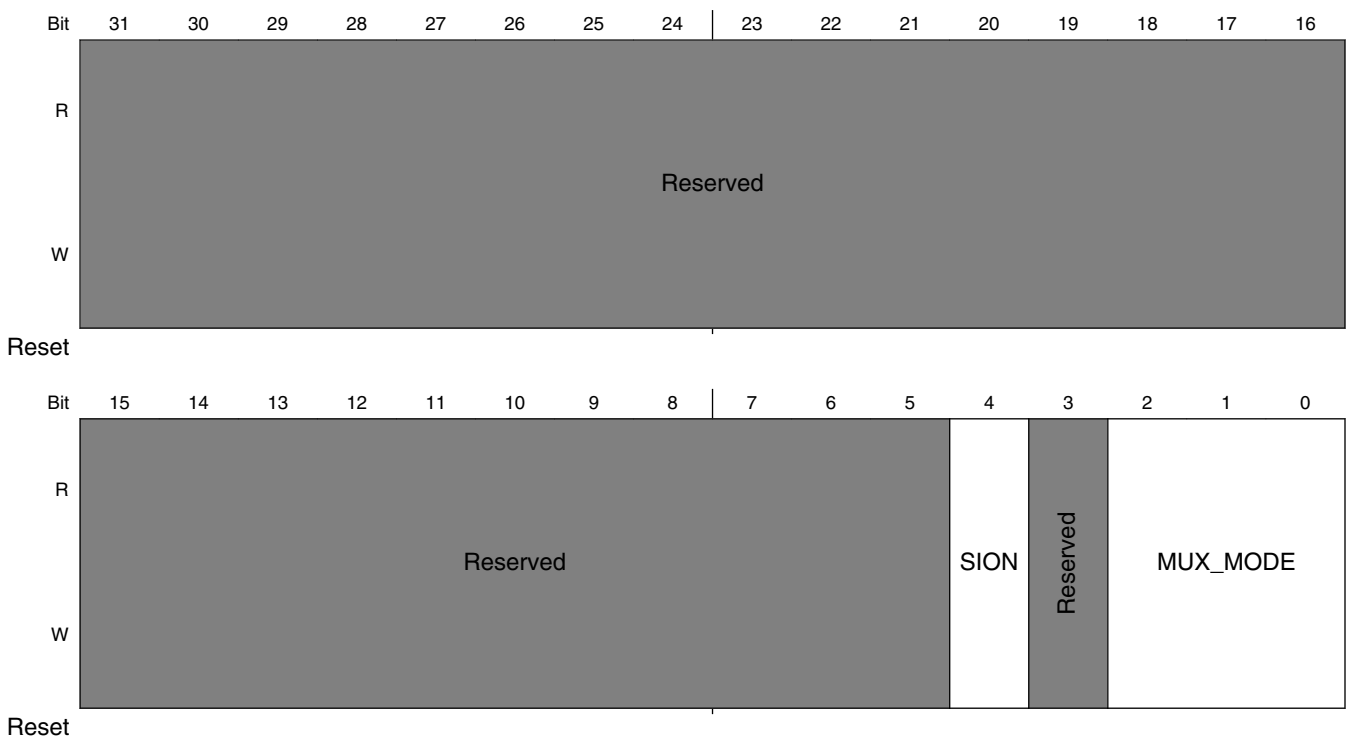
IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_04 field descriptions (continued)

Field	Description
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO_SD_04. 010 **ALT2** — Select mux mode: ALT2 mux port: FLEXPWM1_PWM1_A of instance: FLEXPWM1 011 **ALT3** — Select mux mode: ALT3 mux port: CCM_WAIT of instance: CCM 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO1_IO10 of instance: FLEXIO1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO04 of instance: GPIO2 110 **ALT6** — Select mux mode: ALT6 mux port: SRC_BOOT_MODE00 of instance: SRC 000 **ALT0** — Select mux mode: ALT0 mux port: FLEXSPI_B_DATA03 of instance: FLEXSPI 001 **ALT1** — Select mux mode: ALT1 mux port: SAI3_RX_SYNC of instance: SAI3

11.7.27 SW_MUX_CTL_PAD_GPIO_SD_03 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_03)

SW_MUX_CTL Register

Address: 401F_8000h base + 78h offset = 401F_8078h



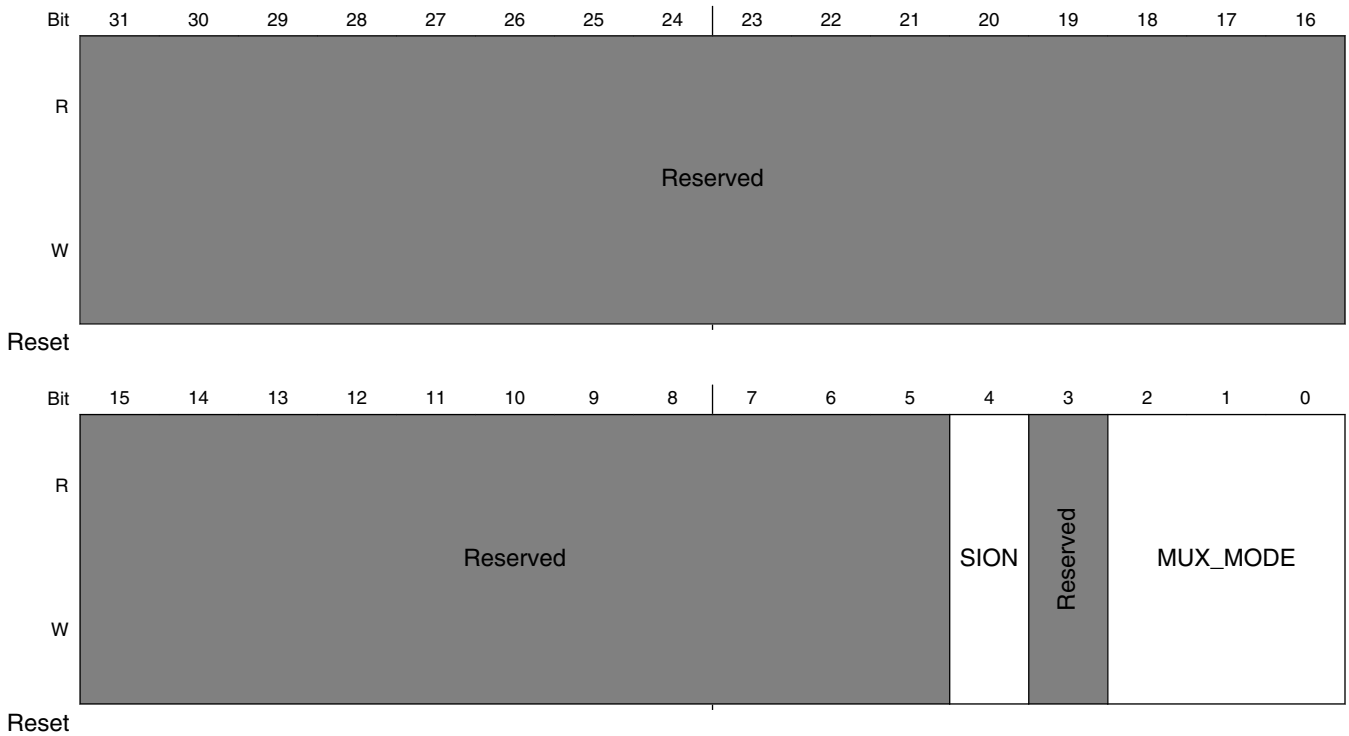
IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_03 field descriptions

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_03 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO_SD_03. 000 **ALT0** — Select mux mode: ALT0 mux port: FLEXSPI_B_DATA00 of instance: FLEXSPI 001 **ALT1** — Select mux mode: ALT1 mux port: SAI3_RX_DATA of instance: SAI3 010 **ALT2** — Select mux mode: ALT2 mux port: FLEXPWM1_PWM1_B of instance: FLEXPWM1 011 **ALT3** — Select mux mode: ALT3 mux port: CCM_REF_EN_B of instance: CCM 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO1_IO09 of instance: FLEXIO1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO03 of instance: GPIO2 110 **ALT6** — Select mux mode: ALT6 mux port: SRC_BOOT_MODE01 of instance: SRC

11.7.28 SW_MUX_CTL_PAD_GPIO_SD_02 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_02)

SW_MUX_CTL Register

Address: 401F_8000h base + 7Ch offset = 401F_807Ch



IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_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_02 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO_SD_02. 000 **ALT0** — Select mux mode: ALT0 mux port: FLEXSPI_B_DATA02 of instance: FLEXSPI 001 **ALT1** — Select mux mode: ALT1 mux port: SAI3_TX_DATA of instance: SAI3 010 **ALT2** — Select mux mode: ALT2 mux port: FLEXPWM1_PWM0_A of instance: FLEXPWM1

Table continues on the next page...

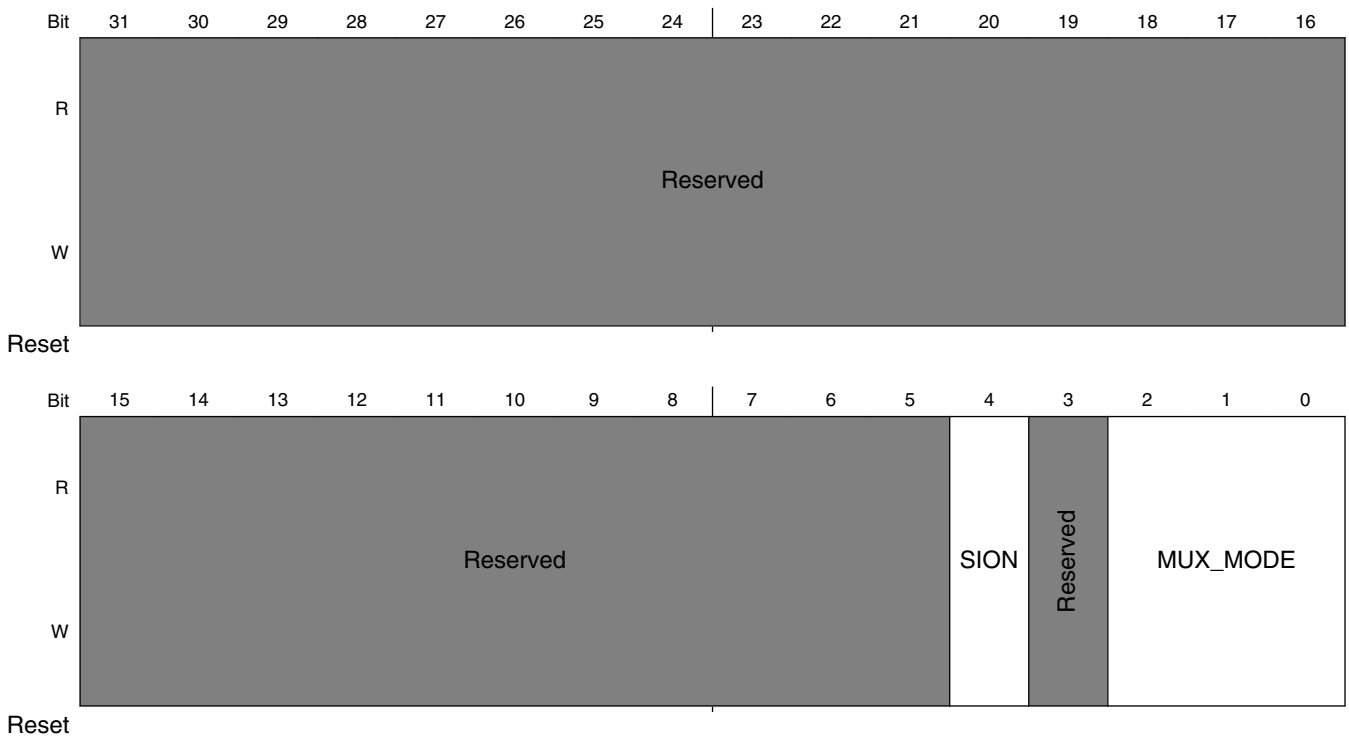
IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_02 field descriptions (continued)

Field	Description
	011 **ALT3** — Select mux mode: ALT3 mux port: CCM_CLKO1 of instance: CCM 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO1_IO08 of instance: FLEXIO1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO02 of instance: GPIO2 110 **ALT6** — Select mux mode: ALT6 mux port: SRC_BT_CFG00 of instance: SRC

11.7.29 SW_MUX_CTL_PAD_GPIO_SD_01 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_01)

SW_MUX_CTL Register

Address: 401F_8000h base + 80h offset = 401F_8080h



IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_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_01 0 **DISABLED** — Input Path is determined by functionality

Table continues on the next page...

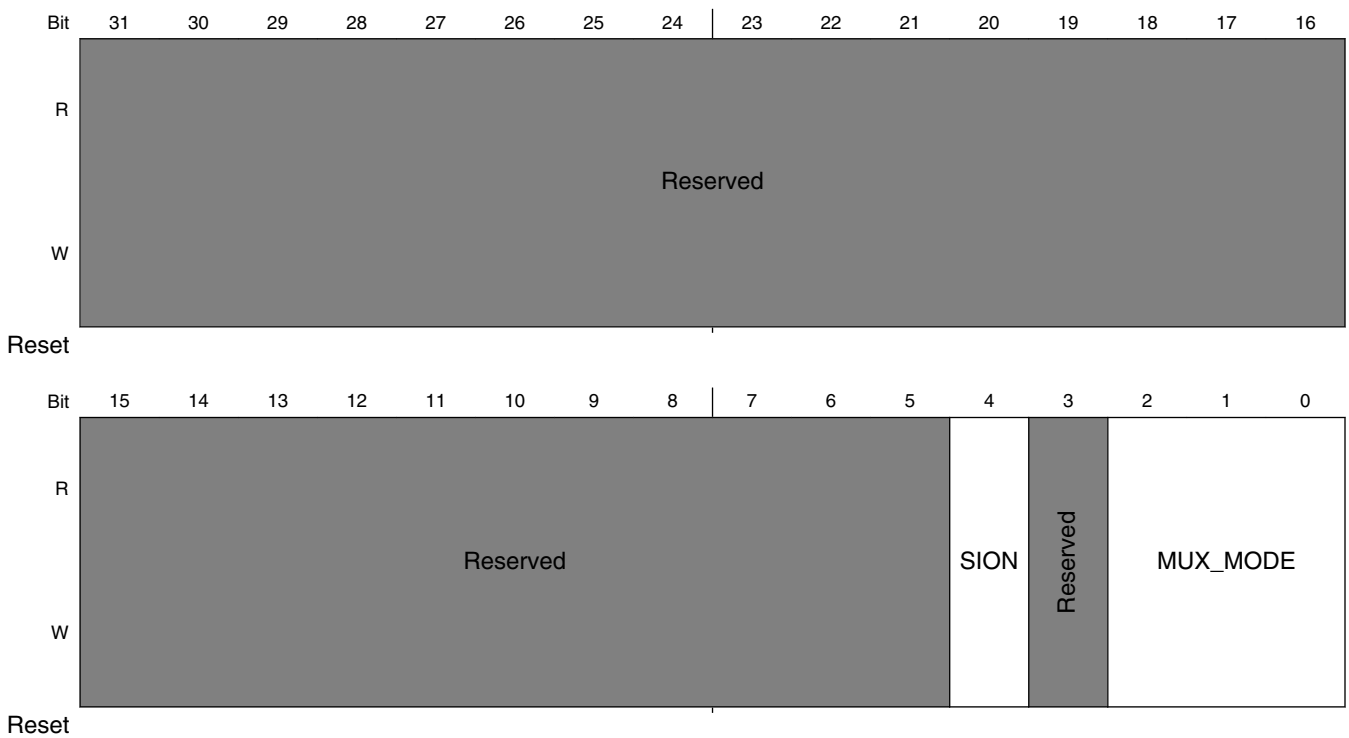
IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_01 field descriptions (continued)

Field	Description
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO_SD_01. 000 **ALT0** — Select mux mode: ALT0 mux port: FLEXSPI_B_DATA01 of instance: FLEXSPI 001 **ALT1** — Select mux mode: ALT1 mux port: SAI3_TX_BCLK of instance: SAI3 010 **ALT2** — Select mux mode: ALT2 mux port: FLEXPWM1_PWM0_B of instance: FLEXPWM1 011 **ALT3** — Select mux mode: ALT3 mux port: CCM_CLKO2 of instance: CCM 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO1_IO07 of instance: FLEXIO1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO01 of instance: GPIO2 110 **ALT6** — Select mux mode: ALT6 mux port: SRC_BT_CFG01 of instance: SRC

11.7.30 SW_MUX_CTL_PAD_GPIO_SD_00 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_00)

SW_MUX_CTL Register

Address: 401F_8000h base + 84h offset = 401F_8084h



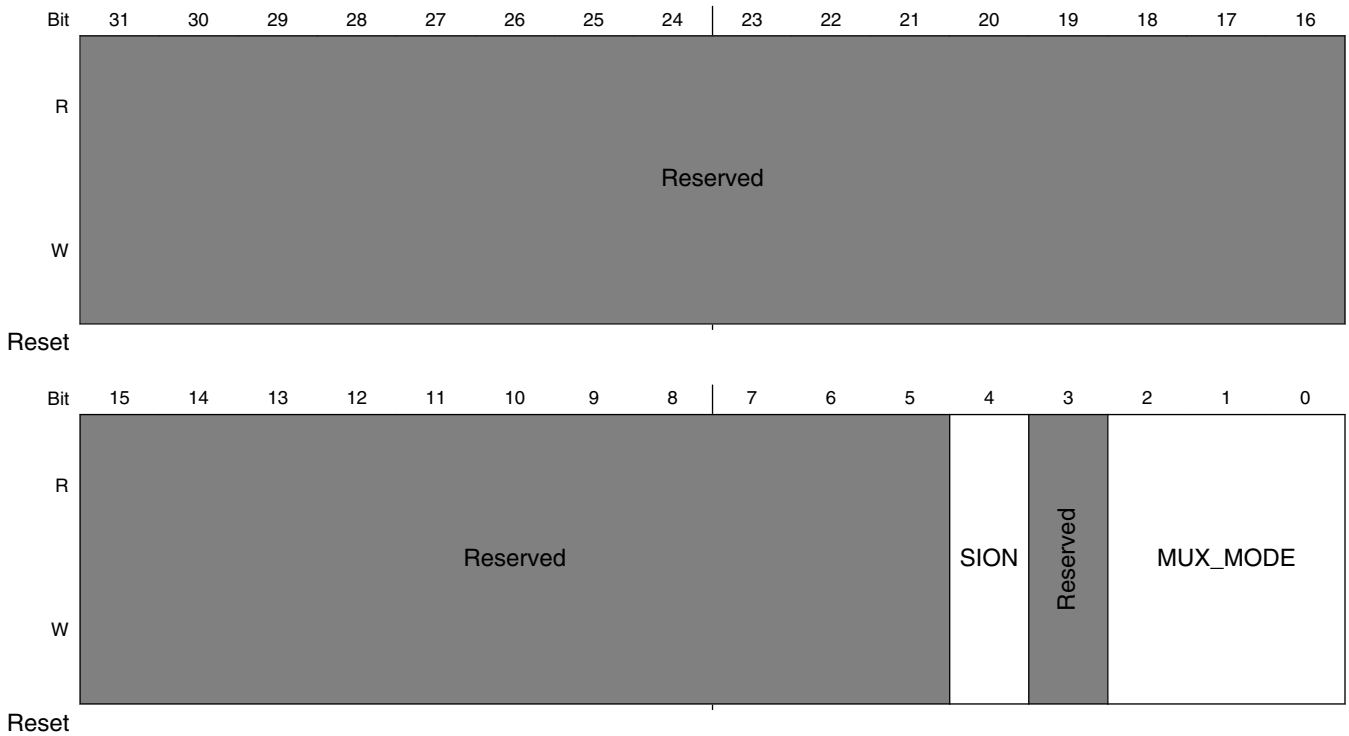
IOMUXC_SW_MUX_CTL_PAD_GPIO_SD_00 field descriptions

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_00 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO_SD_00. 000 **ALT0** — Select mux mode: ALT0 mux port: FLEXSPI_B_SS0_B of instance: FLEXSPI 001 **ALT1** — Select mux mode: ALT1 mux port: SAI3_TX_SYNC of instance: SAI3 010 **ALT2** — Select mux mode: ALT2 mux port: ARM_CM7_RXEV of instance: cm7_mxrt 011 **ALT3** — Select mux mode: ALT3 mux port: CCM_STOP of instance: CCM 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO1_IO06 of instance: FLEXIO1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIO2_IO00 of instance: GPIO2 110 **ALT6** — Select mux mode: ALT6 mux port: SRC_BT_CFG02 of instance: SRC

11.7.31 SW_MUX_CTL_PAD_GPIO_13 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_13)

SW_MUX_CTL Register

Address: 401F_8000h base + 88h offset = 401F_8088h



IOMUXC_SW_MUX_CTL_PAD_GPIO_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_13 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO_13. 000 **ALT0** — Select mux mode: ALT0 mux port: LPUART2_RXD of instance: LPUART2 001 **ALT1** — Select mux mode: ALT1 mux port: LPSPI2_PCS2 of instance: LPSPI2 010 **ALT2** — Select mux mode: ALT2 mux port: KPP_ROW03 of instance: KPP

Table continues on the next page...

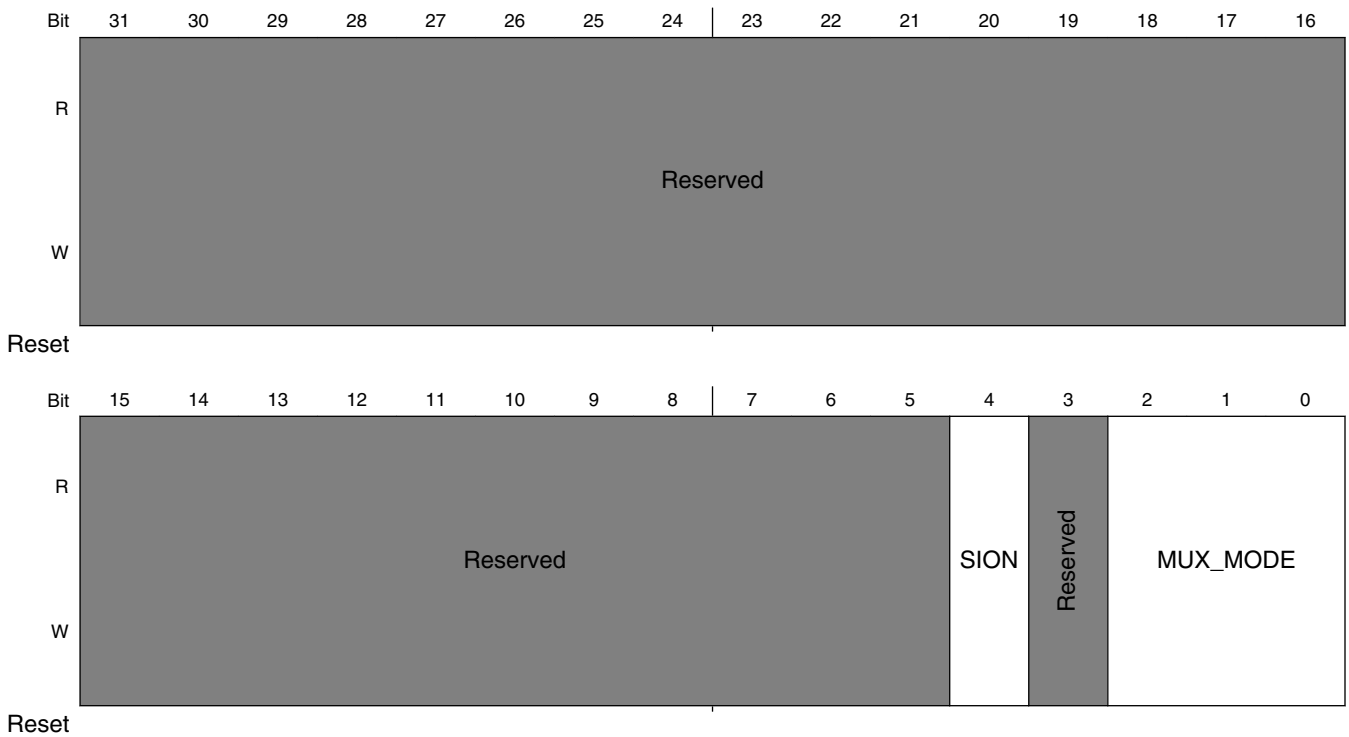
IOMUXC_SW_MUX_CTL_PAD_GPIO_13 field descriptions (continued)

Field	Description
	011 **ALT3** — Select mux mode: ALT3 mux port: OTG1_ID of instance: anatop 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO1_IO05 of instance: FLEXIO1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIOMUX_IO13 of instance: GPIOMUX 110 **ALT6** — Select mux mode: ALT6 mux port: SPDIF_LOCK of instance: SPDIF 111 **ALT7** — Select mux mode: ALT7 mux port: ARM_CM7_TRACE01 of instance: cm7_mxrt

11.7.32 SW_MUX_CTL_PAD_GPIO_12 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_12)

SW_MUX_CTL Register

Address: 401F_8000h base + 8Ch offset = 401F_808Ch



IOMUXC_SW_MUX_CTL_PAD_GPIO_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.

Table continues on the next page...

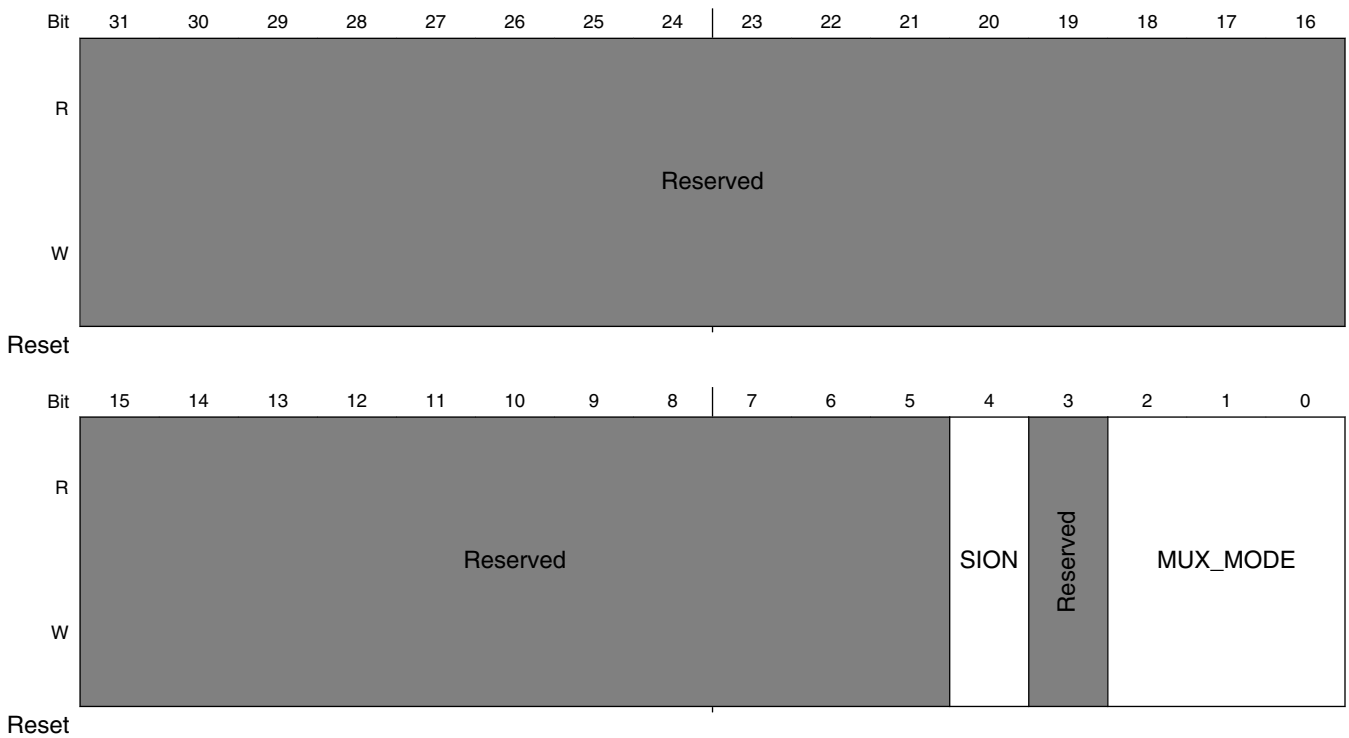
IOMUXC_SW_MUX_CTL_PAD_GPIO_12 field descriptions (continued)

Field	Description
	1 **ENABLED** — Force input path of pad GPIO_12 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO_12. 000 **ALT0** — Select mux mode: ALT0 mux port: LPUART3_TXD of instance: LPUART3 001 **ALT1** — Select mux mode: ALT1 mux port: LPI2C1_SCL of instance: LPI2C1 010 **ALT2** — Select mux mode: ALT2 mux port: KPP_COL00 of instance: KPP 011 **ALT3** — Select mux mode: ALT3 mux port: USB_OTG1_OC of instance: USB 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO1_IO04 of instance: FLEXIO1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIOMUX_IO12 of instance: GPIOMUX 110 **ALT6** — Select mux mode: ALT6 mux port: SPDIF_EXT_CLK of instance: SPDIF 111 **ALT7** — Select mux mode: ALT7 mux port: ARM_CM7_TRACE02 of instance: cm7_mxrt

11.7.33 SW_MUX_CTL_PAD_GPIO_11 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_11)

SW_MUX_CTL Register

Address: 401F_8000h base + 90h offset = 401F_8090h



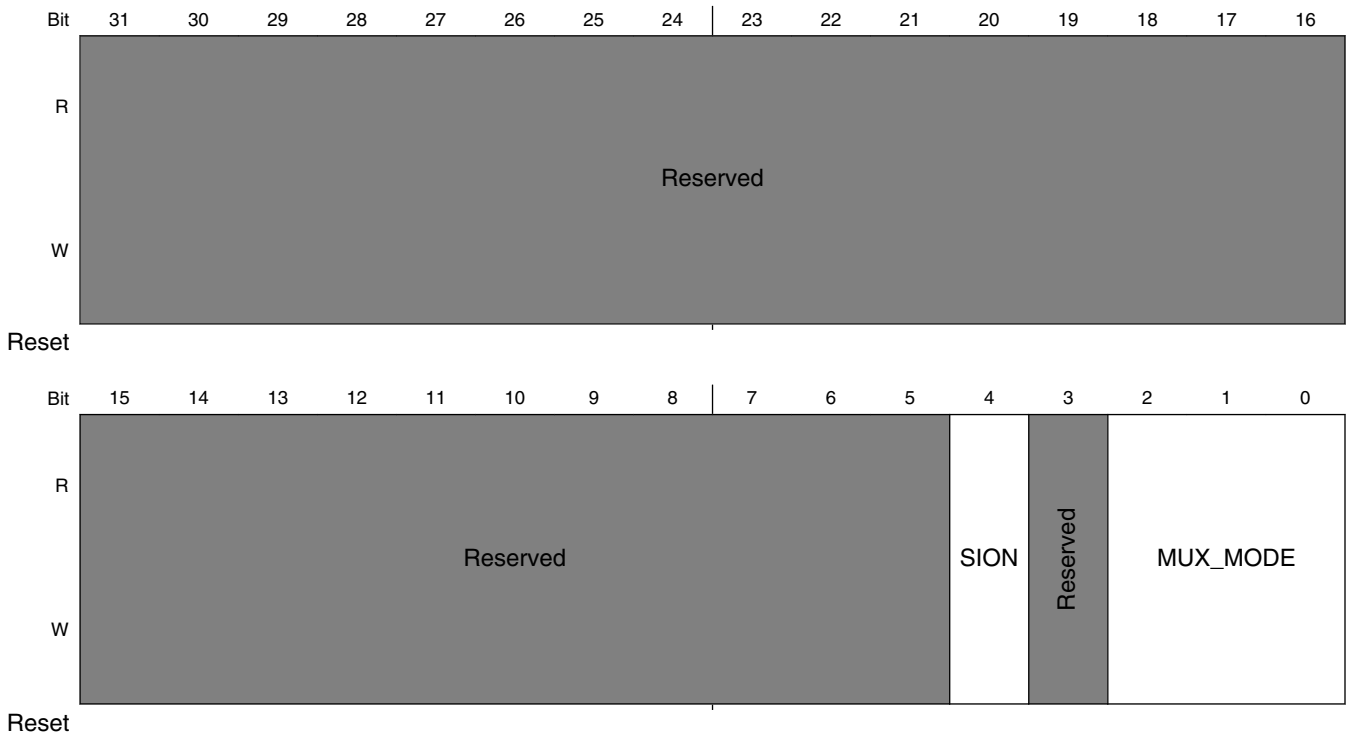
IOMUXC_SW_MUX_CTL_PAD_GPIO_11 field descriptions

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_11 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO_11. 000 **ALT0** — Select mux mode: ALT0 mux port: LPUART3_RXD of instance: LPUART3 001 **ALT1** — Select mux mode: ALT1 mux port: LPI2C1_SDA of instance: LPI2C1 010 **ALT2** — Select mux mode: ALT2 mux port: KPP_ROW00 of instance: KPP 011 **ALT3** — Select mux mode: ALT3 mux port: FLEXSPI_B_SS1_B of instance: FLEXSPI 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO1_IO03 of instance: FLEXIO1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIOMUX_IO11 of instance: GPIOMUX 110 **ALT6** — Select mux mode: ALT6 mux port: SPDIF_OUT of instance: SPDIF 111 **ALT7** — Select mux mode: ALT7 mux port: ARM_CM7_TRACE03 of instance: cm7_mxrt

11.7.34 SW_MUX_CTL_PAD_GPIO_10 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_10)

SW_MUX_CTL Register

Address: 401F_8000h base + 94h offset = 401F_8094h



IOMUXC_SW_MUX_CTL_PAD_GPIO_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_10 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO_10. 000 **ALT0** — Select mux mode: ALT0 mux port: LPUART1_TXD of instance: LPUART1 001 **ALT1** — Select mux mode: ALT1 mux port: LPI2C1_HREQ of instance: LPI2C1 010 **ALT2** — Select mux mode: ALT2 mux port:EWM_OUT_B of instance: EWM

Table continues on the next page...

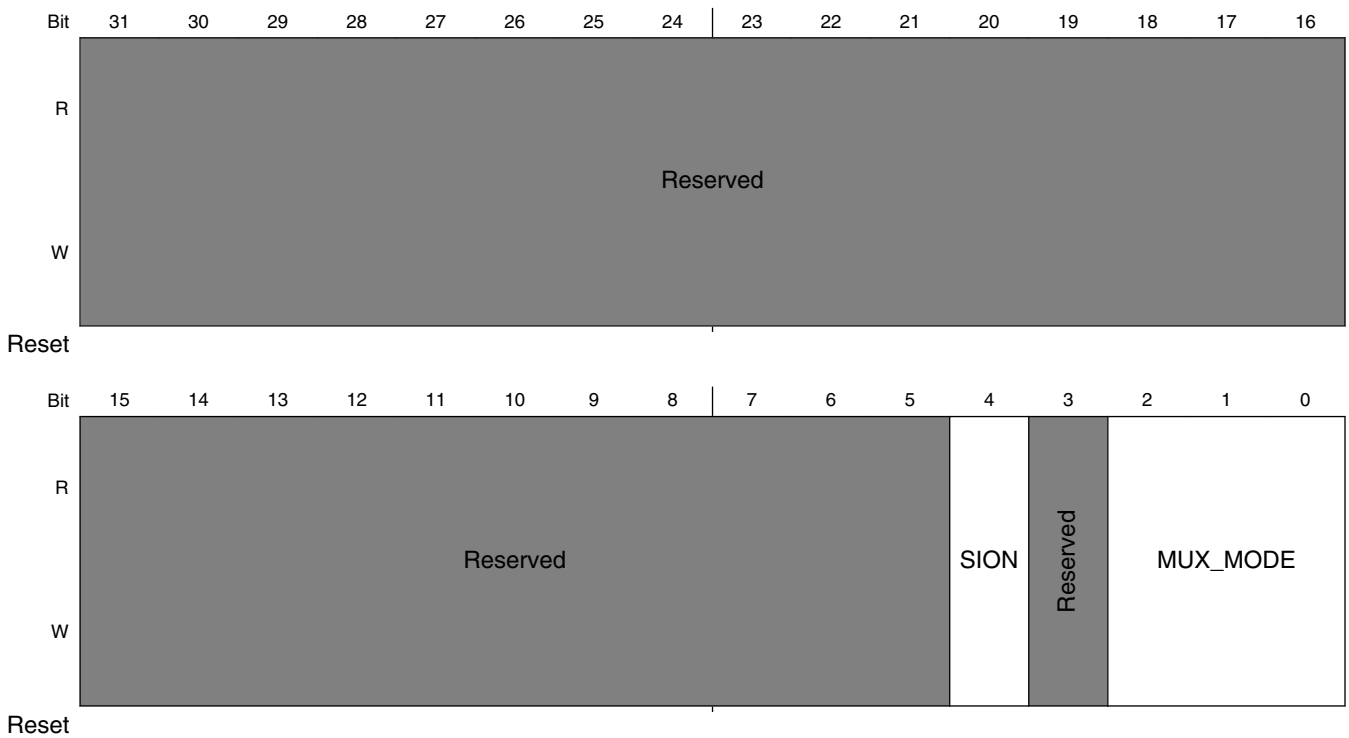
IOMUXC_SW_MUX_CTL_PAD_GPIO_10 field descriptions (continued)

Field	Description
	011 **ALT3** — Select mux mode: ALT3 mux port: LPI2C2_SCL of instance: LPI2C2 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO1_IO02 of instance: FLEXIO1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIOMUX_IO10 of instance: GPIOMUX 110 **ALT6** — Select mux mode: ALT6 mux port: SPDIF_IN of instance: SPDIF

11.7.35 SW_MUX_CTL_PAD_GPIO_09 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_09)

SW_MUX_CTL Register

Address: 401F_8000h base + 98h offset = 401F_8098h



IOMUXC_SW_MUX_CTL_PAD_GPIO_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_09 0 **DISABLED** — Input Path is determined by functionality

Table continues on the next page...

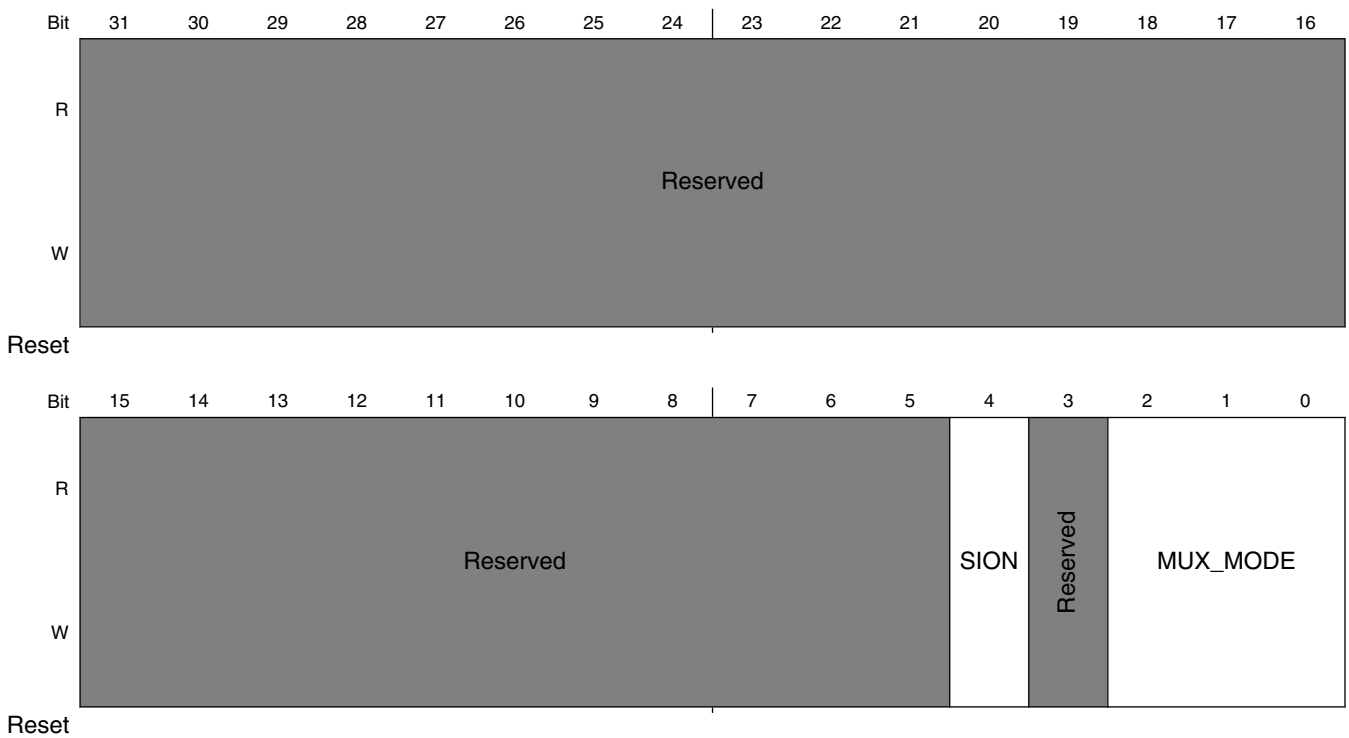
IOMUXC_SW_MUX_CTL_PAD_GPIO_09 field descriptions (continued)

Field	Description
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO_09. 000 **ALT0** — Select mux mode: ALT0 mux port: LPUART1_RXD of instance: LPUART1 001 **ALT1** — Select mux mode: ALT1 mux port: WDOG1_B of instance: WDOG1 010 **ALT2** — Select mux mode: ALT2 mux port: FLEXSPI_A_SS1_B of instance: FLEXSPI 011 **ALT3** — Select mux mode: ALT3 mux port: LPI2C2_SDA of instance: LPI2C2 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO1_IO01 of instance: FLEXIO1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIOMUX_IO09 of instance: GPIOMUX 110 **ALT6** — Select mux mode: ALT6 mux port: SPDIF_SR_CLK of instance: SPDIF

11.7.36 SW_MUX_CTL_PAD_GPIO_08 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_08)

SW_MUX_CTL Register

Address: 401F_8000h base + 9Ch offset = 401F_809Ch



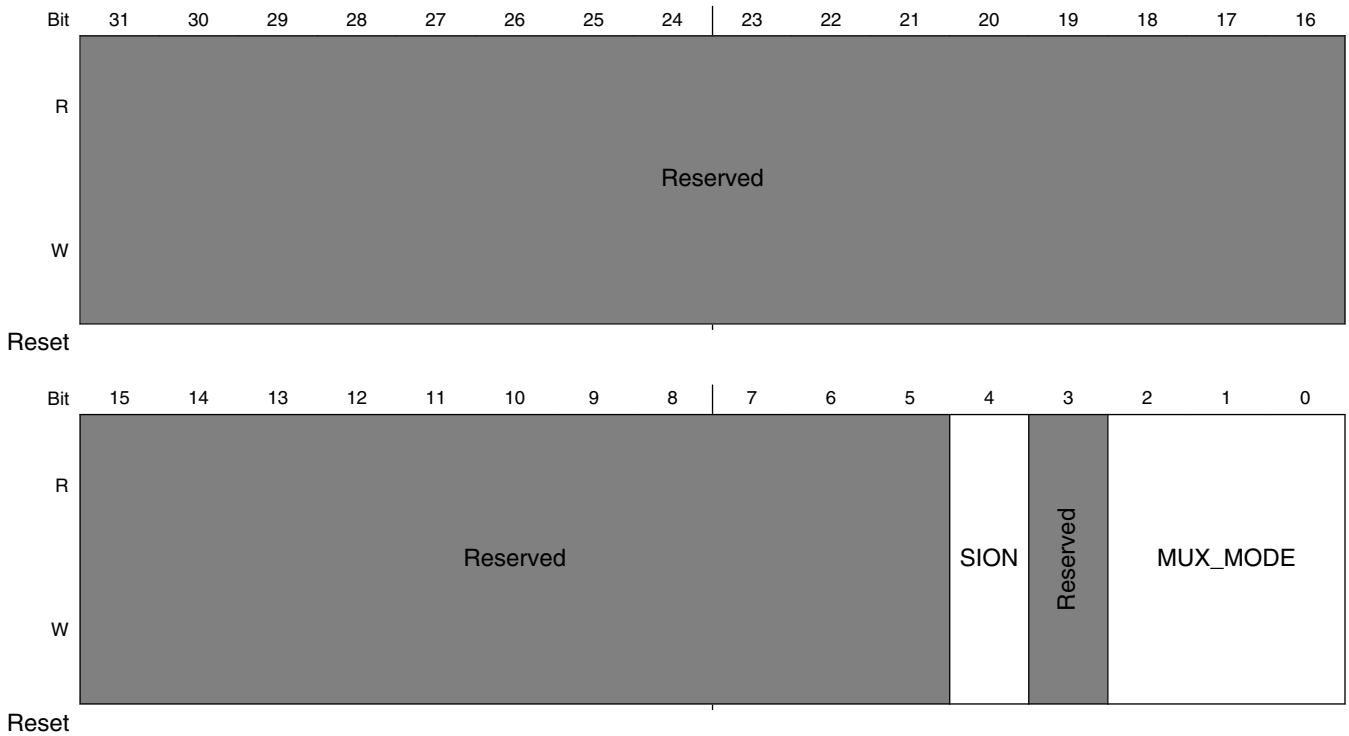
IOMUXC_SW_MUX_CTL_PAD_GPIO_08 field descriptions

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_08 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO_08. 000 **ALT0** — Select mux mode: ALT0 mux port: SAI1_MCLK of instance: SAI1 001 **ALT1** — Select mux mode: ALT1 mux port: GPT1_CLK of instance: GPT1 010 **ALT2** — Select mux mode: ALT2 mux port: FLEXPWM1_PWM3_A of instance: FLEXPWM1 011 **ALT3** — Select mux mode: ALT3 mux port: LPUART3_TXD of instance: LPUART3 100 **ALT4** — Select mux mode: ALT4 mux port: FLEXIO1_IO00 of instance: FLEXIO1 101 **ALT5** — Select mux mode: ALT5 mux port: GPIOMUX_IO08 of instance: GPIOMUX 110 **ALT6** — Select mux mode: ALT6 mux port: LPUART1_CTS_B of instance: LPUART1

11.7.37 SW_MUX_CTL_PAD_GPIO_07 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_07)

SW_MUX_CTL Register

Address: 401F_8000h base + A0h offset = 401F_80A0h



IOMUXC_SW_MUX_CTL_PAD_GPIO_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_07 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO_07. 000 **ALT0** — Select mux mode: ALT0 mux port: SAI1_TX_SYNC of instance: SAI1 001 **ALT1** — Select mux mode: ALT1 mux port: GPT1_COMPARE1 of instance: GPT1 010 **ALT2** — Select mux mode: ALT2 mux port: FLEXPWM1_PWM3_B of instance: FLEXPWM1

Table continues on the next page...

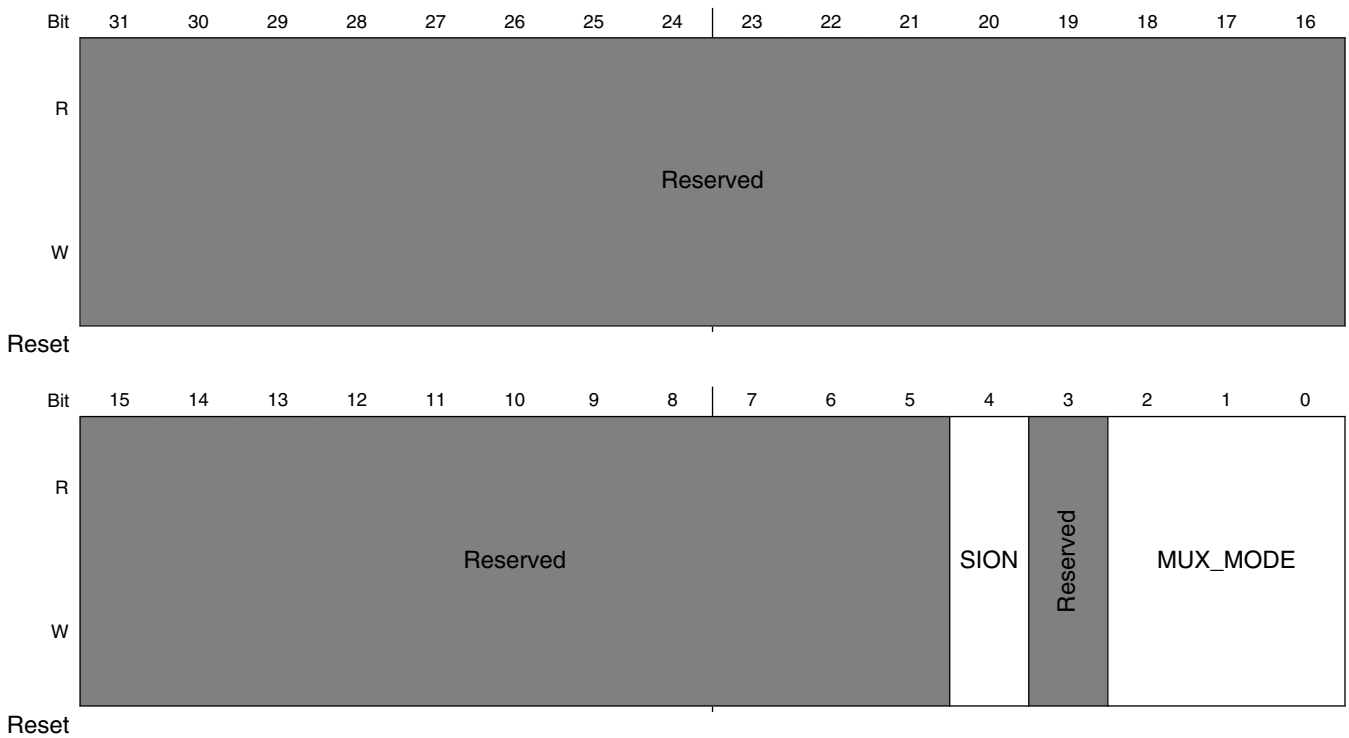
IOMUXC_SW_MUX_CTL_PAD_GPIO_07 field descriptions (continued)

Field	Description
	011 **ALT3** — Select mux mode: ALT3 mux port: LPUART3_RXD of instance: LPUART3 100 **ALT4** — Select mux mode: ALT4 mux port: SPDIF_LOCK of instance: SPDIF 101 **ALT5** — Select mux mode: ALT5 mux port: GPIOMUX_IO07 of instance: GPIOMUX 110 **ALT6** — Select mux mode: ALT6 mux port: LPUART1_RTS_B of instance: LPUART1

11.7.38 SW_MUX_CTL_PAD_GPIO_06 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_06)

SW_MUX_CTL Register

Address: 401F_8000h base + A4h offset = 401F_80A4h



IOMUXC_SW_MUX_CTL_PAD_GPIO_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_06 0 **DISABLED** — Input Path is determined by functionality

Table continues on the next page...

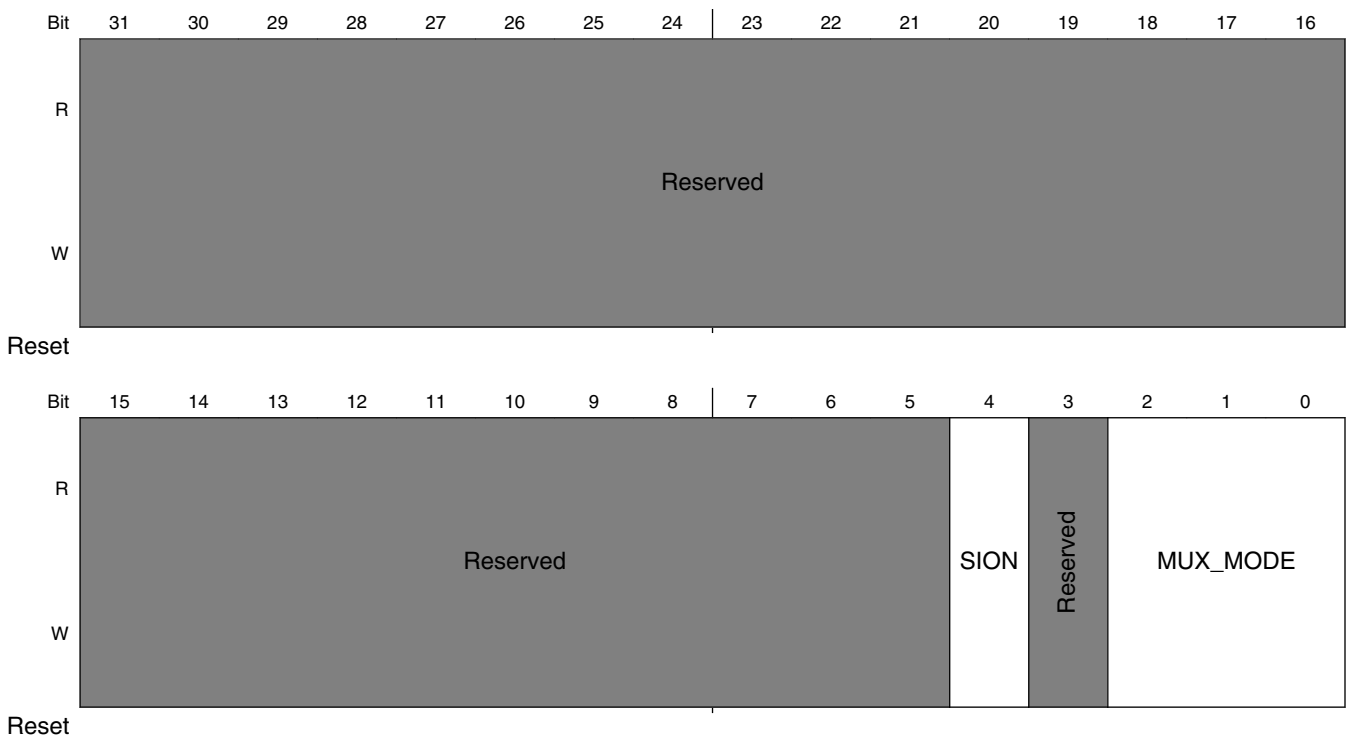
IOMUXC_SW_MUX_CTL_PAD_GPIO_06 field descriptions (continued)

Field	Description
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO_06. 000 **ALT0** — Select mux mode: ALT0 mux port: SAI1_TX_BCLK of instance: SAI1 001 **ALT1** — Select mux mode: ALT1 mux port: GPT1_CAPTURE1 of instance: GPT1 010 **ALT2** — Select mux mode: ALT2 mux port: FLEXPWM1_PWM2_A of instance: FLEXPWM1 011 **ALT3** — Select mux mode: ALT3 mux port: LPUART4_TXD of instance: LPUART4 100 **ALT4** — Select mux mode: ALT4 mux port: SPDIF_EXT_CLK of instance: SPDIF 101 **ALT5** — Select mux mode: ALT5 mux port: GPIOMUX_IO06 of instance: GPIOMUX

11.7.39 SW_MUX_CTL_PAD_GPIO_05 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_05)

SW_MUX_CTL Register

Address: 401F_8000h base + A8h offset = 401F_80A8h



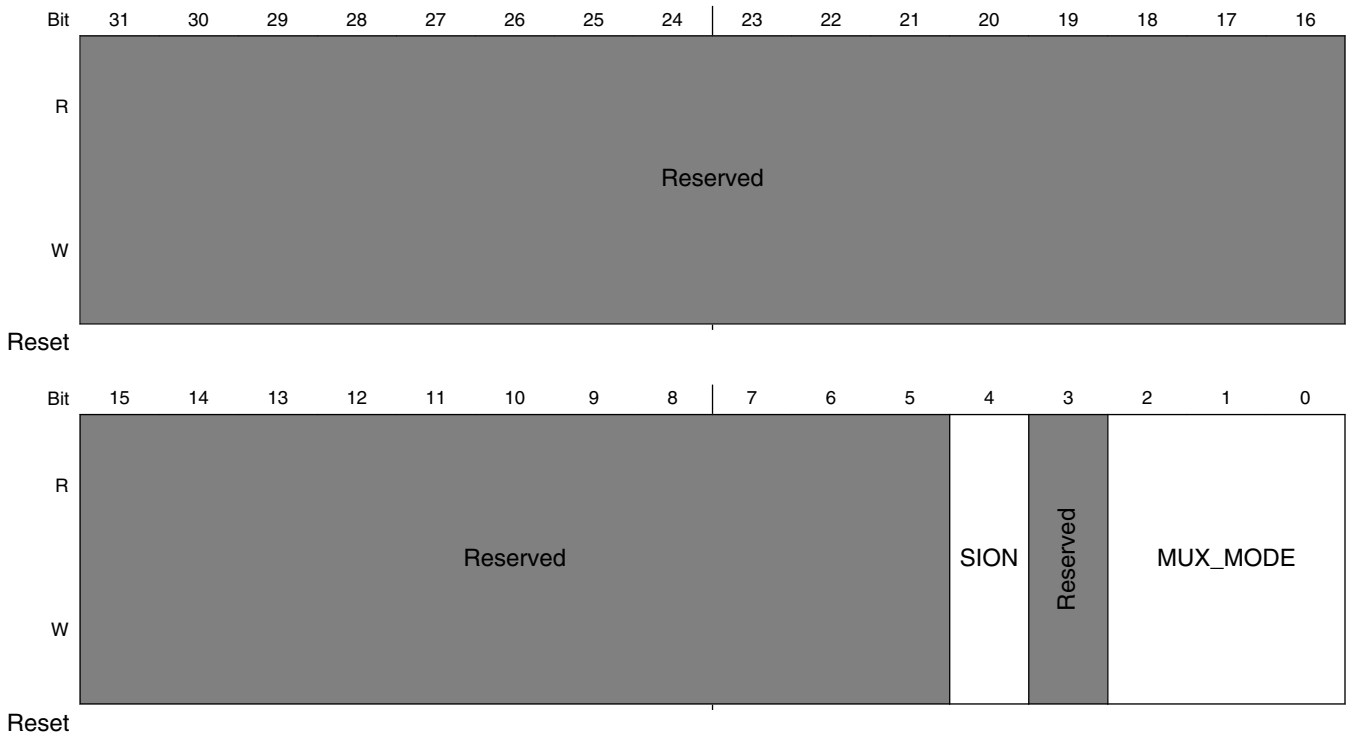
IOMUXC_SW_MUX_CTL_PAD_GPIO_05 field descriptions

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_05 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO_05. 000 **ALT0** — Select mux mode: ALT0 mux port: SAI1_TX_DATA01 of instance: SAI1 001 **ALT1** — Select mux mode: ALT1 mux port: GPT1_COMPARE2 of instance: GPT1 010 **ALT2** — Select mux mode: ALT2 mux port: FLEXPWM1_PWM2_B of instance: FLEXPWM1 011 **ALT3** — Select mux mode: ALT3 mux port: LPUART4_RXD of instance: LPUART4 100 **ALT4** — Select mux mode: ALT4 mux port: SPDIF_OUT of instance: SPDIF 101 **ALT5** — Select mux mode: ALT5 mux port: GPIOMUX_IO05 of instance: GPIOMUX

11.7.40 SW_MUX_CTL_PAD_GPIO_04 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_04)

SW_MUX_CTL Register

Address: 401F_8000h base + ACh offset = 401F_80ACh



IOMUXC_SW_MUX_CTL_PAD_GPIO_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_04 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO_04. 000 **ALT0** — Select mux mode: ALT0 mux port: SAI1_TX_DATA00 of instance: SAI1 001 **ALT1** — Select mux mode: ALT1 mux port: GPT1_CAPTURE2 of instance: GPT1 010 **ALT2** — Select mux mode: ALT2 mux port: FLEXPWM1_PWM1_A of instance: FLEXPWM1

Table continues on the next page...

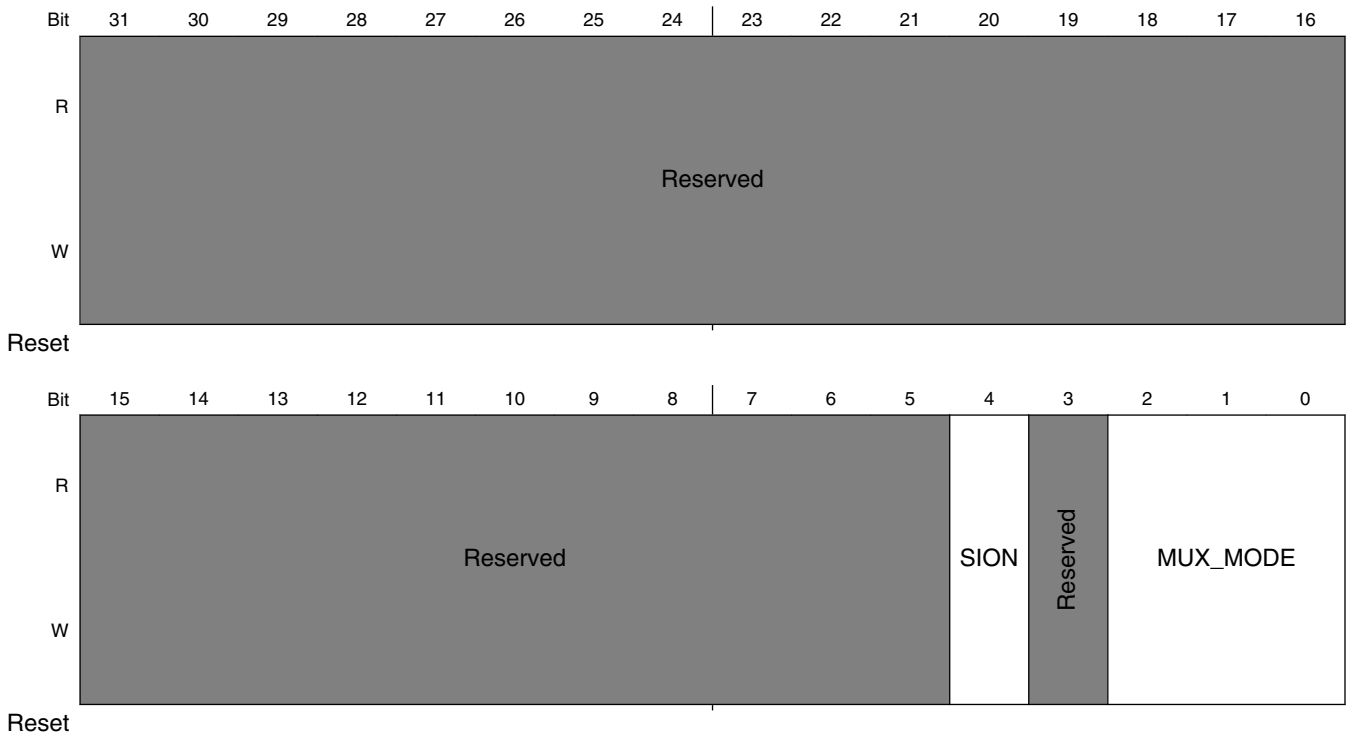
IOMUXC_SW_MUX_CTL_PAD_GPIO_04 field descriptions (continued)

Field	Description
	100 **ALT4** — Select mux mode: ALT4 mux port: SPDIF_IN of instance: SPDIF 101 **ALT5** — Select mux mode: ALT5 mux port: GPIOMUX_IO04 of instance: GPIOMUX

11.7.41 SW_MUX_CTL_PAD_GPIO_03 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_03)

SW_MUX_CTL Register

Address: 401F_8000h base + B0h offset = 401F_80B0h



IOMUXC_SW_MUX_CTL_PAD_GPIO_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_03 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved

Table continues on the next page...

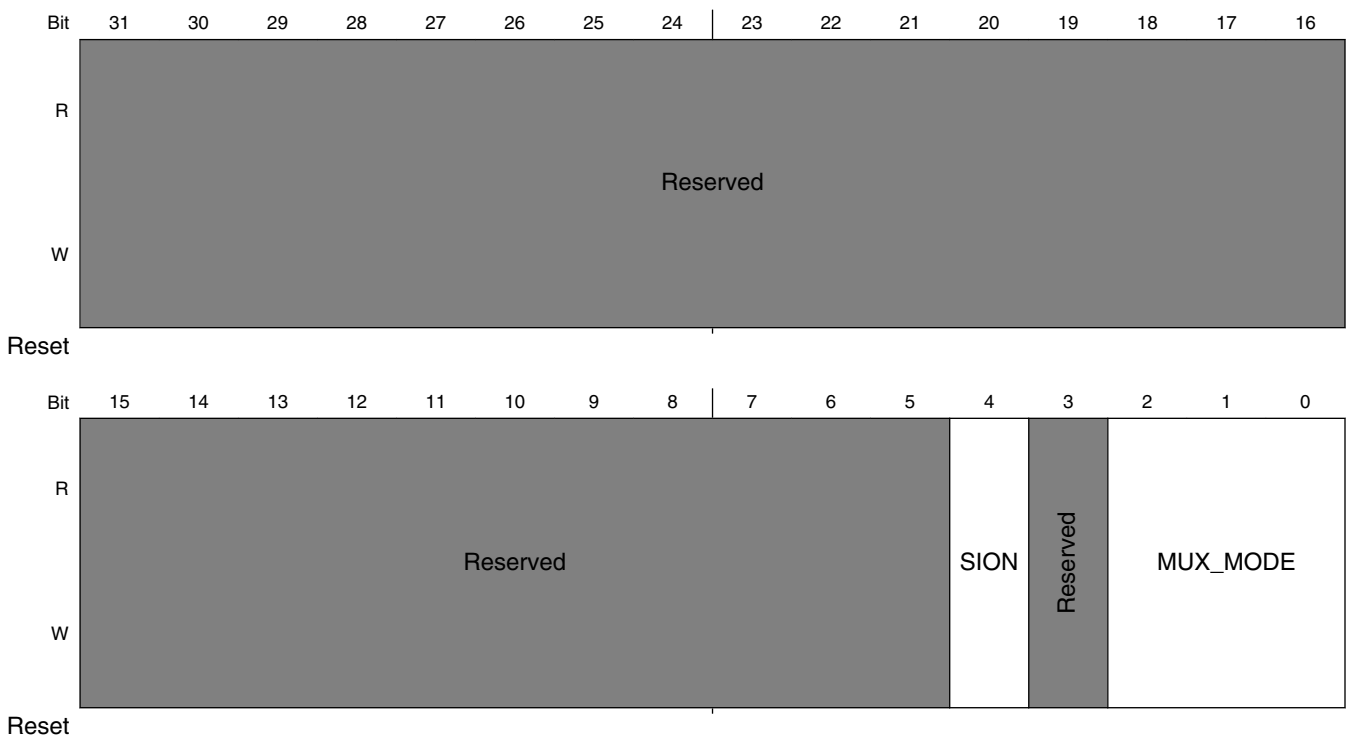
IOMUXC_SW_MUX_CTL_PAD_GPIO_03 field descriptions (continued)

Field	Description
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO_03. 000 **ALT0** — Select mux mode: ALT0 mux port: SAI1_RX_DATA00 of instance: SAI1 001 **ALT1** — Select mux mode: ALT1 mux port: GPT1_COMPARE3 of instance: GPT1 010 **ALT2** — Select mux mode: ALT2 mux port: FLEXPWM1_PWM1_B of instance: FLEXPWM1 100 **ALT4** — Select mux mode: ALT4 mux port: SPDIF_SR_CLK of instance: SPDIF 101 **ALT5** — Select mux mode: ALT5 mux port: GPIOMUX_IO03 of instance: GPIOMUX

11.7.42 SW_MUX_CTL_PAD_GPIO_02 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_02)

SW_MUX_CTL Register

Address: 401F_8000h base + B4h offset = 401F_80B4h



IOMUXC_SW_MUX_CTL_PAD_GPIO_02 field descriptions

Field	Description
31–5 -	This field is reserved. Reserved

Table continues on the next page...

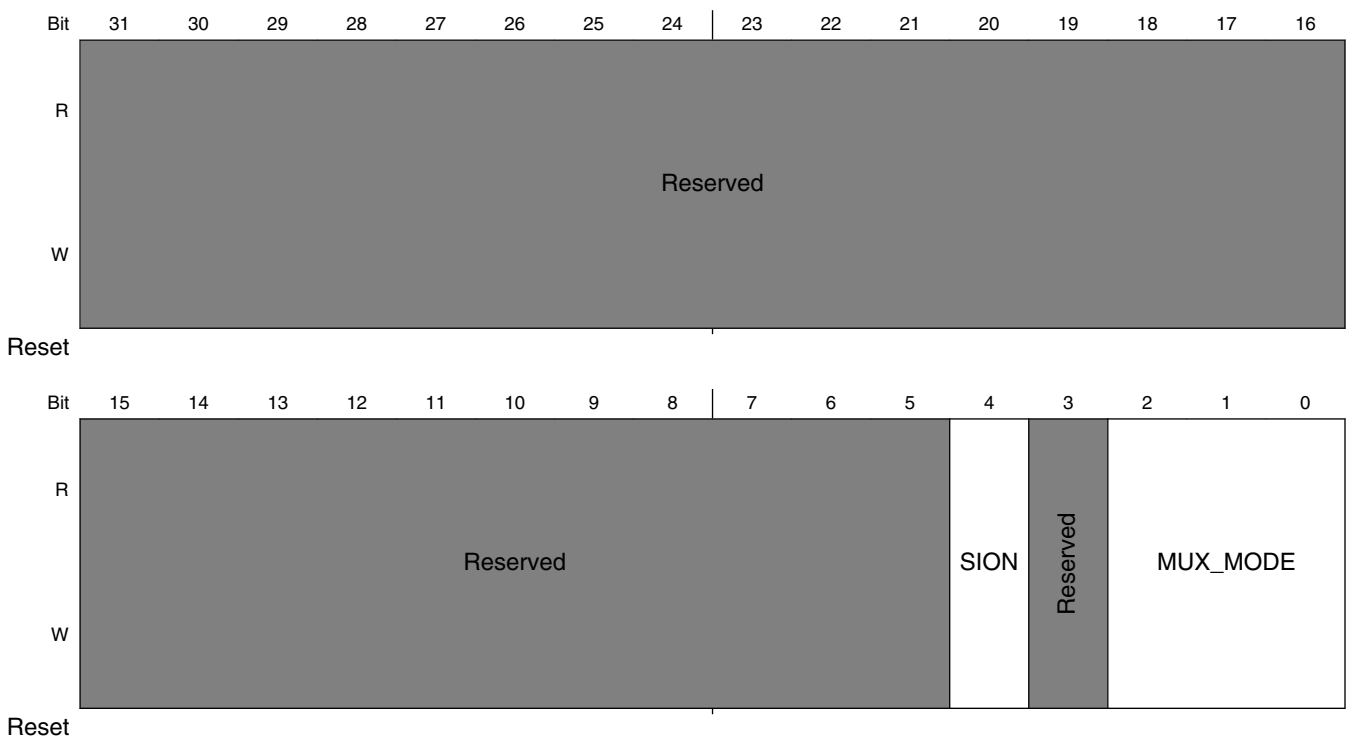
IOMUXC_SW_MUX_CTL_PAD_GPIO_02 field descriptions (continued)

Field	Description
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_02 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO_02. 000 **ALT0** — Select mux mode: ALT0 mux port: SAI1_RX_SYNC of instance: SAI1 001 **ALT1** — Select mux mode: ALT1 mux port: WDOG2_B of instance: WDOG2 010 **ALT2** — Select mux mode: ALT2 mux port: FLEXPWM1_PWM0_A of instance: FLEXPWM1 011 **ALT3** — Select mux mode: ALT3 mux port: LPI2C1_SCL of instance: LPI2C1 100 **ALT4** — Select mux mode: ALT4 mux port: KPP_COL03 of instance: KPP 101 **ALT5** — Select mux mode: ALT5 mux port: GPIOMUX_IO02 of instance: GPIOMUX

11.7.43 SW_MUX_CTL_PAD_GPIO_01 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_01)

SW_MUX_CTL Register

Address: 401F_8000h base + B8h offset = 401F_80B8h



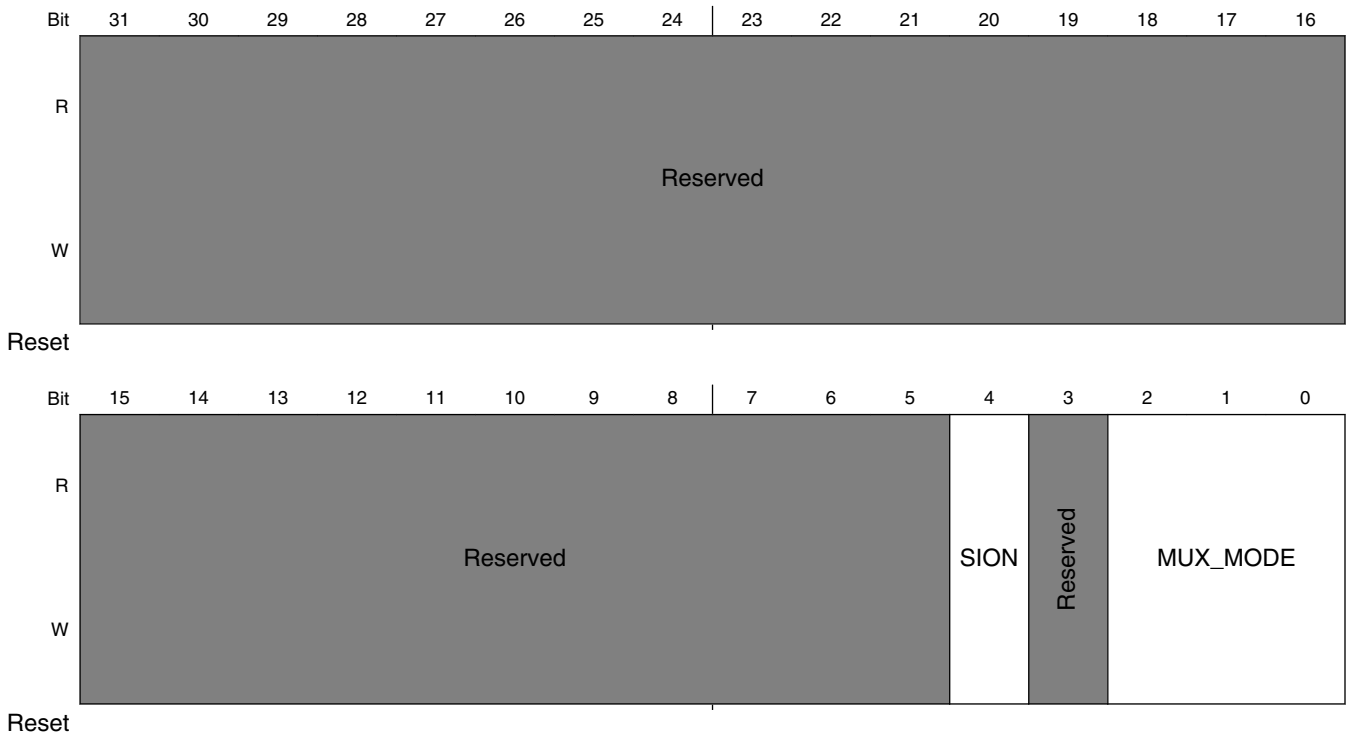
IOMUXC_SW_MUX_CTL_PAD_GPIO_01 field descriptions

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_01 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO_01. 000 **ALT0** — Select mux mode: ALT0 mux port: SAI1_RX_BCLK of instance: SAI1 001 **ALT1** — Select mux mode: ALT1 mux port: WDOG1_ANY of instance: WDOG1 010 **ALT2** — Select mux mode: ALT2 mux port: FLEXPWM1_PWM0_B of instance: FLEXPWM1 011 **ALT3** — Select mux mode: ALT3 mux port: LPI2C1_SDA of instance: LPI2C1 100 **ALT4** — Select mux mode: ALT4 mux port: KPP_ROW03 of instance: KPP 101 **ALT5** — Select mux mode: ALT5 mux port: GPIOMUX_IO01 of instance: GPIOMUX

11.7.44 SW_MUX_CTL_PAD_GPIO_00 SW MUX Control Register (IOMUXC_SW_MUX_CTL_PAD_GPIO_00)

SW_MUX_CTL Register

Address: 401F_8000h base + BCh offset = 401F_80BCh



IOMUXC_SW_MUX_CTL_PAD_GPIO_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_00 0 **DISABLED** — Input Path is determined by functionality
3 -	This field is reserved. Reserved
MUX_MODE	MUX Mode Select Field. Select 1 of 8 iomux modes to be used for pad: GPIO_00. 000 **ALT0** — Select mux mode: ALT0 mux port: FLEXSPI_B_DQS of instance: FLEXSPI 001 **ALT1** — Select mux mode: ALT1 mux port: SAI3_MCLK of instance: SAI3 010 **ALT2** — Select mux mode: ALT2 mux port: LPSPI2_PCS3 of instance: LPSPI2

Table continues on the next page...

IOMUXC_SW_MUX_CTL_PAD_GPIO_00 field descriptions (continued)

Field	Description
	011 **ALT3** — Select mux mode: ALT3 mux port: LPSPI1_PCS3 of instance: LPSPI1 100 **ALT4** — Select mux mode: ALT4 mux port: PIT_TRIGGER00 of instance: PIT 101 **ALT5** — Select mux mode: ALT5 mux port: GPIOMUX_IO00 of instance: GPIOMUX

11.7.45 SW_PAD_CTL_PAD_GPIO_AD_14 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_14)

SW_PAD_CTL Register

Address: 401F_8000h base + C0h offset = 401F_80C0h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved															HYS
Reset																

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																

IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_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_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_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_14 0 **PUE_0_Keeper** — Keeper 1 **PUE_1_Pull** — Pull
12 PKE	Pull / Keep Enable Field

Table continues on the next page...

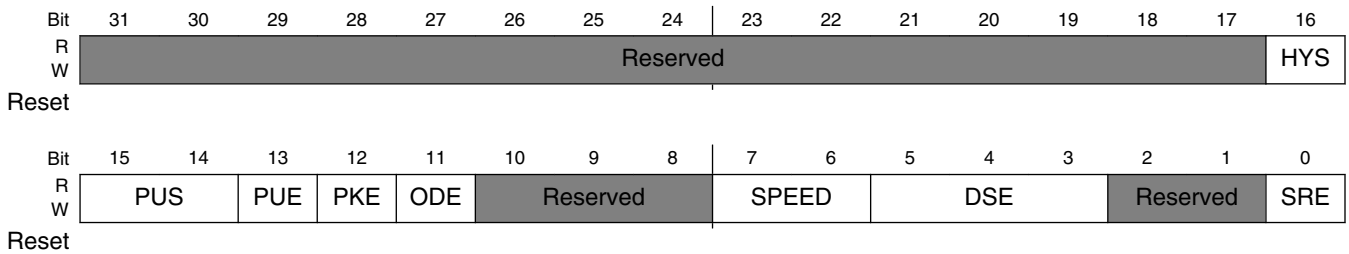
IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_14 field descriptions (continued)

Field	Description
	Select one out of next values for pad: GPIO_AD_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_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_AD_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_14 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_150_Ohm___3_3V__260_Ohm_1_8V__240_Ohm_for_DDR_** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V, 240 Ohm for DDR) 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_14 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.7.46 SW_PAD_CTL_PAD_GPIO_AD_13 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_13)

SW_PAD_CTL Register

Address: 401F_8000h base + C4h offset = 401F_80C4h



IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_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_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_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_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_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_13

Table continues on the next page...

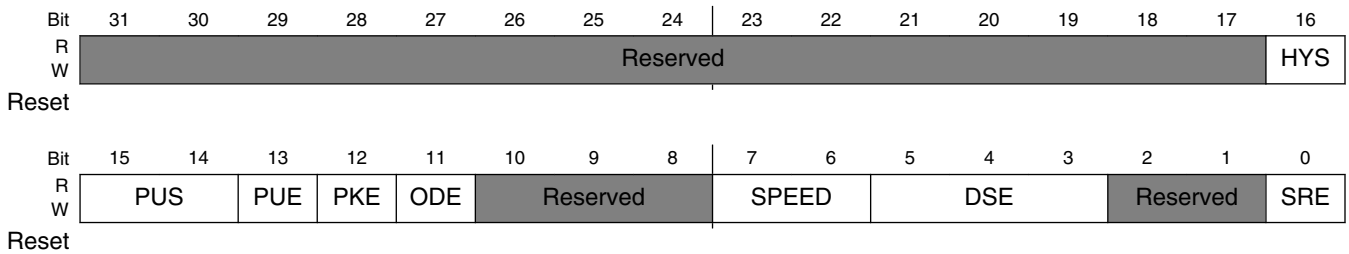
IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_13 field descriptions (continued)

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_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_13 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_150_Ohm___3_3V__260_Ohm_1_8V__240_Ohm_for_DDR_** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V, 240 Ohm for DDR) 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_13 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.7.47 SW_PAD_CTL_PAD_GPIO_AD_12 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_12)

SW_PAD_CTL Register

Address: 401F_8000h base + C8h offset = 401F_80C8h



IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_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_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_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_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_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_12

Table continues on the next page...

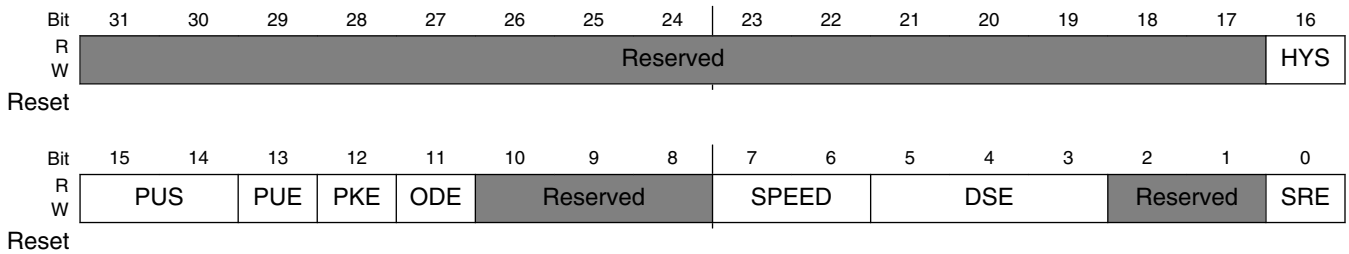
IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_12 field descriptions (continued)

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_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_12 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_150_Ohm___3_3V__260_Ohm_1_8V__240_Ohm_for_DDR_** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V, 240 Ohm for DDR) 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_12 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.7.48 SW_PAD_CTL_PAD_GPIO_AD_11 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_11)

SW_PAD_CTL Register

Address: 401F_8000h base + CCh offset = 401F_80CCh



IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_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_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_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_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_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_11

Table continues on the next page...

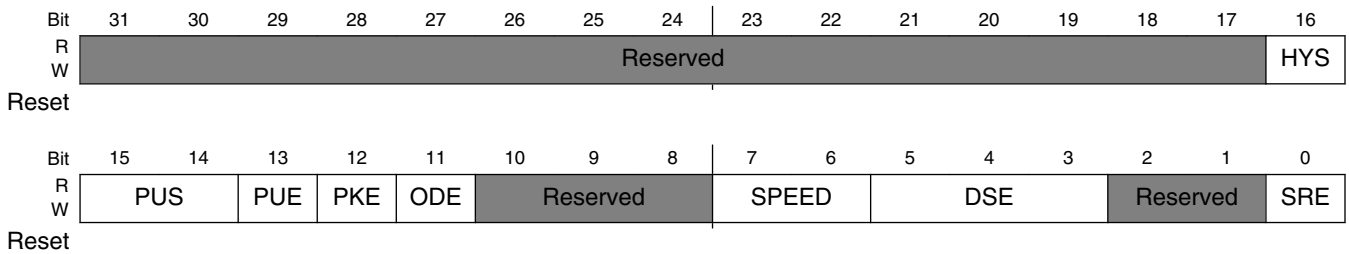
IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_11 field descriptions (continued)

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_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_11 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_150_Ohm___3_3V__260_Ohm_1_8V__240_Ohm_for_DDR_** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V, 240 Ohm for DDR) 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_11 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.7.49 SW_PAD_CTL_PAD_GPIO_AD_10 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_10)

SW_PAD_CTL Register

Address: 401F_8000h base + D0h offset = 401F_80D0h



IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_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_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_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_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_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_10

Table continues on the next page...

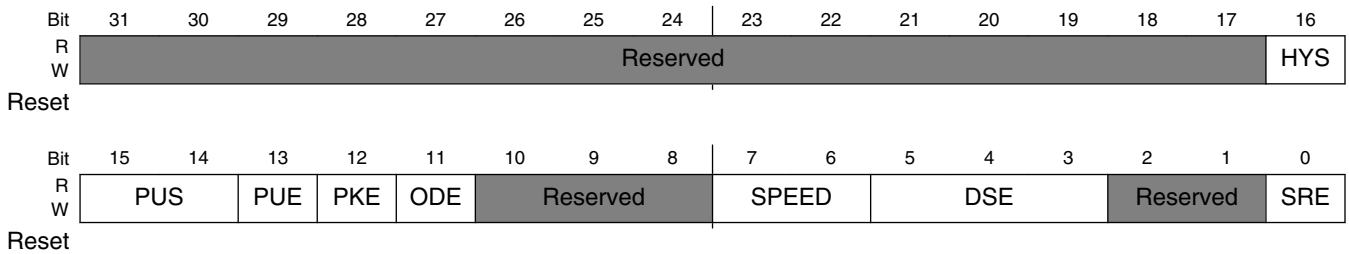
IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_10 field descriptions (continued)

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_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_10 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_150_Ohm___3_3V__260_Ohm_1_8V__240_Ohm_for_DDR_** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V, 240 Ohm for DDR) 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_10 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.7.50 SW_PAD_CTL_PAD_GPIO_AD_09 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_09)

SW_PAD_CTL Register

Address: 401F_8000h base + D4h offset = 401F_80D4h



IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_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_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_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_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_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_09

Table continues on the next page...

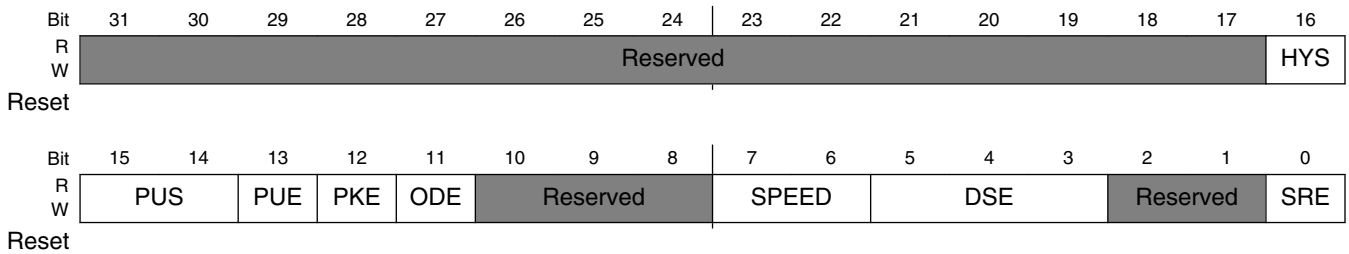
IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_09 field descriptions (continued)

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_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_09 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_150_Ohm___3_3V__260_Ohm_1_8V__240_Ohm_for_DDR_** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V, 240 Ohm for DDR) 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_09 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.7.51 SW_PAD_CTL_PAD_GPIO_AD_08 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_08)

SW_PAD_CTL Register

Address: 401F_8000h base + D8h offset = 401F_80D8h



IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_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_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_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_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_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_08

Table continues on the next page...

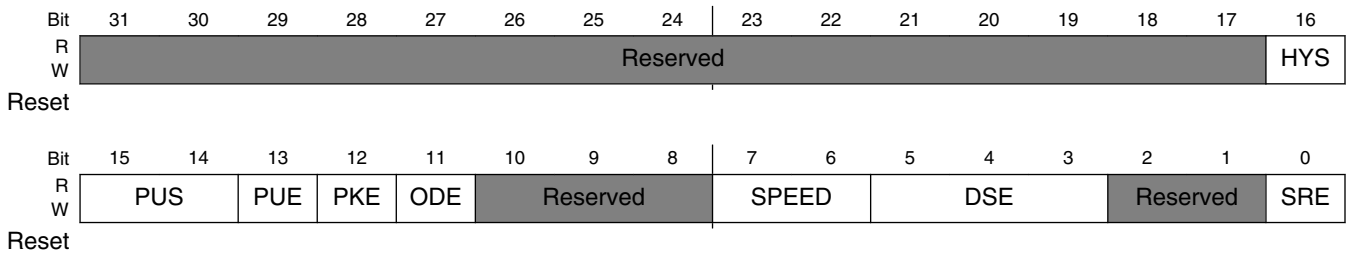
IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_08 field descriptions (continued)

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_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_08 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_150_Ohm___3_3V__260_Ohm_1_8V__240_Ohm_for_DDR_** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V, 240 Ohm for DDR) 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_08 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.7.52 SW_PAD_CTL_PAD_GPIO_AD_07 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_07)

SW_PAD_CTL Register

Address: 401F_8000h base + DCh offset = 401F_80DCh



IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_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_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_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_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_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_07

Table continues on the next page...

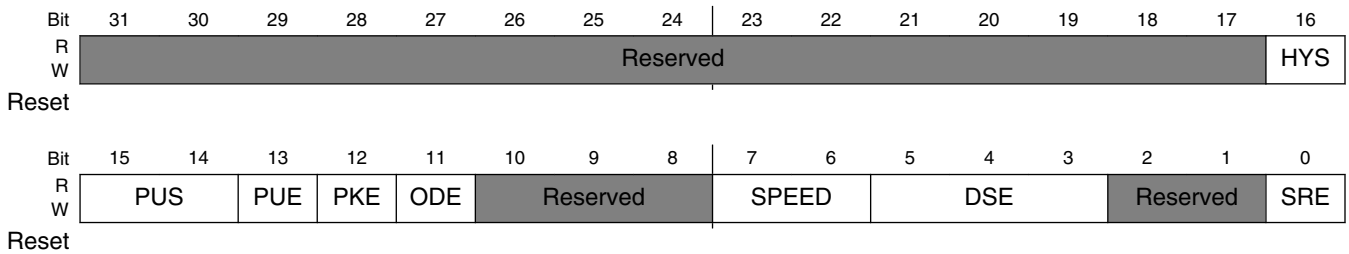
IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_07 field descriptions (continued)

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_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_07 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_150_Ohm___3_3V__260_Ohm_1_8V__240_Ohm_for_DDR_** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V, 240 Ohm for DDR) 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_07 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.7.53 SW_PAD_CTL_PAD_GPIO_AD_06 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_06)

SW_PAD_CTL Register

Address: 401F_8000h base + E0h offset = 401F_80E0h



IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_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_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_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_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_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_06

Table continues on the next page...

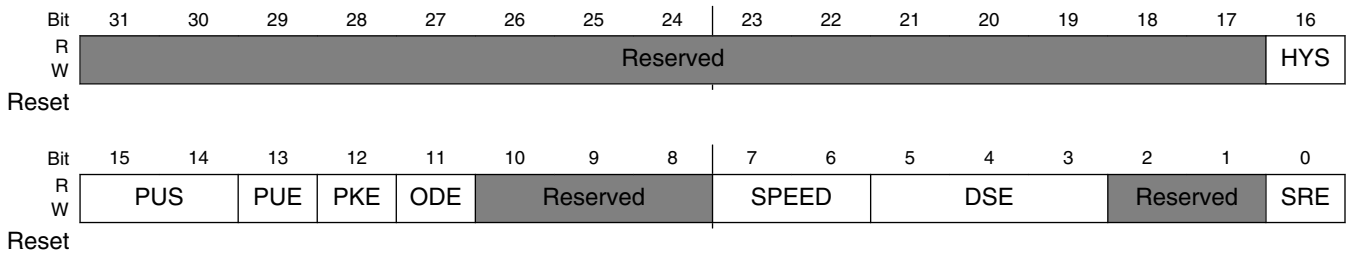
IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_06 field descriptions (continued)

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_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_06 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_150_Ohm___3_3V__260_Ohm_1_8V__240_Ohm_for_DDR_** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V, 240 Ohm for DDR) 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_06 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.7.54 SW_PAD_CTL_PAD_GPIO_AD_05 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_05)

SW_PAD_CTL Register

Address: 401F_8000h base + E4h offset = 401F_80E4h



IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_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_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_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_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_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_05

Table continues on the next page...

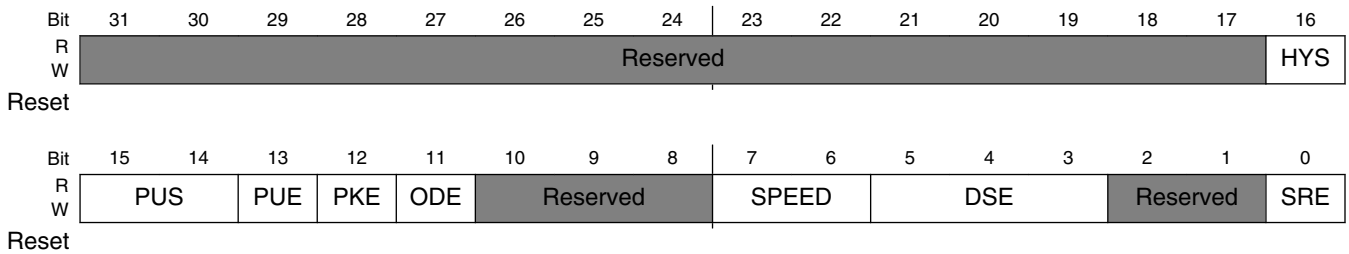
IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_05 field descriptions (continued)

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_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_05 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_150_Ohm___3_3V__260_Ohm_1_8V__240_Ohm_for_DDR_** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V, 240 Ohm for DDR) 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_05 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.7.55 SW_PAD_CTL_PAD_GPIO_AD_04 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_04)

SW_PAD_CTL Register

Address: 401F_8000h base + E8h offset = 401F_80E8h



IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_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_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_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_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_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_04

Table continues on the next page...

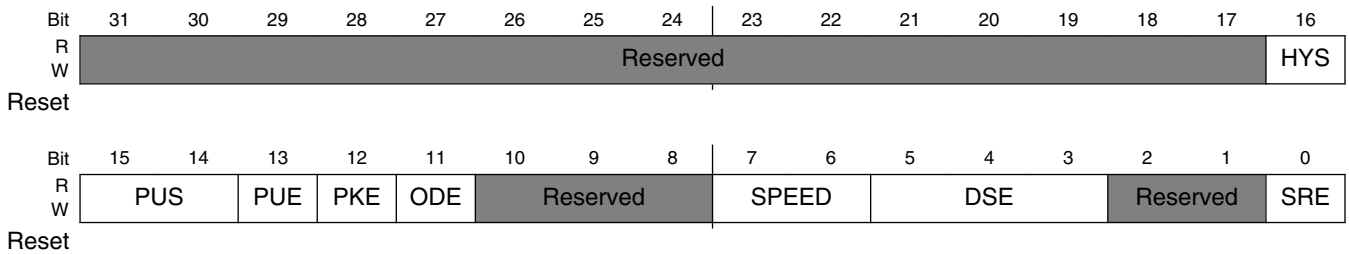
IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_04 field descriptions (continued)

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_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_04 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_150_Ohm___3_3V__260_Ohm_1_8V__240_Ohm_for_DDR_** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V, 240 Ohm for DDR) 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_04 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.7.56 SW_PAD_CTL_PAD_GPIO_AD_03 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_03)

SW_PAD_CTL Register

Address: 401F_8000h base + ECh offset = 401F_80ECh



IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_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_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_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_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_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_03

Table continues on the next page...

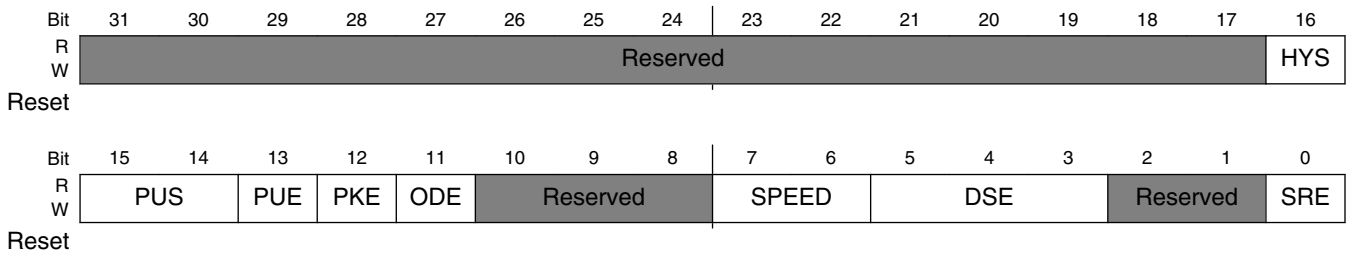
IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_03 field descriptions (continued)

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_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_03 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_150_Ohm___3_3V__260_Ohm_1_8V__240_Ohm_for_DDR_** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V, 240 Ohm for DDR) 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_03 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.7.57 SW_PAD_CTL_PAD_GPIO_AD_02 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_02)

SW_PAD_CTL Register

Address: 401F_8000h base + F0h offset = 401F_80F0h



IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_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_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_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_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_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_02

Table continues on the next page...

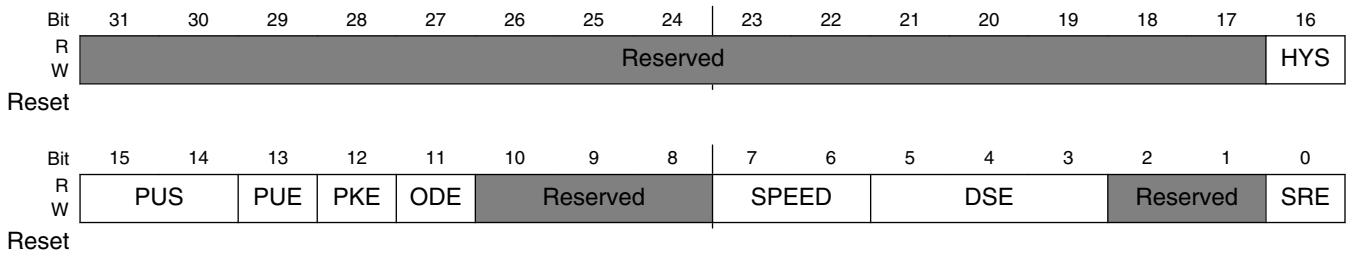
IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_02 field descriptions (continued)

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_AD_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_02 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_150_Ohm___3_3V__260_Ohm_1_8V__240_Ohm_for_DDR_** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V, 240 Ohm for DDR) 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_02 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.7.58 SW_PAD_CTL_PAD_GPIO_AD_01 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_01)

SW_PAD_CTL Register

Address: 401F_8000h base + F4h offset = 401F_80F4h



IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_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_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_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_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_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_01

Table continues on the next page...

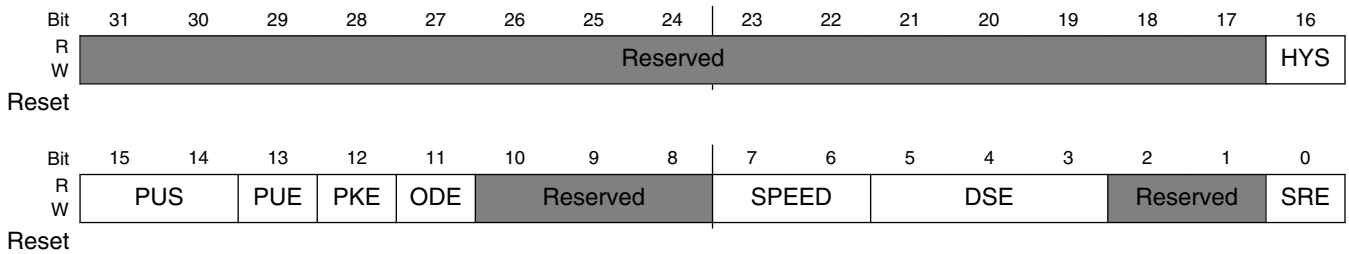
IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_01 field descriptions (continued)

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_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_01 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_150_Ohm___3_3V__260_Ohm_1_8V__240_Ohm_for_DDR_** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V, 240 Ohm for DDR) 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_01 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.7.59 SW_PAD_CTL_PAD_GPIO_AD_00 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_00)

SW_PAD_CTL Register

Address: 401F_8000h base + F8h offset = 401F_80F8h



IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_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_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_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_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_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_00

Table continues on the next page...

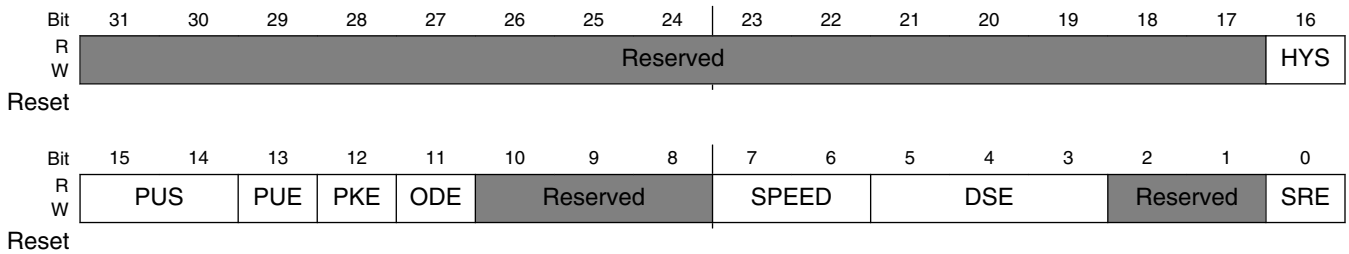
IOMUXC_SW_PAD_CTL_PAD_GPIO_AD_00 field descriptions (continued)

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_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_00 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_150_Ohm___3_3V__260_Ohm_1_8V__240_Ohm_for_DDR_** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V, 240 Ohm for DDR) 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_00 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.7.60 SW_PAD_CTL_PAD_GPIO_SD_14 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_14)

SW_PAD_CTL Register

Address: 401F_8000h base + FCh offset = 401F_80FCh



IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_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_SD_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_SD_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_SD_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_SD_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_SD_14

Table continues on the next page...

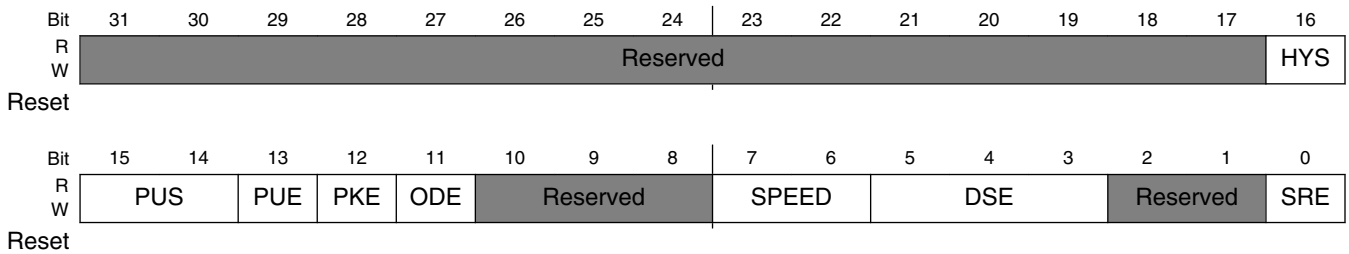
IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_14 field descriptions (continued)

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_SD_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_SD_14 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_150_Ohm___3_3V__260_Ohm_1_8V__240_Ohm_for_DDR_** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V, 240 Ohm for DDR) 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_14 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.7.61 SW_PAD_CTL_PAD_GPIO_SD_13 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_13)

SW_PAD_CTL Register

Address: 401F_8000h base + 100h offset = 401F_8100h



IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_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_SD_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_SD_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_SD_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_SD_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_SD_13

Table continues on the next page...

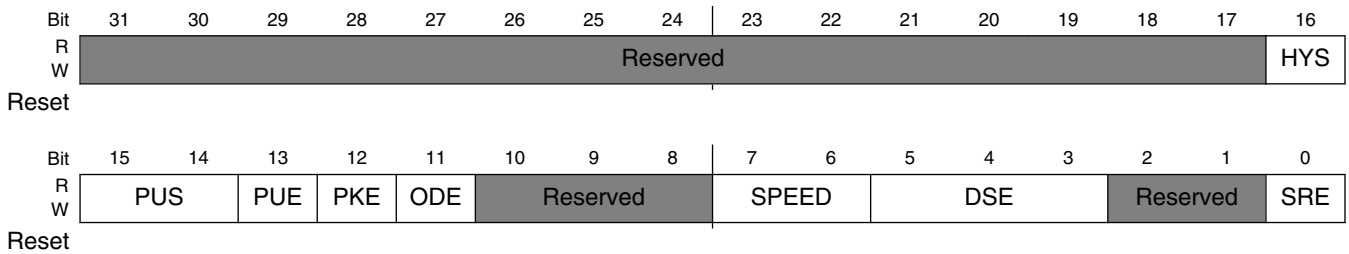
IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_13 field descriptions (continued)

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_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_SD_13 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_150_Ohm___3_3V__260_Ohm_1_8V__240_Ohm_for_DDR_** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V, 240 Ohm for DDR) 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_13 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.7.62 SW_PAD_CTL_PAD_GPIO_SD_12 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_12)

SW_PAD_CTL Register

Address: 401F_8000h base + 104h offset = 401F_8104h



IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_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_SD_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_SD_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_SD_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_SD_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_SD_12

Table continues on the next page...

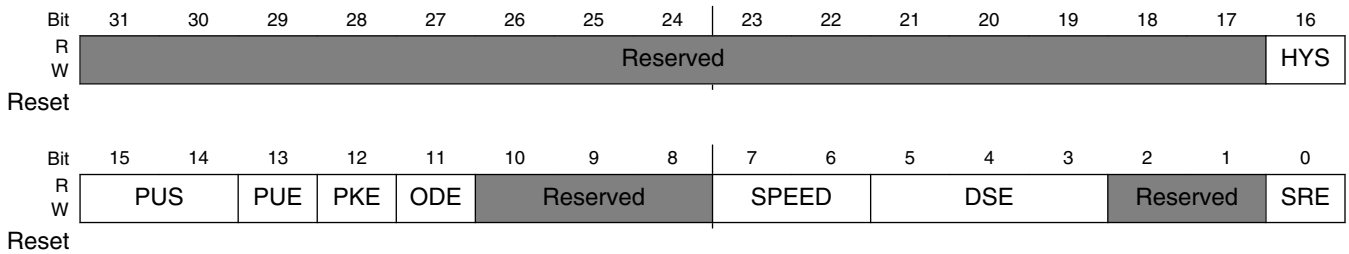
IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_12 field descriptions (continued)

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_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_SD_12 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_150_Ohm___3_3V__260_Ohm_1_8V__240_Ohm_for_DDR_** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V, 240 Ohm for DDR) 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_12 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.7.63 SW_PAD_CTL_PAD_GPIO_SD_11 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_11)

SW_PAD_CTL Register

Address: 401F_8000h base + 108h offset = 401F_8108h



IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_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_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_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_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_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_11

Table continues on the next page...

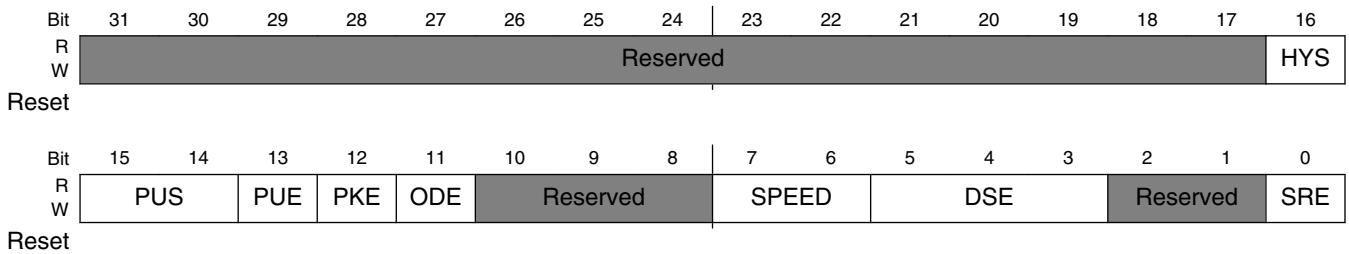
IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_11 field descriptions (continued)

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_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_11 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_150_Ohm___3_3V__260_Ohm_1_8V__240_Ohm_for_DDR_** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V, 240 Ohm for DDR) 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_11 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.7.64 SW_PAD_CTL_PAD_GPIO_SD_10 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_10)

SW_PAD_CTL Register

Address: 401F_8000h base + 10Ch offset = 401F_810Ch



IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_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_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_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_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_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_10

Table continues on the next page...

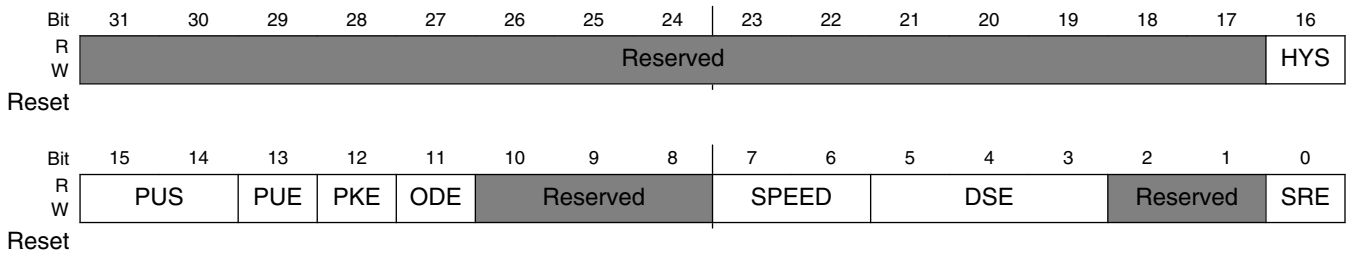
IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_10 field descriptions (continued)

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_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_10 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_150_Ohm___3_3V__260_Ohm_1_8V__240_Ohm_for_DDR_** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V, 240 Ohm for DDR) 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_10 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.7.65 SW_PAD_CTL_PAD_GPIO_SD_09 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_09)

SW_PAD_CTL Register

Address: 401F_8000h base + 110h offset = 401F_8110h



IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_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_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_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_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_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_09

Table continues on the next page...

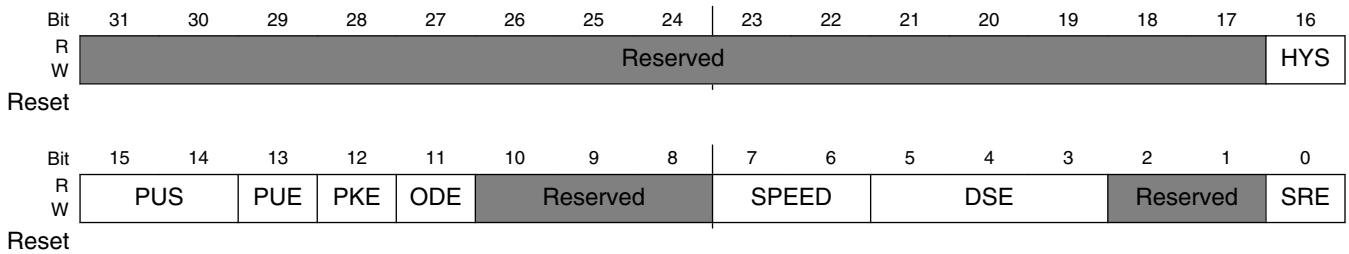
IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_09 field descriptions (continued)

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_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_09 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_150_Ohm___3_3V__260_Ohm_1_8V__240_Ohm_for_DDR_** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V, 240 Ohm for DDR) 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_09 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.7.66 SW_PAD_CTL_PAD_GPIO_SD_08 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_08)

SW_PAD_CTL Register

Address: 401F_8000h base + 114h offset = 401F_8114h



IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_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_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_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_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_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_08

Table continues on the next page...

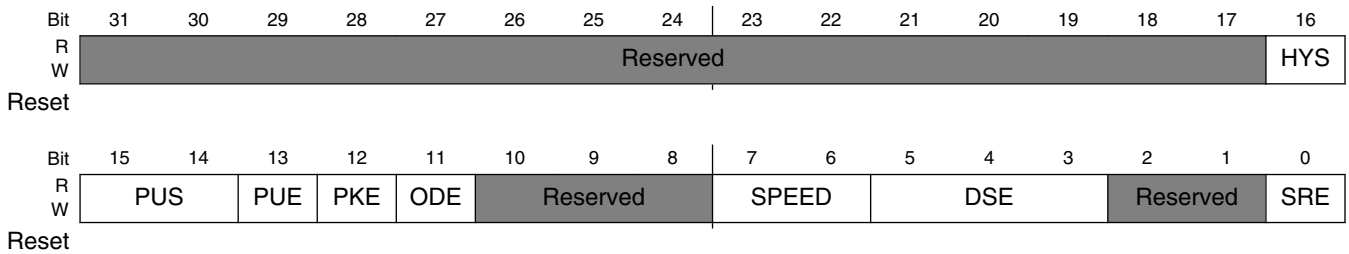
IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_08 field descriptions (continued)

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_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_08 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_150_Ohm___3_3V__260_Ohm_1_8V__240_Ohm_for_DDR_** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V, 240 Ohm for DDR) 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_08 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.7.67 SW_PAD_CTL_PAD_GPIO_SD_07 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_07)

SW_PAD_CTL Register

Address: 401F_8000h base + 118h offset = 401F_8118h



IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_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_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_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_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_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_07

Table continues on the next page...

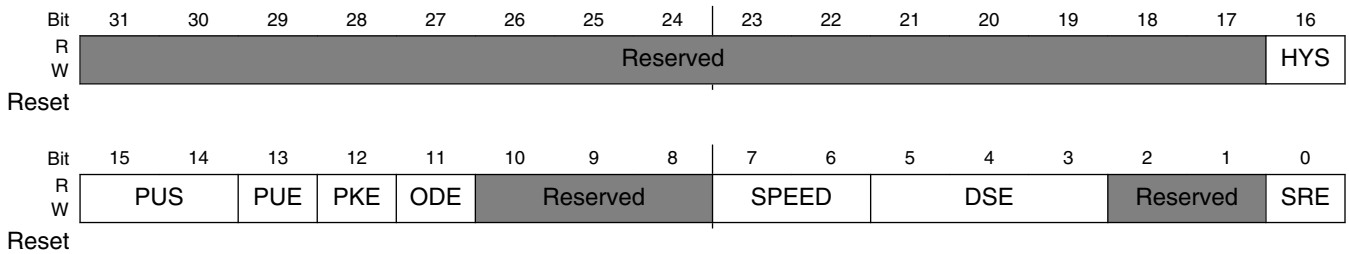
IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_07 field descriptions (continued)

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_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_07 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_150_Ohm___3_3V__260_Ohm_1_8V__240_Ohm_for_DDR_** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V, 240 Ohm for DDR) 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_07 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.7.68 SW_PAD_CTL_PAD_GPIO_SD_06 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_06)

SW_PAD_CTL Register

Address: 401F_8000h base + 11Ch offset = 401F_811Ch



IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_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_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_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_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_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_06

Table continues on the next page...

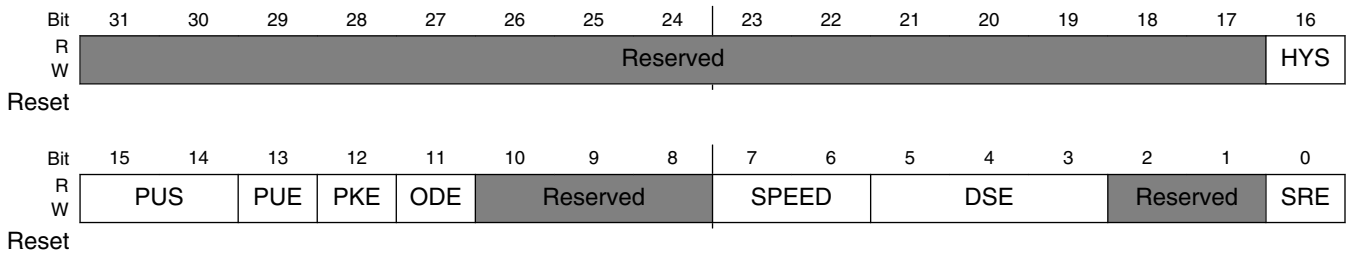
IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_06 field descriptions (continued)

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_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_06 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_150_Ohm___3_3V__260_Ohm_1_8V__240_Ohm_for_DDR_** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V, 240 Ohm for DDR) 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_06 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.7.69 SW_PAD_CTL_PAD_GPIO_SD_05 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_05)

SW_PAD_CTL Register

Address: 401F_8000h base + 120h offset = 401F_8120h



IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_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_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_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_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_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_05

Table continues on the next page...

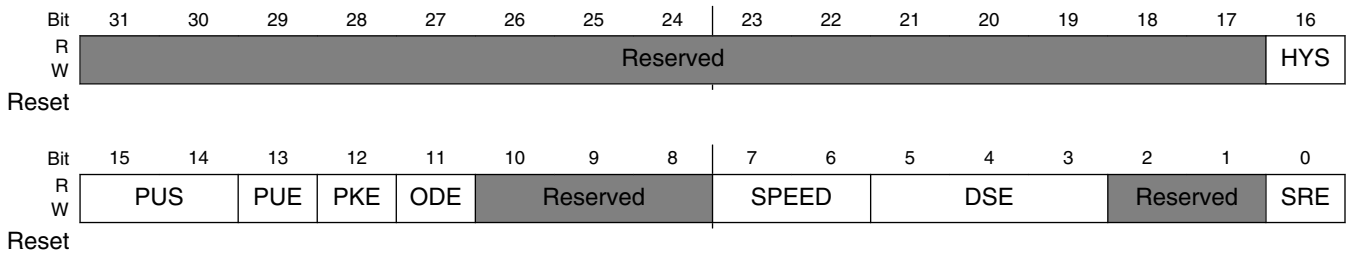
IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_05 field descriptions (continued)

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_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_05 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_150_Ohm___3_3V__260_Ohm_1_8V__240_Ohm_for_DDR_** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V, 240 Ohm for DDR) 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_05 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.7.70 SW_PAD_CTL_PAD_GPIO_SD_04 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_04)

SW_PAD_CTL Register

Address: 401F_8000h base + 124h offset = 401F_8124h



IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_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_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_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_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_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_04

Table continues on the next page...

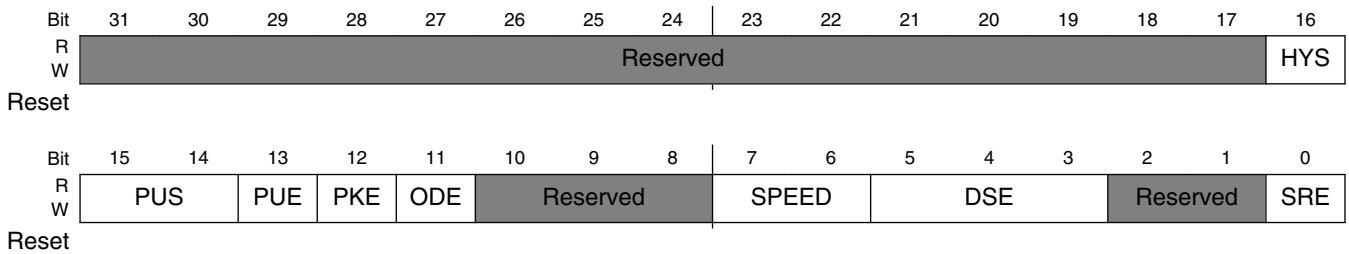
IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_04 field descriptions (continued)

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_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_04 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_150_Ohm___3_3V__260_Ohm_1_8V__240_Ohm_for_DDR_** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V, 240 Ohm for DDR) 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_04 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.7.71 SW_PAD_CTL_PAD_GPIO_SD_03 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_03)

SW_PAD_CTL Register

Address: 401F_8000h base + 128h offset = 401F_8128h



IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_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_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_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_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_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_03

Table continues on the next page...

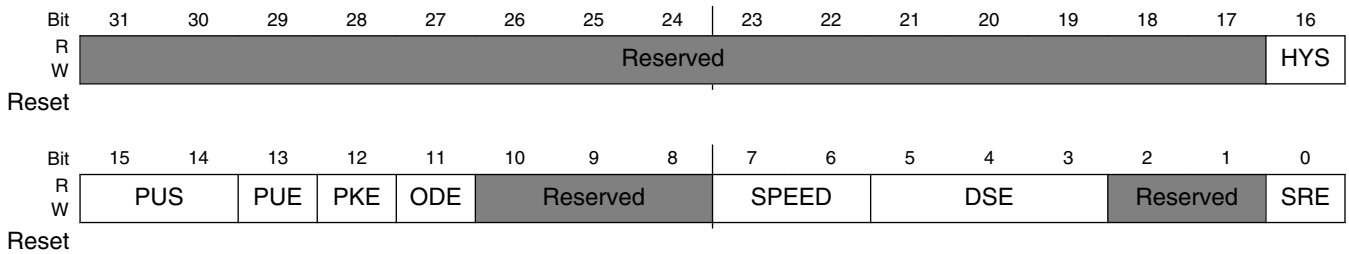
IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_03 field descriptions (continued)

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_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_03 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_150_Ohm___3_3V__260_Ohm_1_8V__240_Ohm_for_DDR_** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V, 240 Ohm for DDR) 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_03 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.7.72 SW_PAD_CTL_PAD_GPIO_SD_02 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_02)

SW_PAD_CTL Register

Address: 401F_8000h base + 12Ch offset = 401F_812Ch



IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_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_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_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_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_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_02

Table continues on the next page...

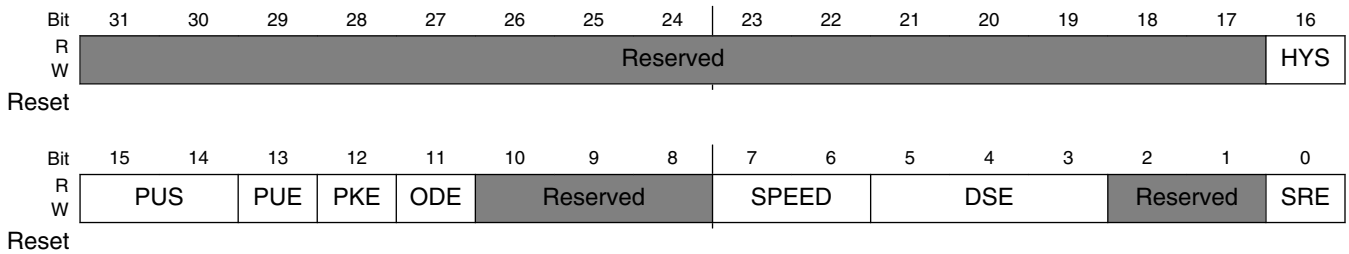
IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_02 field descriptions (continued)

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_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_02 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_150_Ohm___3_3V__260_Ohm_1_8V__240_Ohm_for_DDR_** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V, 240 Ohm for DDR) 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_02 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.7.73 SW_PAD_CTL_PAD_GPIO_SD_01 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_01)

SW_PAD_CTL Register

Address: 401F_8000h base + 130h offset = 401F_8130h



IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_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_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_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_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_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_01

Table continues on the next page...

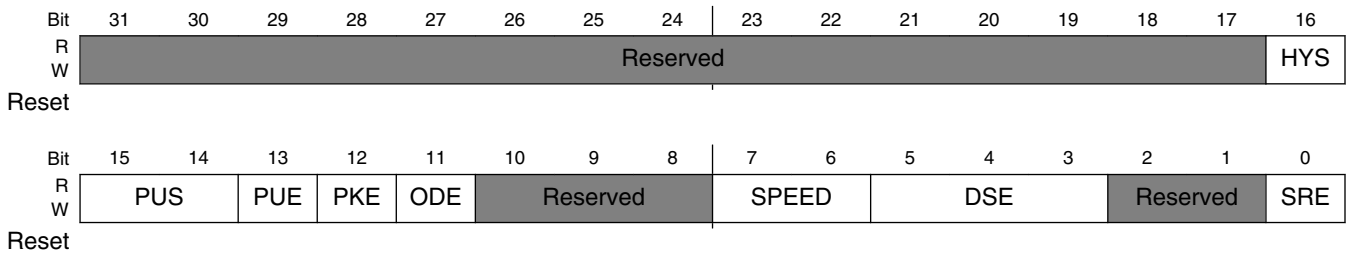
IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_01 field descriptions (continued)

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_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_01 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_150_Ohm___3_3V__260_Ohm_1_8V__240_Ohm_for_DDR_** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V, 240 Ohm for DDR) 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_01 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.7.74 SW_PAD_CTL_PAD_GPIO_SD_00 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_00)

SW_PAD_CTL Register

Address: 401F_8000h base + 134h offset = 401F_8134h



IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_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_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_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_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_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_00

Table continues on the next page...

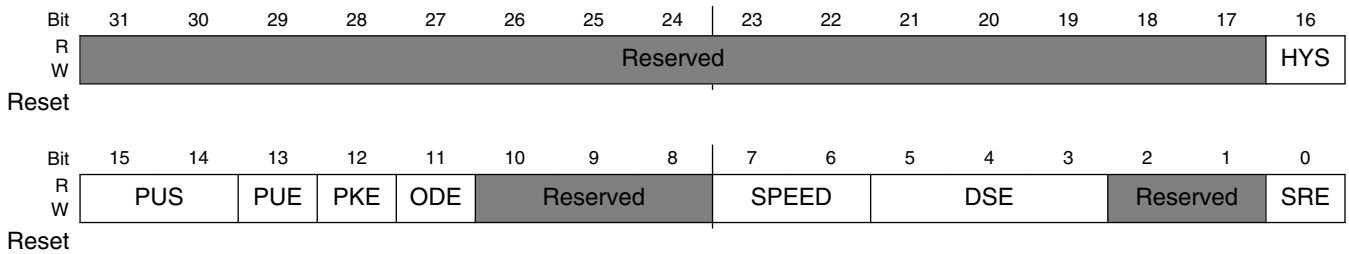
IOMUXC_SW_PAD_CTL_PAD_GPIO_SD_00 field descriptions (continued)

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_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_00 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_150_Ohm___3_3V__260_Ohm_1_8V__240_Ohm_for_DDR_** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V, 240 Ohm for DDR) 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_00 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.7.75 SW_PAD_CTL_PAD_GPIO_13 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_13)

SW_PAD_CTL Register

Address: 401F_8000h base + 138h offset = 401F_8138h



IOMUXC_SW_PAD_CTL_PAD_GPIO_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_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_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_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_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_13

Table continues on the next page...

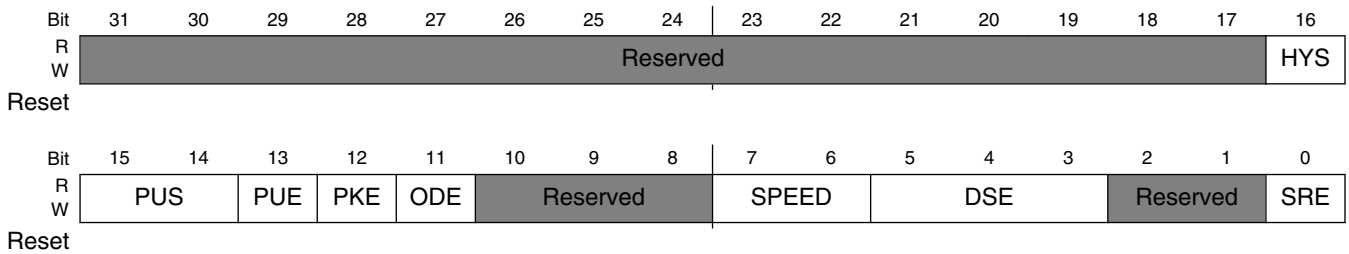
IOMUXC_SW_PAD_CTL_PAD_GPIO_13 field descriptions (continued)

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_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_13 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_150_Ohm___3_3V__260_Ohm_1_8V__240_Ohm_for_DDR_** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V, 240 Ohm for DDR) 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_13 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.7.76 SW_PAD_CTL_PAD_GPIO_12 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_12)

SW_PAD_CTL Register

Address: 401F_8000h base + 13Ch offset = 401F_813Ch



IOMUXC_SW_PAD_CTL_PAD_GPIO_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_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_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_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_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_12

Table continues on the next page...

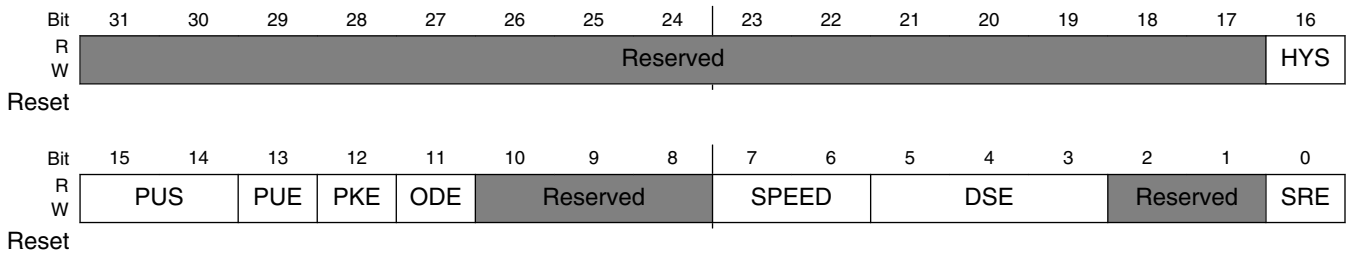
IOMUXC_SW_PAD_CTL_PAD_GPIO_12 field descriptions (continued)

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_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_12 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_150_Ohm___3_3V__260_Ohm_1_8V__240_Ohm_for_DDR_** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V, 240 Ohm for DDR) 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_12 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.7.77 SW_PAD_CTL_PAD_GPIO_11 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_11)

SW_PAD_CTL Register

Address: 401F_8000h base + 140h offset = 401F_8140h



IOMUXC_SW_PAD_CTL_PAD_GPIO_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_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_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_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_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_11

Table continues on the next page...

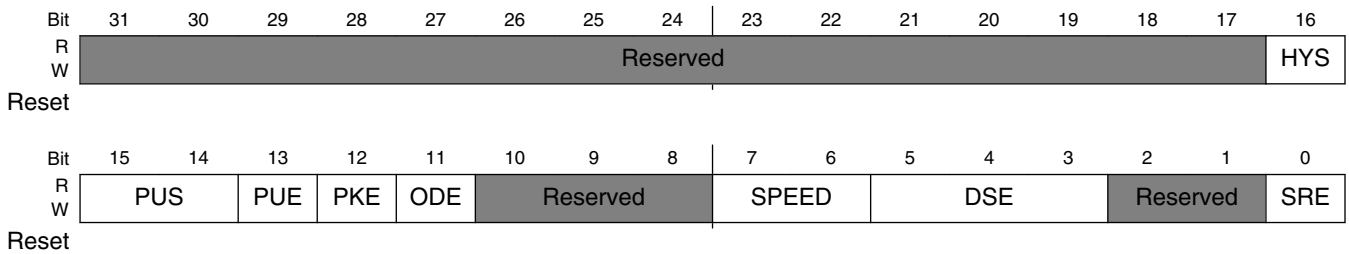
IOMUXC_SW_PAD_CTL_PAD_GPIO_11 field descriptions (continued)

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_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_11 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_150_Ohm___3_3V__260_Ohm_1_8V__240_Ohm_for_DDR_** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V, 240 Ohm for DDR) 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_11 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.7.78 SW_PAD_CTL_PAD_GPIO_10 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_10)

SW_PAD_CTL Register

Address: 401F_8000h base + 144h offset = 401F_8144h



IOMUXC_SW_PAD_CTL_PAD_GPIO_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_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_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_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_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_10

Table continues on the next page...

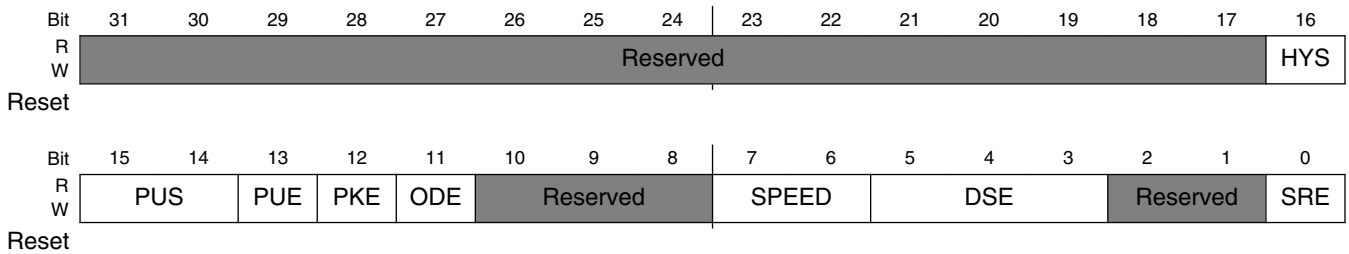
IOMUXC_SW_PAD_CTL_PAD_GPIO_10 field descriptions (continued)

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_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_10 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_150_Ohm___3_3V__260_Ohm_1_8V__240_Ohm_for_DDR_** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V, 240 Ohm for DDR) 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_10 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.7.79 SW_PAD_CTL_PAD_GPIO_09 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_09)

SW_PAD_CTL Register

Address: 401F_8000h base + 148h offset = 401F_8148h



IOMUXC_SW_PAD_CTL_PAD_GPIO_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_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_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_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_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_09

Table continues on the next page...

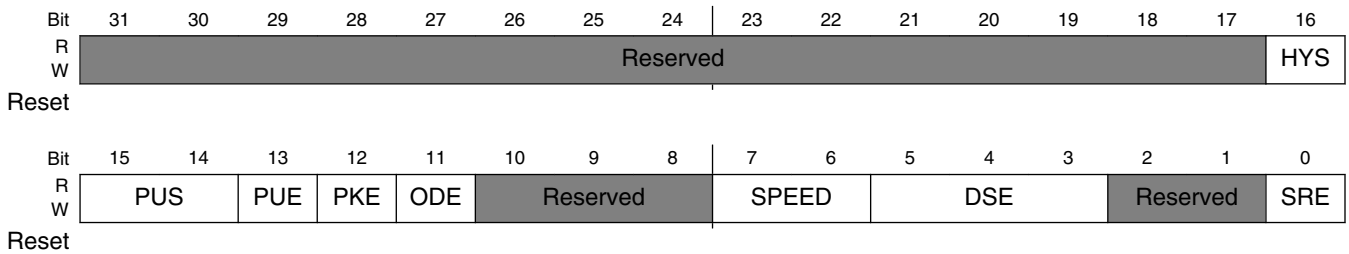
IOMUXC_SW_PAD_CTL_PAD_GPIO_09 field descriptions (continued)

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_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_09 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_150_Ohm___3_3V__260_Ohm_1_8V__240_Ohm_for_DDR_** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V, 240 Ohm for DDR) 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_09 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.7.80 SW_PAD_CTL_PAD_GPIO_08 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_08)

SW_PAD_CTL Register

Address: 401F_8000h base + 14Ch offset = 401F_814Ch



IOMUXC_SW_PAD_CTL_PAD_GPIO_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_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_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_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_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_08

Table continues on the next page...

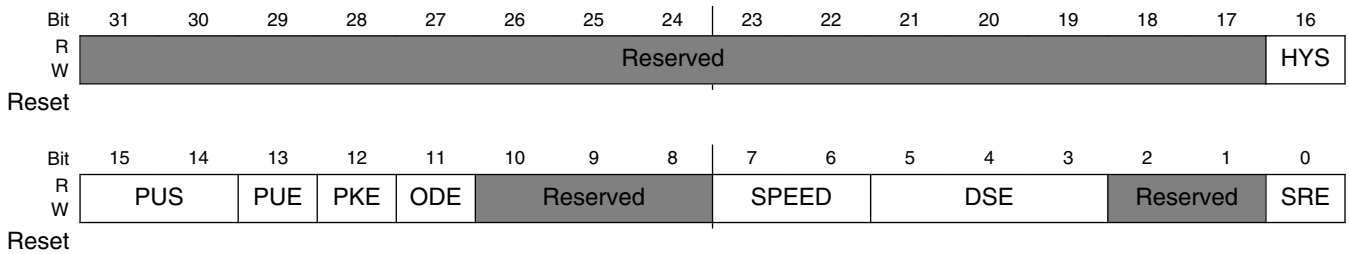
IOMUXC_SW_PAD_CTL_PAD_GPIO_08 field descriptions (continued)

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_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_08 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_150_Ohm___3_3V__260_Ohm_1_8V__240_Ohm_for_DDR_** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V, 240 Ohm for DDR) 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_08 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.7.81 SW_PAD_CTL_PAD_GPIO_07 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_07)

SW_PAD_CTL Register

Address: 401F_8000h base + 150h offset = 401F_8150h



IOMUXC_SW_PAD_CTL_PAD_GPIO_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_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_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_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_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_07

Table continues on the next page...

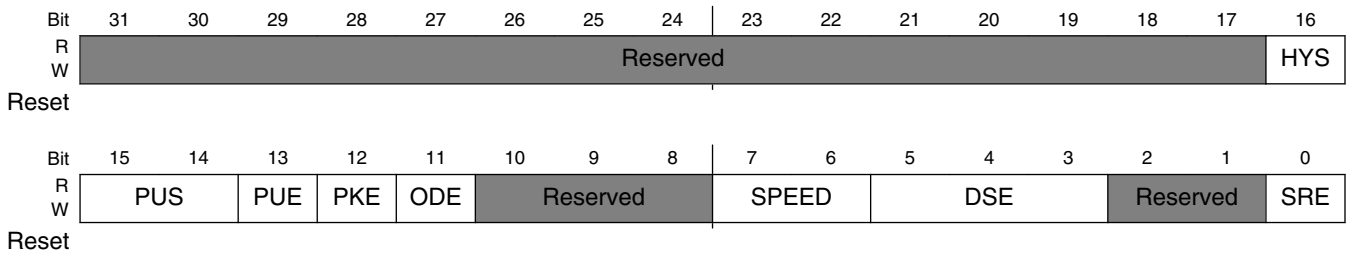
IOMUXC_SW_PAD_CTL_PAD_GPIO_07 field descriptions (continued)

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_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_07 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_150_Ohm___3_3V__260_Ohm_1_8V__240_Ohm_for_DDR_** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V, 240 Ohm for DDR) 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_07 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.7.82 SW_PAD_CTL_PAD_GPIO_06 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_06)

SW_PAD_CTL Register

Address: 401F_8000h base + 154h offset = 401F_8154h



IOMUXC_SW_PAD_CTL_PAD_GPIO_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_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_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_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_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_06

Table continues on the next page...

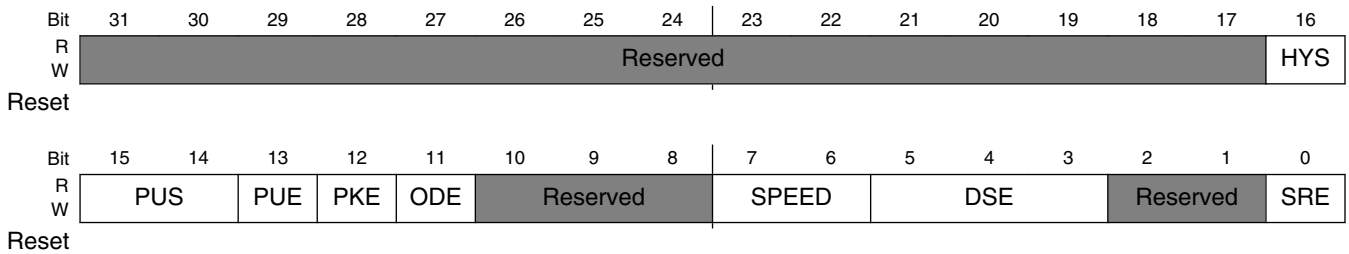
IOMUXC_SW_PAD_CTL_PAD_GPIO_06 field descriptions (continued)

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_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_06 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_150_Ohm___3_3V__260_Ohm_1_8V__240_Ohm_for_DDR_** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V, 240 Ohm for DDR) 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_06 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.7.83 SW_PAD_CTL_PAD_GPIO_05 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_05)

SW_PAD_CTL Register

Address: 401F_8000h base + 158h offset = 401F_8158h



IOMUXC_SW_PAD_CTL_PAD_GPIO_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_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_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_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_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_05

Table continues on the next page...

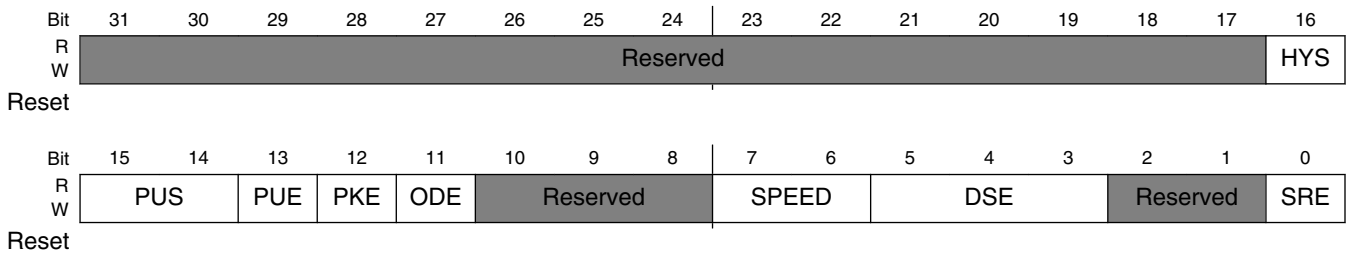
IOMUXC_SW_PAD_CTL_PAD_GPIO_05 field descriptions (continued)

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_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_05 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_150_Ohm___3_3V__260_Ohm_1_8V__240_Ohm_for_DDR_** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V, 240 Ohm for DDR) 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_05 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.7.84 SW_PAD_CTL_PAD_GPIO_04 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_04)

SW_PAD_CTL Register

Address: 401F_8000h base + 15Ch offset = 401F_815Ch



IOMUXC_SW_PAD_CTL_PAD_GPIO_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_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_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_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_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_04

Table continues on the next page...

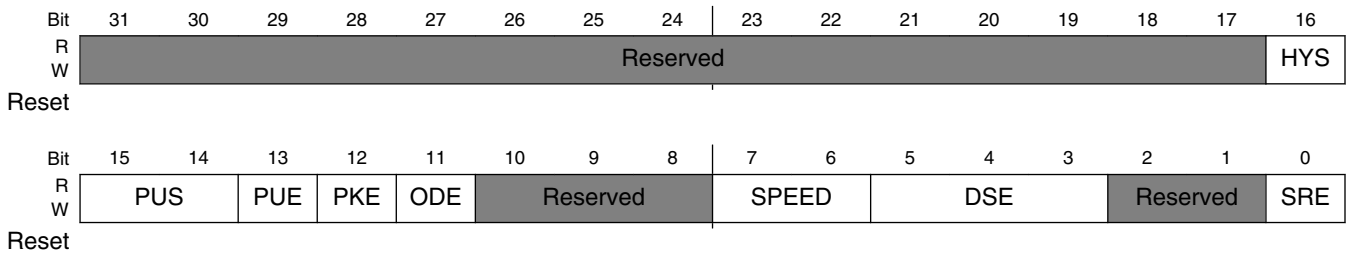
IOMUXC_SW_PAD_CTL_PAD_GPIO_04 field descriptions (continued)

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_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_04 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_150_Ohm___3_3V__260_Ohm_1_8V__240_Ohm_for_DDR_** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V, 240 Ohm for DDR) 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_04 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.7.85 SW_PAD_CTL_PAD_GPIO_03 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_03)

SW_PAD_CTL Register

Address: 401F_8000h base + 160h offset = 401F_8160h



IOMUXC_SW_PAD_CTL_PAD_GPIO_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_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_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_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_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_03

Table continues on the next page...

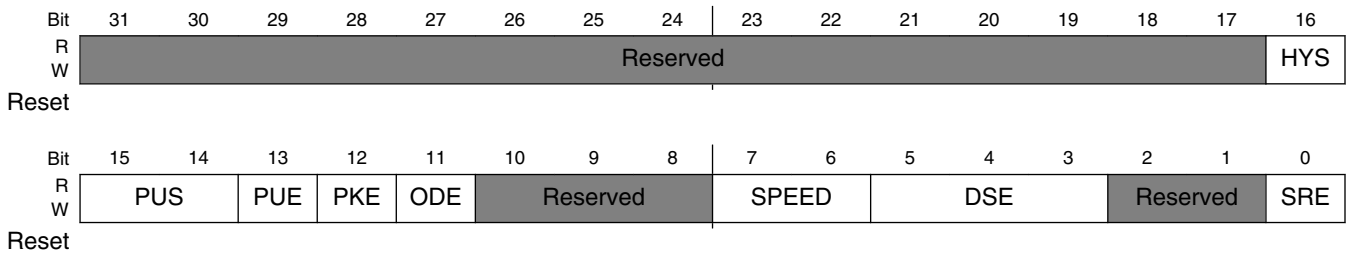
IOMUXC_SW_PAD_CTL_PAD_GPIO_03 field descriptions (continued)

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_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_03 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_150_Ohm___3_3V__260_Ohm_1_8V__240_Ohm_for_DDR_** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V, 240 Ohm for DDR) 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_03 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.7.86 SW_PAD_CTL_PAD_GPIO_02 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_02)

SW_PAD_CTL Register

Address: 401F_8000h base + 164h offset = 401F_8164h



IOMUXC_SW_PAD_CTL_PAD_GPIO_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_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_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_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_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_02

Table continues on the next page...

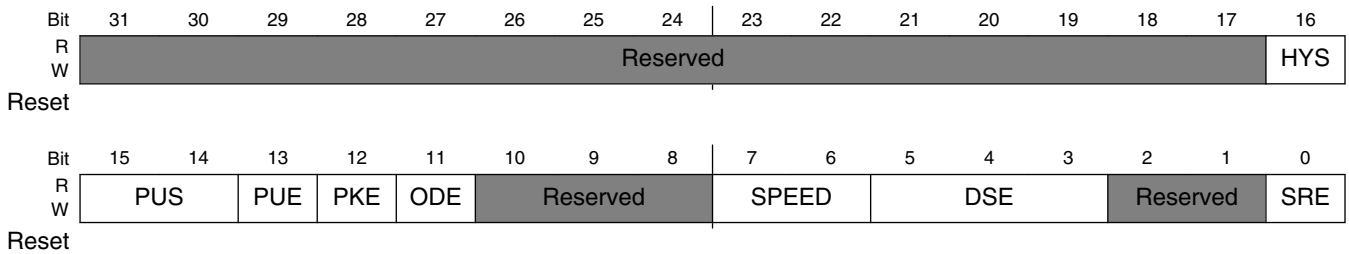
IOMUXC_SW_PAD_CTL_PAD_GPIO_02 field descriptions (continued)

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_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_02 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_150_Ohm___3_3V__260_Ohm_1_8V__240_Ohm_for_DDR_** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V, 240 Ohm for DDR) 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_02 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.7.87 SW_PAD_CTL_PAD_GPIO_01 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_01)

SW_PAD_CTL Register

Address: 401F_8000h base + 168h offset = 401F_8168h



IOMUXC_SW_PAD_CTL_PAD_GPIO_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_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_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_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_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_01

Table continues on the next page...

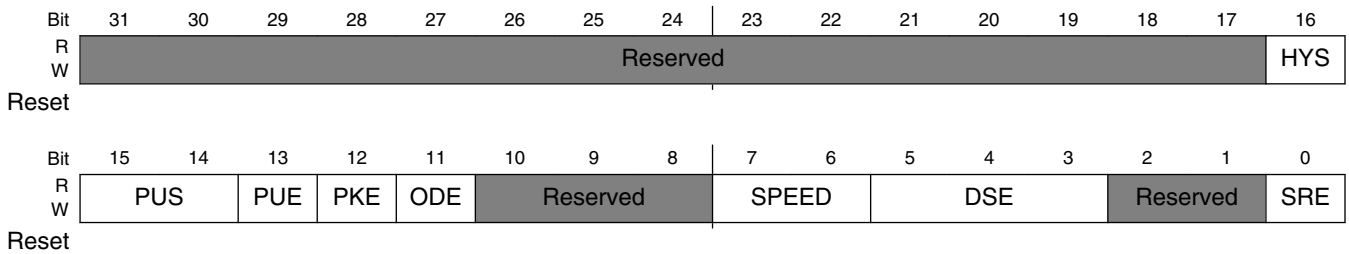
IOMUXC_SW_PAD_CTL_PAD_GPIO_01 field descriptions (continued)

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_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_01 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_150_Ohm___3_3V__260_Ohm_1_8V__240_Ohm_for_DDR_** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V, 240 Ohm for DDR) 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_01 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.7.88 SW_PAD_CTL_PAD_GPIO_00 SW PAD Control Register (IOMUXC_SW_PAD_CTL_PAD_GPIO_00)

SW_PAD_CTL Register

Address: 401F_8000h base + 16Ch offset = 401F_816Ch



IOMUXC_SW_PAD_CTL_PAD_GPIO_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_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_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_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_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_00

Table continues on the next page...

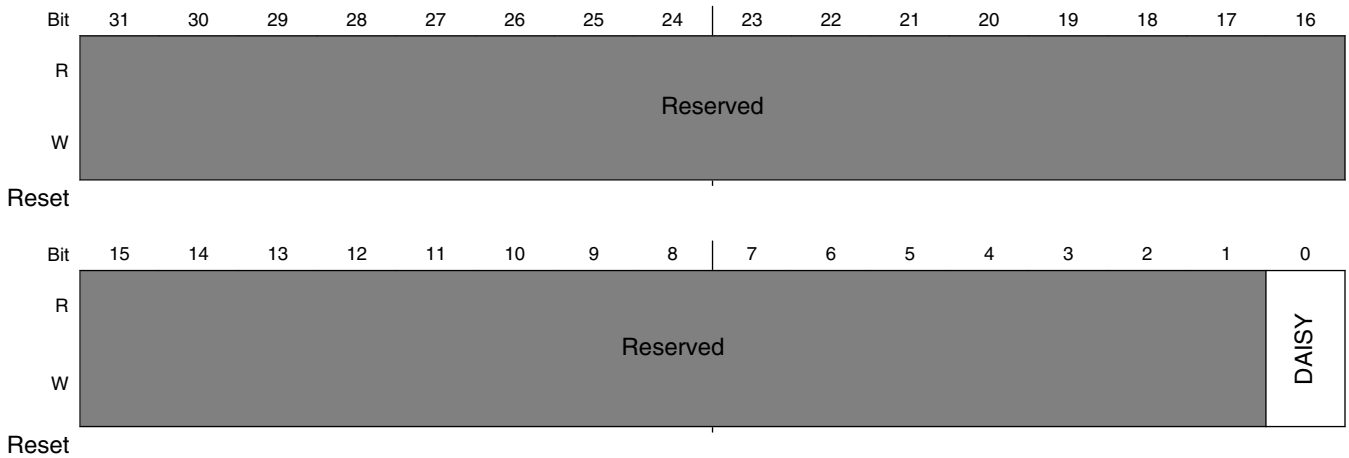
IOMUXC_SW_PAD_CTL_PAD_GPIO_00 field descriptions (continued)

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_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_00 000 **DSE_0_output_driver_disabled_** — output driver disabled; 001 **DSE_1_R0_150_Ohm___3_3V__260_Ohm_1_8V__240_Ohm_for_DDR_** — R0(150 Ohm @ 3.3V, 260 Ohm@1.8V, 240 Ohm for DDR) 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_00 0 **SRE_0_Slow_Slew_Rate** — Slow Slew Rate 1 **SRE_1_Fast_Slew_Rate** — Fast Slew Rate

11.7.89 USB_OTG_ID_SELECT_INPUT DAISY Register (IOMUXC_USB_OTG_ID_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 170h offset = 401F_8170h



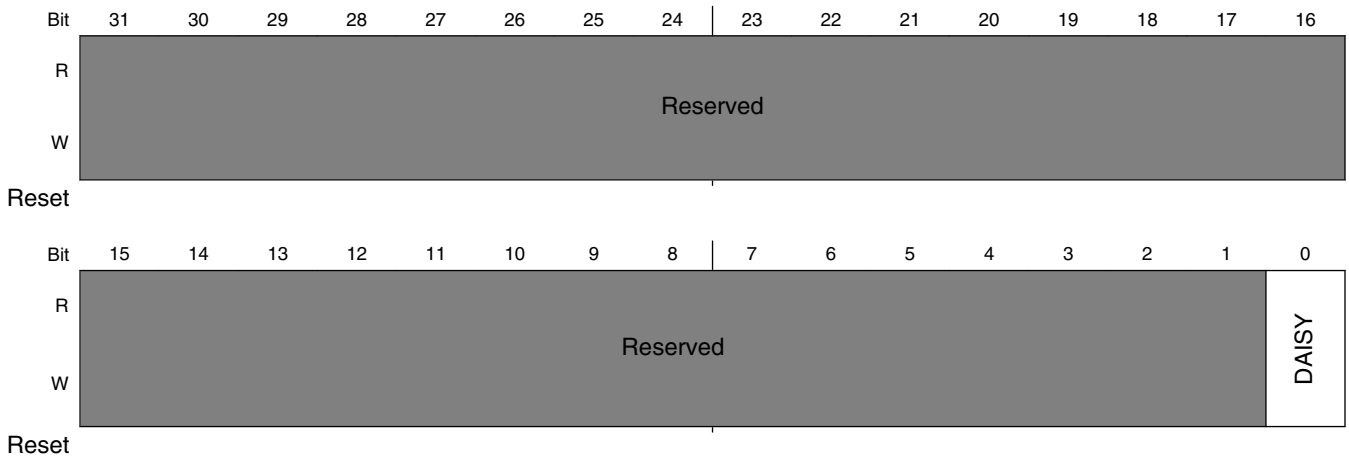
IOMUXC_USB_OTG_ID_SELECT_INPUT field descriptions

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_10_ALT6** — Selecting Pad: GPIO_AD_10 for Mode: ALT6 1 **GPIO_13_ALT3** — Selecting Pad: GPIO_13 for Mode: ALT3

11.7.90 FLEXPWM1_PWMA_SELECT_INPUT_0 DAISY Register (IOMUXC_FLEXPWM1_PWMA_SELECT_INPUT_0)

DAISY Register

Address: 401F_8000h base + 174h offset = 401F_8174h



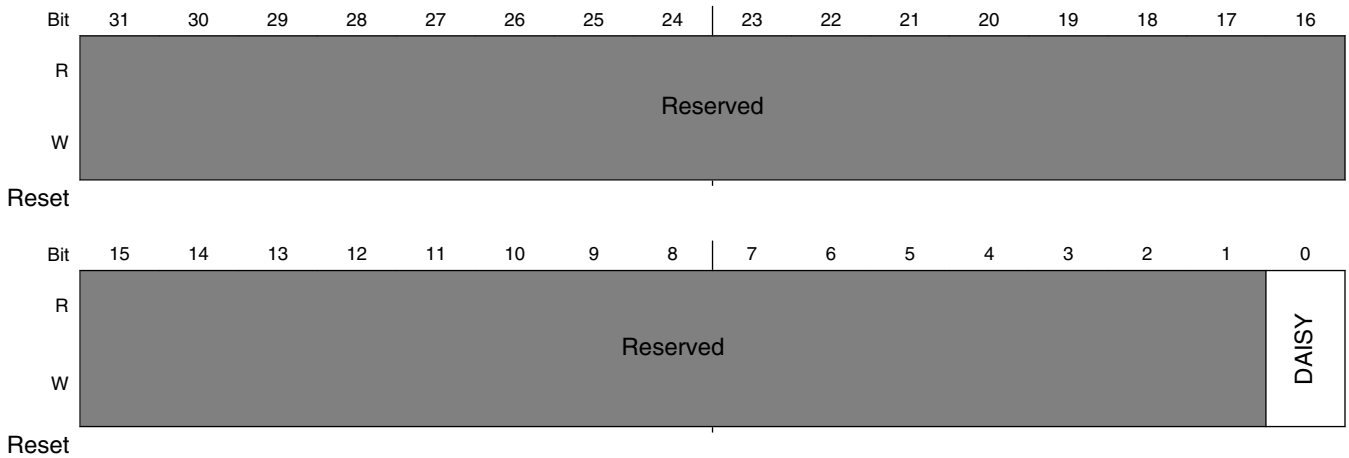
IOMUXC_FLEXPWM1_PWMA_SELECT_INPUT_0 field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. instance: FLEXPWM1, In Pin: pwma0 0 **GPIO_SD_02_ALT2** — Selecting Pad: GPIO_SD_02 for Mode: ALT2 1 **GPIO_02_ALT2** — Selecting Pad: GPIO_02 for Mode: ALT2

11.7.91 FLEXPWM1_PWMA_SELECT_INPUT_1 DAISY Register (IOMUXC_FLEXPWM1_PWMA_SELECT_INPUT_1)

DAISY Register

Address: 401F_8000h base + 178h offset = 401F_8178h



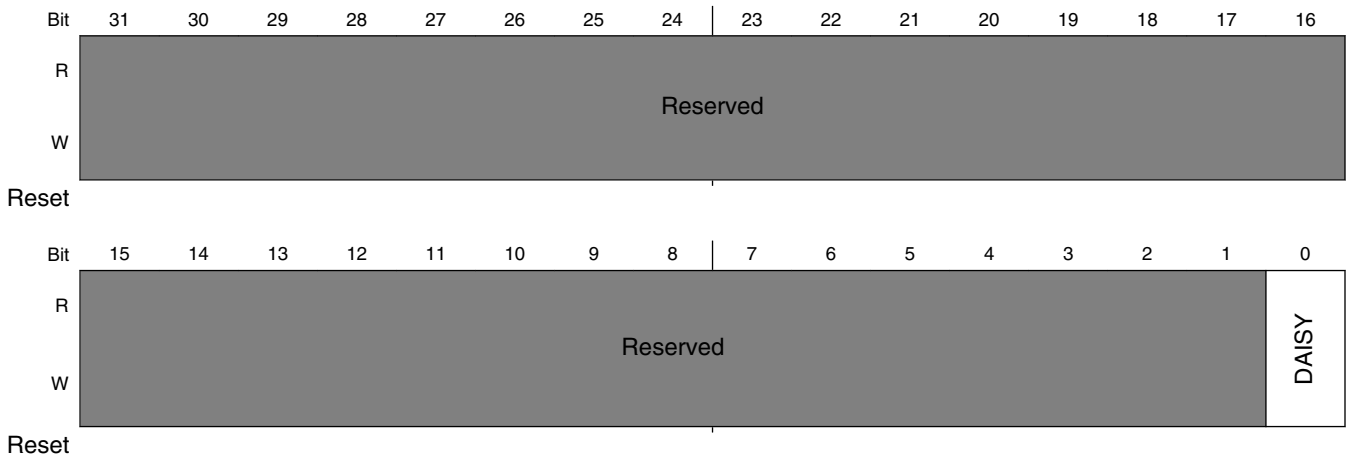
IOMUXC_FLEXPWM1_PWMA_SELECT_INPUT_1 field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. instance: FLEXPWM1, In Pin: pwma1 0 **GPIO_SD_04_ALT2** — Selecting Pad: GPIO_SD_04 for Mode: ALT2 1 **GPIO_04_ALT2** — Selecting Pad: GPIO_04 for Mode: ALT2

11.7.92 FLEXPWM1_PWMA_SELECT_INPUT_2 DAISY Register (IOMUXC_FLEXPWM1_PWMA_SELECT_INPUT_2)

DAISY Register

Address: 401F_8000h base + 17Ch offset = 401F_817Ch



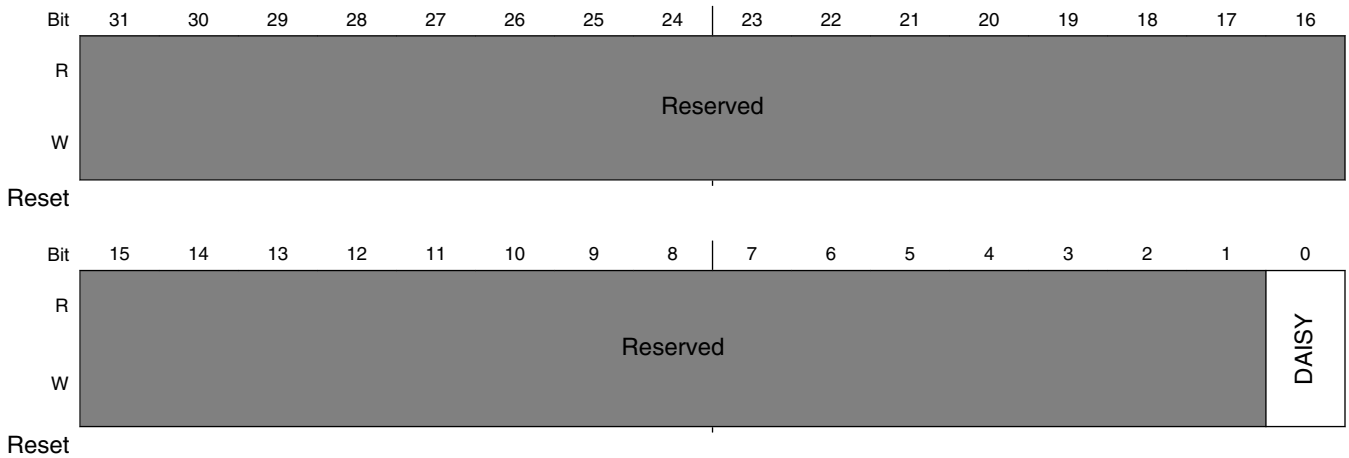
IOMUXC_FLEXPWM1_PWMA_SELECT_INPUT_2 field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. instance: FLEXPWM1, In Pin: pwma2 0 **GPIO_AD_04_ALT2** — Selecting Pad: GPIO_AD_04 for Mode: ALT2 1 **GPIO_06_ALT2** — Selecting Pad: GPIO_06 for Mode: ALT2

11.7.93 FLEXPWM1_PWMA_SELECT_INPUT_3 DAISY Register (IOMUXC_FLEXPWM1_PWMA_SELECT_INPUT_3)

DAISY Register

Address: 401F_8000h base + 180h offset = 401F_8180h



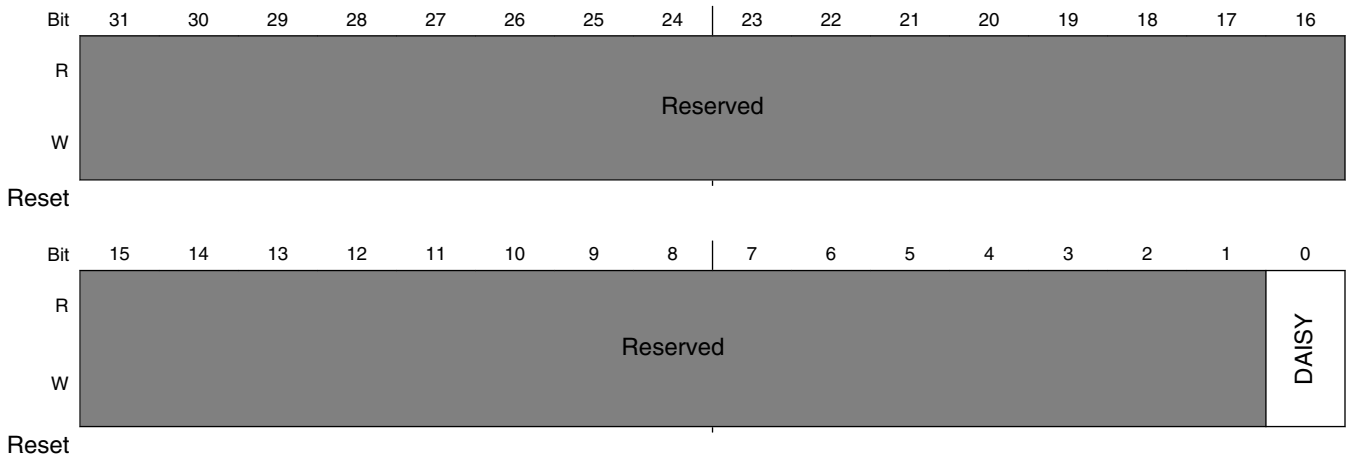
IOMUXC_FLEXPWM1_PWMA_SELECT_INPUT_3 field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. instance: FLEXPWM1, In Pin: pwma3 0 **GPIO_AD_06_ALT2** — Selecting Pad: GPIO_AD_06 for Mode: ALT2 1 **GPIO_08_ALT2** — Selecting Pad: GPIO_08 for Mode: ALT2

11.7.94 FLEXPWM1_PWMB_SELECT_INPUT_0 DAISY Register (IOMUXC_FLEXPWM1_PWMB_SELECT_INPUT_0)

DAISY Register

Address: 401F_8000h base + 184h offset = 401F_8184h



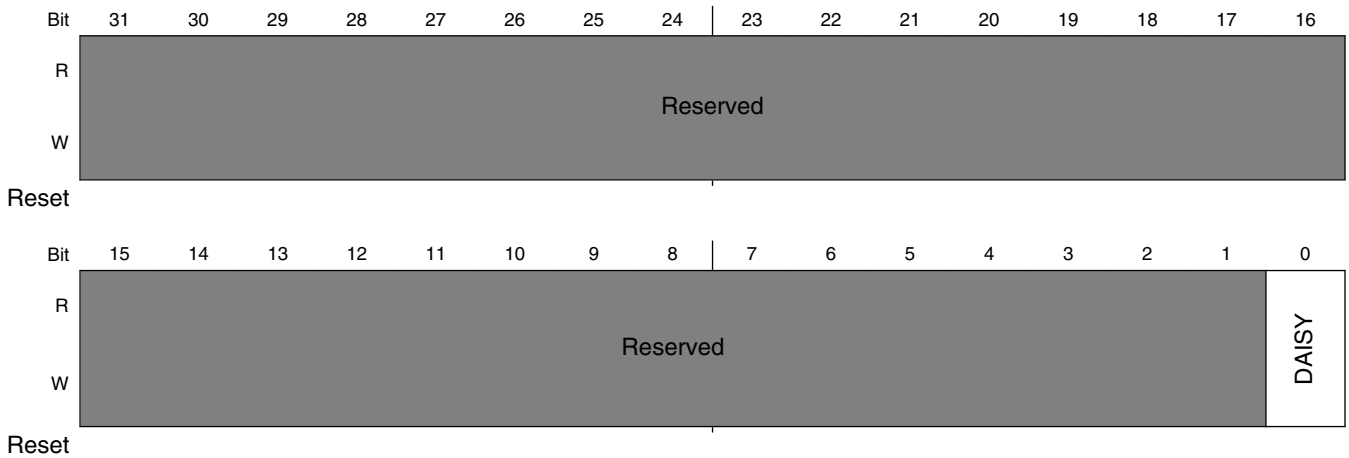
IOMUXC_FLEXPWM1_PWMB_SELECT_INPUT_0 field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. instance: FLEXPWM1, In Pin: pwmb0 0 **GPIO_SD_01_ALT2** — Selecting Pad: GPIO_SD_01 for Mode: ALT2 1 **GPIO_01_ALT2** — Selecting Pad: GPIO_01 for Mode: ALT2

11.7.95 FLEXPWM1_PWMB_SELECT_INPUT_1 DAISY Register (IOMUXC_FLEXPWM1_PWMB_SELECT_INPUT_1)

DAISY Register

Address: 401F_8000h base + 188h offset = 401F_8188h



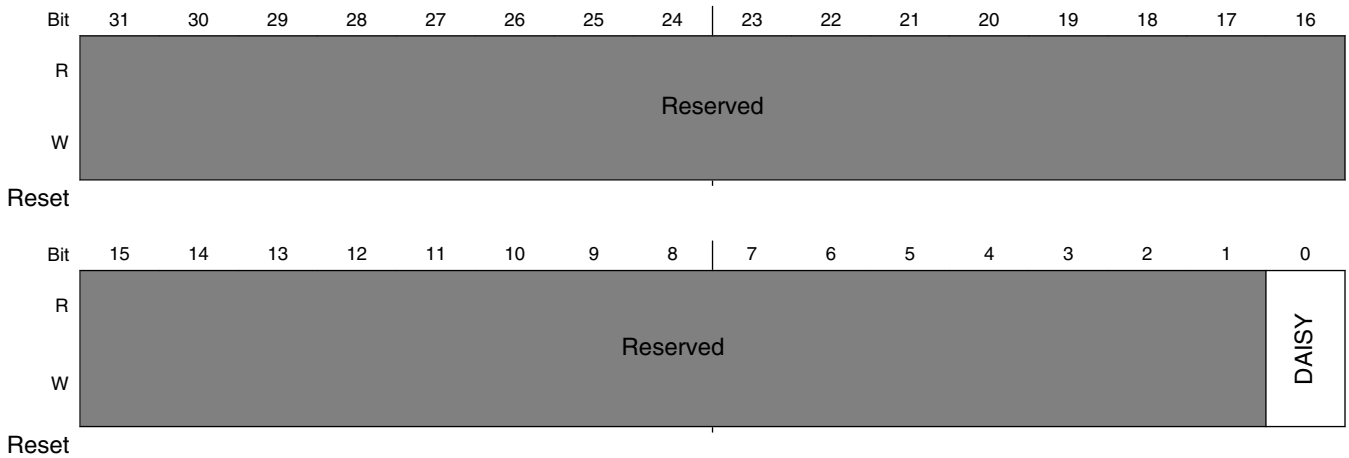
IOMUXC_FLEXPWM1_PWMB_SELECT_INPUT_1 field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. instance: FLEXPWM1, In Pin: pwmb1 0 **GPIO_SD_03_ALT2** — Selecting Pad: GPIO_SD_03 for Mode: ALT2 1 **GPIO_03_ALT2** — Selecting Pad: GPIO_03 for Mode: ALT2

11.7.96 FLEXPWM1_PWMB_SELECT_INPUT_2 DAISY Register (IOMUXC_FLEXPWM1_PWMB_SELECT_INPUT_2)

DAISY Register

Address: 401F_8000h base + 18Ch offset = 401F_818Ch



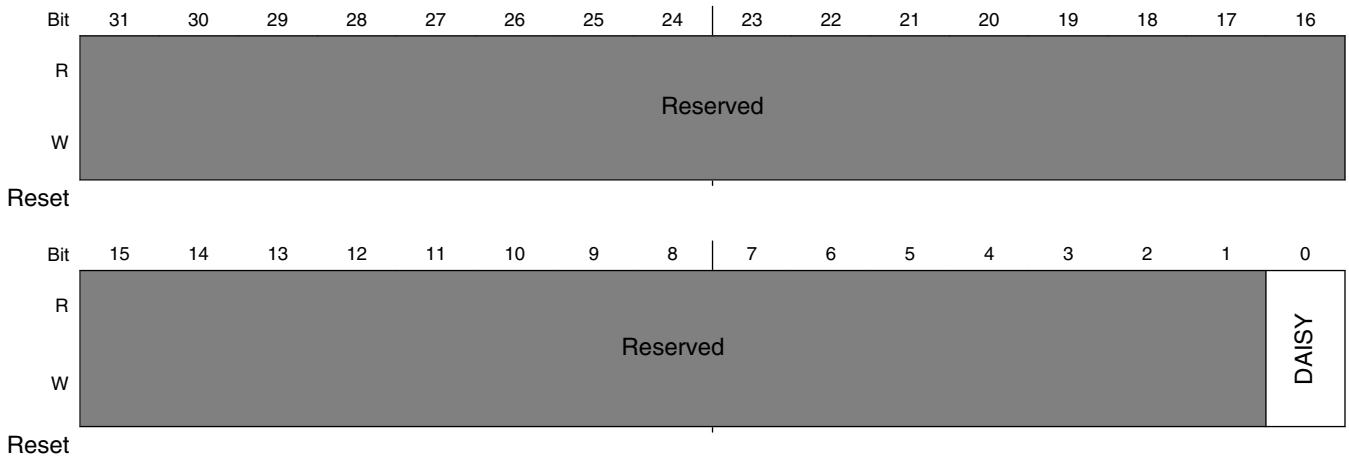
IOMUXC_FLEXPWM1_PWMB_SELECT_INPUT_2 field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. instance: FLEXPWM1, In Pin: pwmb2 0 **GPIO_AD_03_ALT2** — Selecting Pad: GPIO_AD_03 for Mode: ALT2 1 **GPIO_05_ALT2** — Selecting Pad: GPIO_05 for Mode: ALT2

11.7.97 FLEXPWM1_PWMB_SELECT_INPUT_3 DAISY Register (IOMUXC_FLEXPWM1_PWMB_SELECT_INPUT_3)

DAISY Register

Address: 401F_8000h base + 190h offset = 401F_8190h



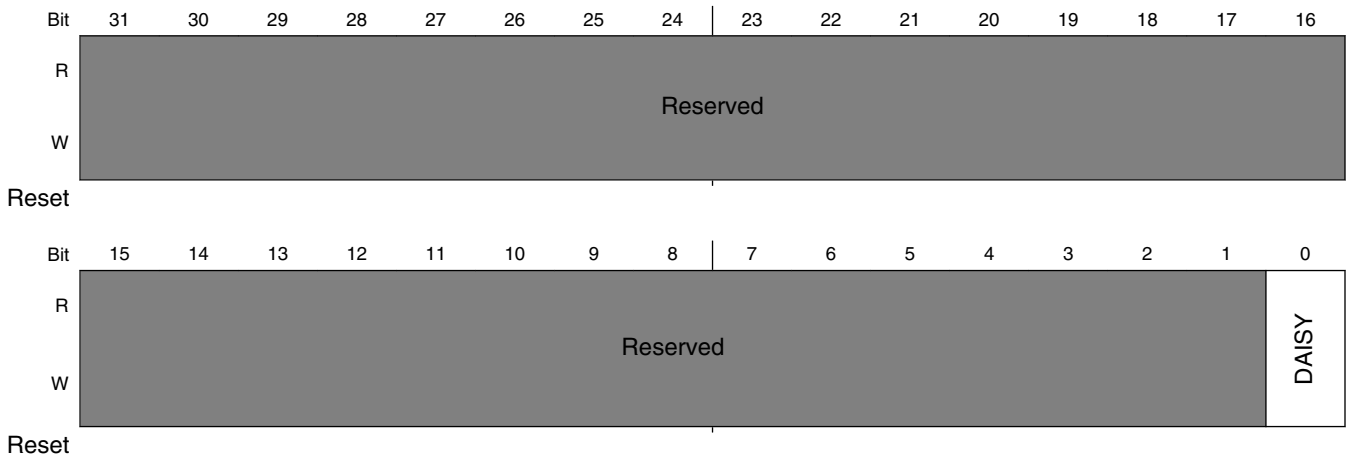
IOMUXC_FLEXPWM1_PWMB_SELECT_INPUT_3 field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. instance: FLEXPWM1, In Pin: pwmb3 0 **GPIO_AD_05_ALT2** — Selecting Pad: GPIO_AD_05 for Mode: ALT2 1 **GPIO_07_ALT2** — Selecting Pad: GPIO_07 for Mode: ALT2

11.7.98 FLEXSPI_DQS_FA_SELECT_INPUT DAISY Register (IOMUXC_FLEXSPI_DQS_FA_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 194h offset = 401F_8194h



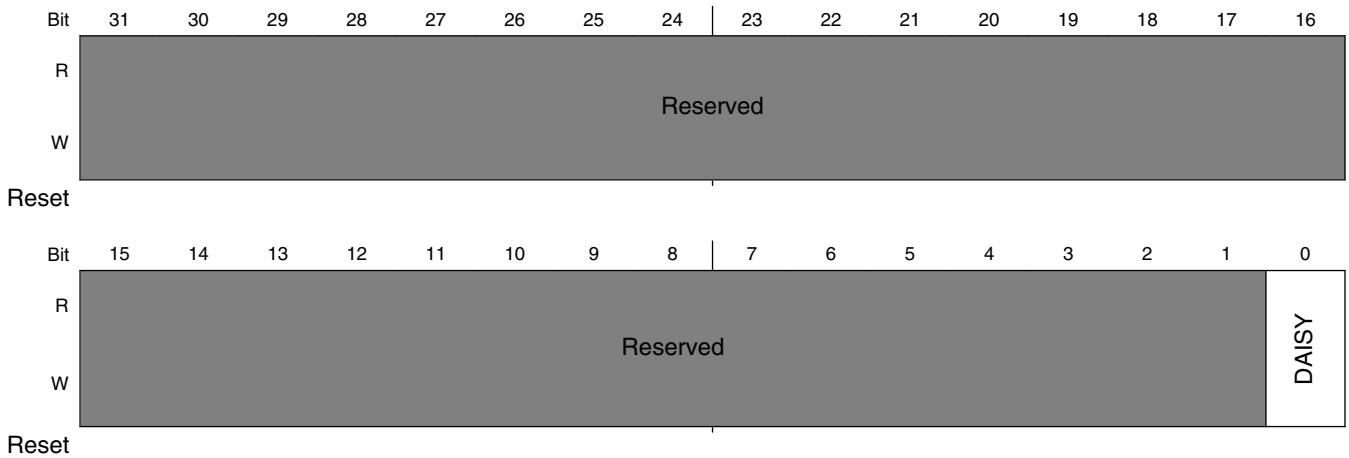
IOMUXC_FLEXSPI_DQS_FA_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: FLEXSPI, In Pin: dqs_fa 0 **GPIO_SD_14_ALT0** — Selecting Pad: GPIO_SD_14 for Mode: ALT0 1 **GPIO_SD_12_ALT0** — Selecting Pad: GPIO_SD_12 for Mode: ALT0

11.7.99 FLEXSPI_DQS_FB_SELECT_INPUT DAISY Register (IOMUXC_FLEXSPI_DQS_FB_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 198h offset = 401F_8198h



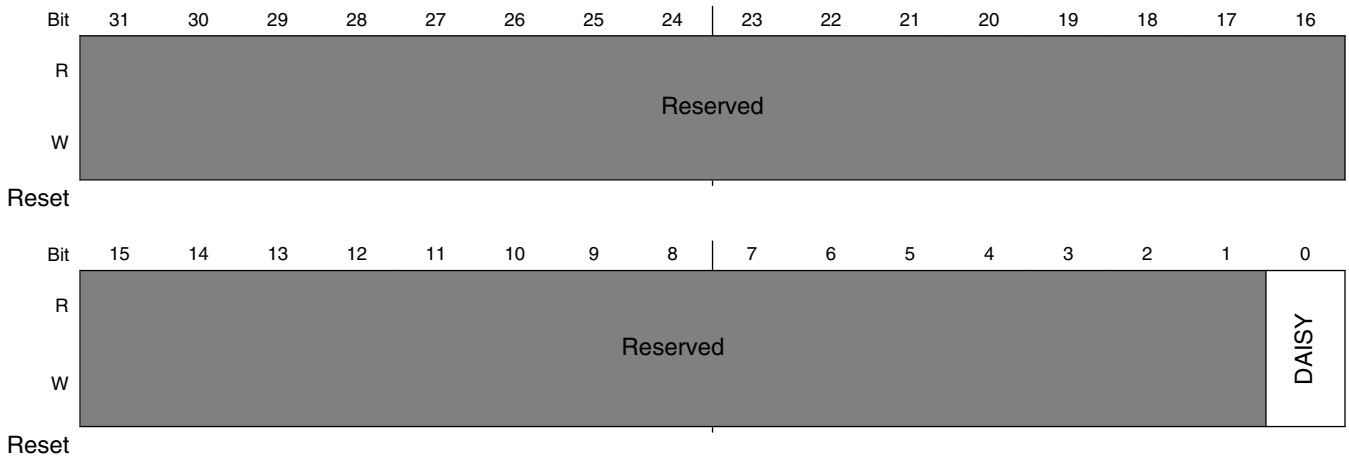
IOMUXC_FLEXSPI_DQS_FB_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: FLEXSPI, In Pin: dqs_fb 0 **GPIO_SD_14_ALT1** — Selecting Pad: GPIO_SD_14 for Mode: ALT1 1 **GPIO_00_ALT0** — Selecting Pad: GPIO_00 for Mode: ALT0

11.7.100 KPP_COL_SELECT_INPUT_0 DAISY Register (IOMUXC_KPP_COL_SELECT_INPUT_0)

DAISY Register

Address: 401F_8000h base + 19Ch offset = 401F_819Ch



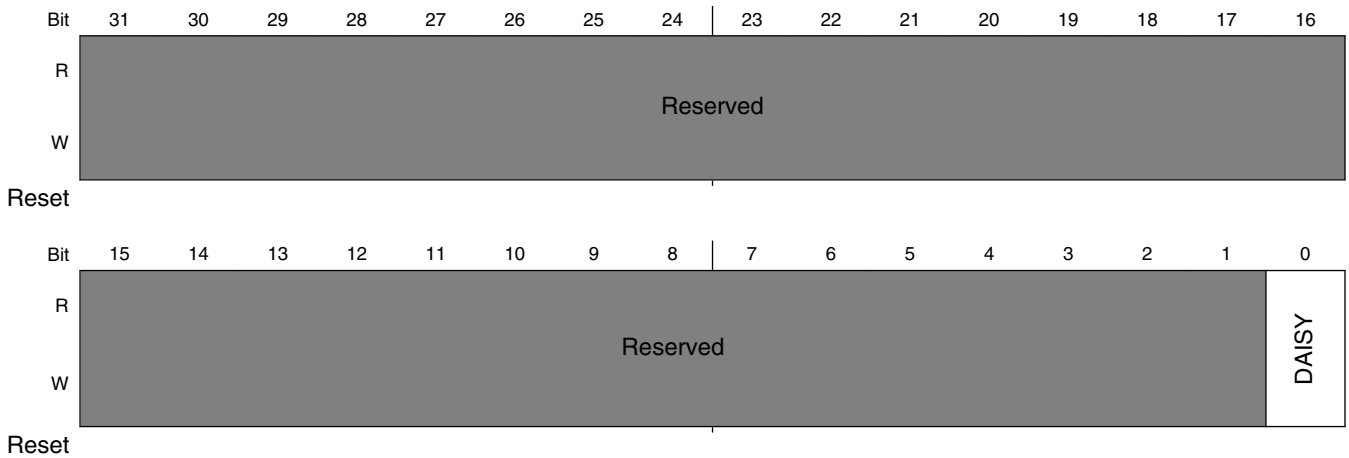
IOMUXC_KPP_COL_SELECT_INPUT_0 field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. instance: KPP, In Pin: col0 0 **GPIO_AD_14_ALT2** — Selecting Pad: GPIO_AD_14 for Mode: ALT2 1 **GPIO_12_ALT2** — Selecting Pad: GPIO_12 for Mode: ALT2

11.7.101 KPP_COL_SELECT_INPUT_1 DAISY Register (IOMUXC_KPP_COL_SELECT_INPUT_1)

DAISY Register

Address: 401F_8000h base + 1A0h offset = 401F_81A0h



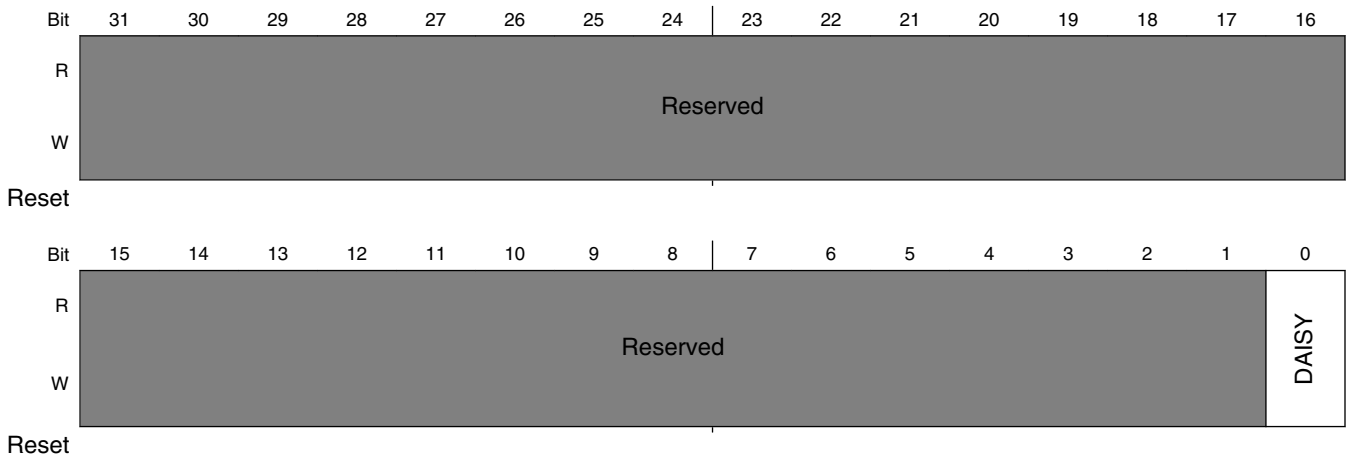
IOMUXC_KPP_COL_SELECT_INPUT_1 field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. instance: KPP, In Pin: col1 0 **GPIO_AD_12_ALT2** — Selecting Pad: GPIO_AD_12 for Mode: ALT2 1 **GPIO_AD_06_ALT3** — Selecting Pad: GPIO_AD_06 for Mode: ALT3

11.7.102 KPP_COL_SELECT_INPUT_2 DAISY Register (IOMUXC_KPP_COL_SELECT_INPUT_2)

DAISY Register

Address: 401F_8000h base + 1A4h offset = 401F_81A4h



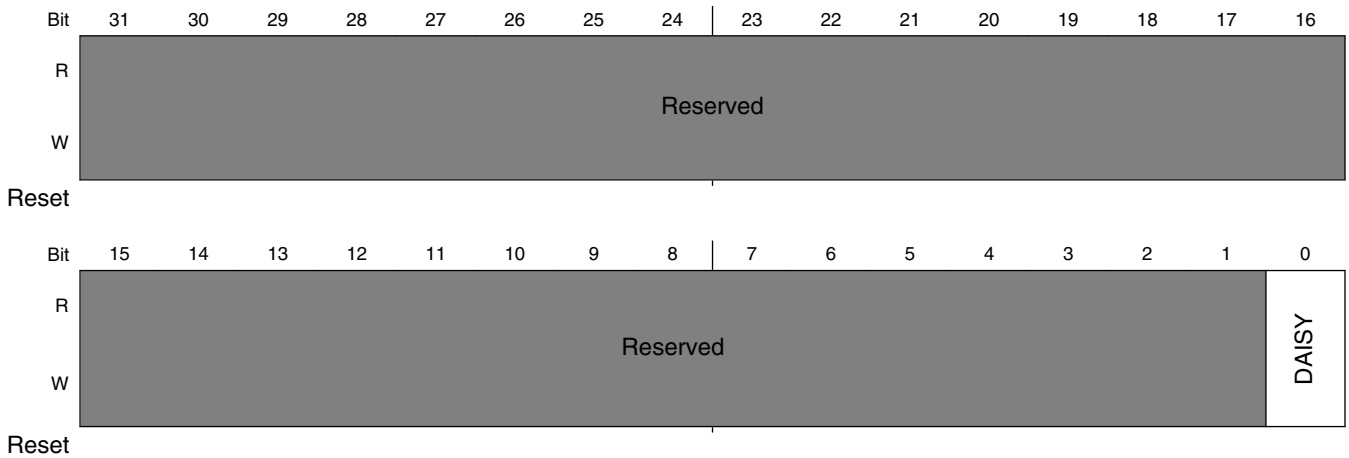
IOMUXC_KPP_COL_SELECT_INPUT_2 field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. instance: KPP, In Pin: col2 0 **GPIO_AD_10_ALT2** — Selecting Pad: GPIO_AD_10 for Mode: ALT2 1 **GPIO_AD_04_ALT3** — Selecting Pad: GPIO_AD_04 for Mode: ALT3

11.7.103 KPP_COL_SELECT_INPUT_3 DAISY Register (IOMUXC_KPP_COL_SELECT_INPUT_3)

DAISY Register

Address: 401F_8000h base + 1A8h offset = 401F_81A8h



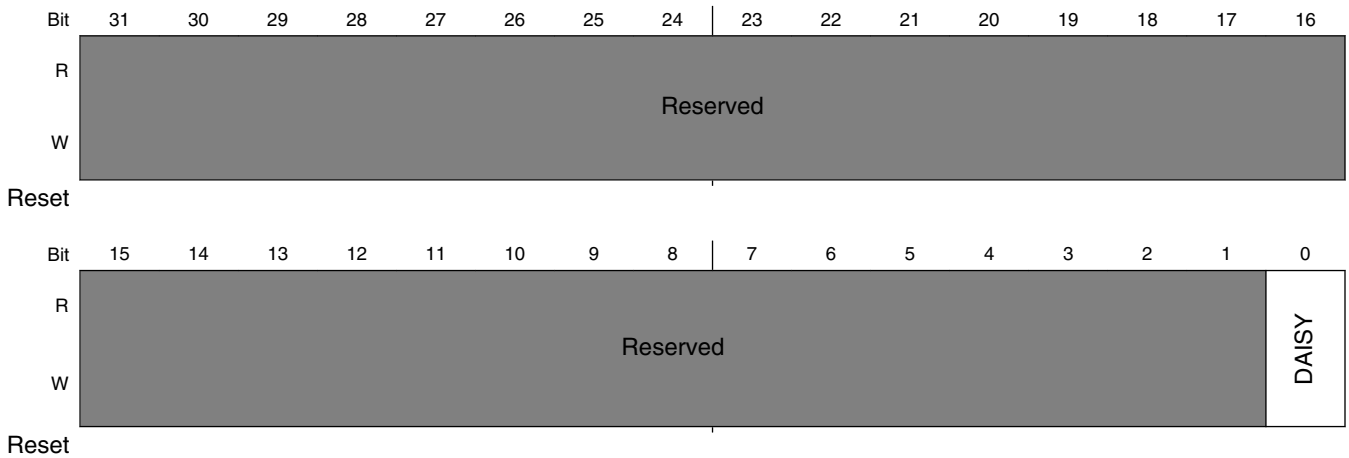
IOMUXC_KPP_COL_SELECT_INPUT_3 field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. instance: KPP, In Pin: col3 0 **GPIO_AD_00_ALT2** — Selecting Pad: GPIO_AD_00 for Mode: ALT2 1 **GPIO_02_ALT4** — Selecting Pad: GPIO_02 for Mode: ALT4

11.7.104 KPP_ROW_SELECT_INPUT_0 DAISY Register (IOMUXC_KPP_ROW_SELECT_INPUT_0)

DAISY Register

Address: 401F_8000h base + 1ACh offset = 401F_81ACh



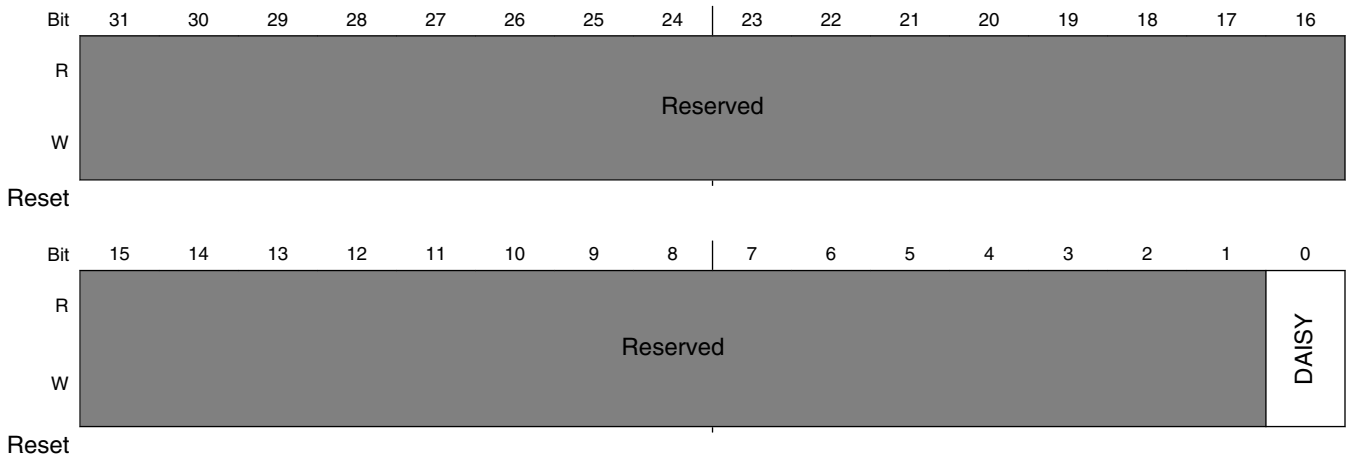
IOMUXC_KPP_ROW_SELECT_INPUT_0 field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. instance: KPP, In Pin: row0 0 **GPIO_AD_13_ALT2** — Selecting Pad: GPIO_AD_13 for Mode: ALT2 1 **GPIO_11_ALT2** — Selecting Pad: GPIO_11 for Mode: ALT2

11.7.105 KPP_ROW_SELECT_INPUT_1 DAISY Register (IOMUXC_KPP_ROW_SELECT_INPUT_1)

DAISY Register

Address: 401F_8000h base + 1B0h offset = 401F_81B0h



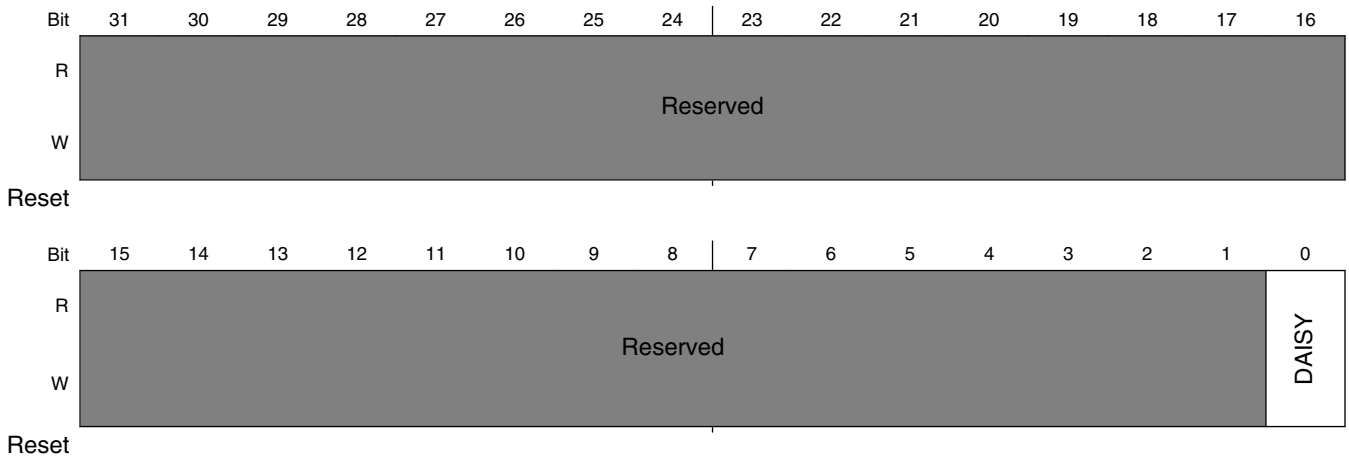
IOMUXC_KPP_ROW_SELECT_INPUT_1 field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. instance: KPP, In Pin: row1 0 **GPIO_AD_11_ALT2** — Selecting Pad: GPIO_AD_11 for Mode: ALT2 1 **GPIO_AD_05_ALT3** — Selecting Pad: GPIO_AD_05 for Mode: ALT3

11.7.106 KPP_ROW_SELECT_INPUT_2 DAISY Register (IOMUXC_KPP_ROW_SELECT_INPUT_2)

DAISY Register

Address: 401F_8000h base + 1B4h offset = 401F_81B4h



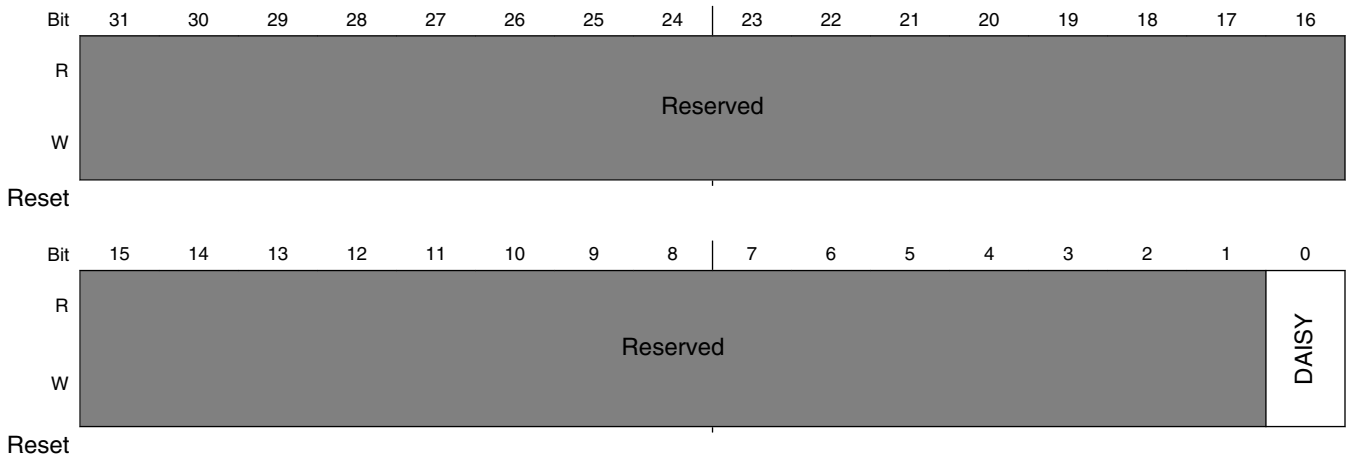
IOMUXC_KPP_ROW_SELECT_INPUT_2 field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. instance: KPP, In Pin: row2 0 **GPIO_AD_09_ALT2** — Selecting Pad: GPIO_AD_09 for Mode: ALT2 1 **GPIO_AD_03_ALT3** — Selecting Pad: GPIO_AD_03 for Mode: ALT3

11.7.107 KPP_ROW_SELECT_INPUT_3 DAISY Register (IOMUXC_KPP_ROW_SELECT_INPUT_3)

DAISY Register

Address: 401F_8000h base + 1B8h offset = 401F_81B8h



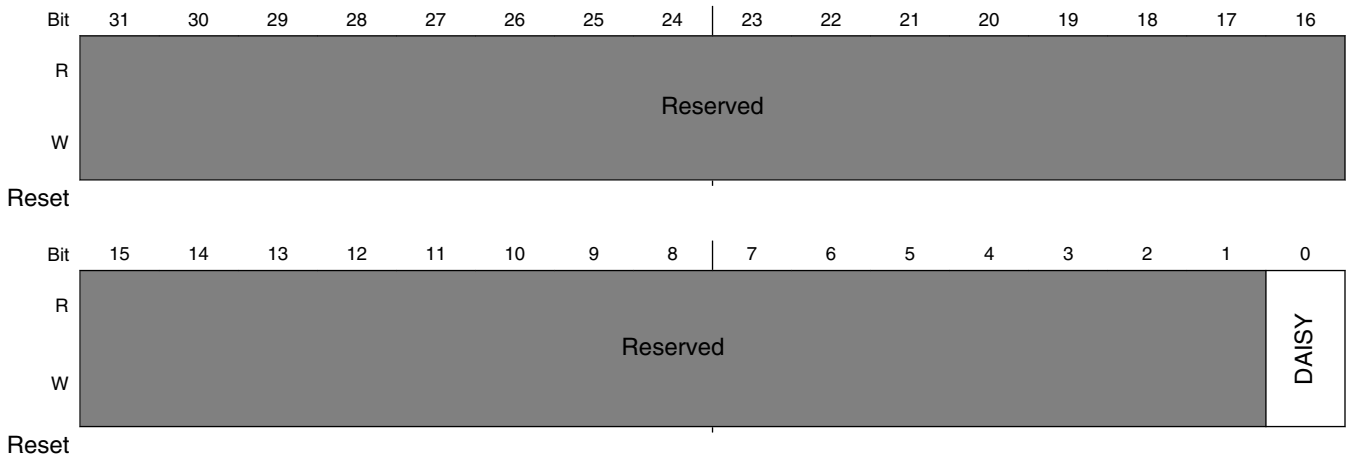
IOMUXC_KPP_ROW_SELECT_INPUT_3 field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. instance: KPP, In Pin: row3 0 **GPIO_13_ALT2** — Selecting Pad: GPIO_13 for Mode: ALT2 1 **GPIO_01_ALT4** — Selecting Pad: GPIO_01 for Mode: ALT4

11.7.108 LPI2C1_HREQ_SELECT_INPUT DAISY Register (IOMUXC_LPI2C1_HREQ_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 1BCh offset = 401F_81BCh



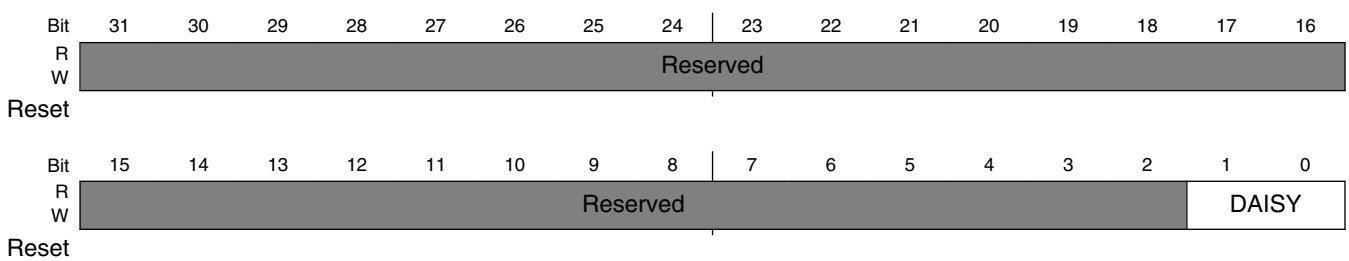
IOMUXC_LPI2C1_HREQ_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. instance: LPI2C1, In Pin: lpi2c_hreq 0 **GPIO_AD_06_ALT6** — Selecting Pad: GPIO_AD_06 for Mode: ALT6 1 **GPIO_10_ALT1** — Selecting Pad: GPIO_10 for Mode: ALT1

11.7.109 LPI2C1_SCL_SELECT_INPUT DAISY Register (IOMUXC_LPI2C1_SCL_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 1C0h offset = 401F_81C0h



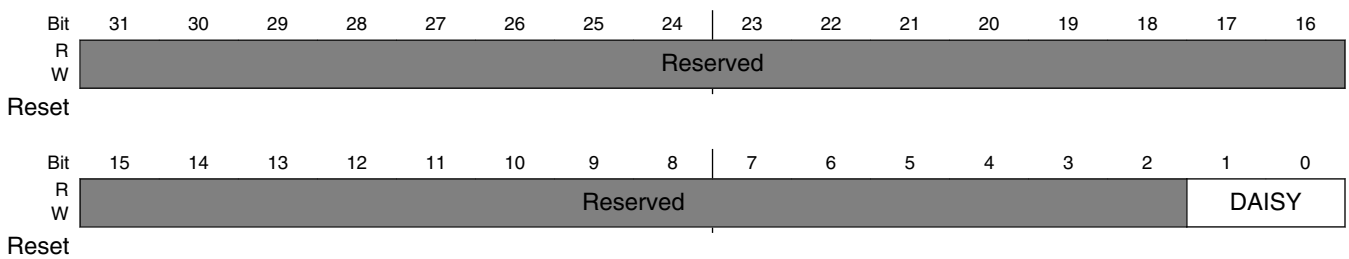
IOMUXC_LPI2C1_SCL_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. instance: LPI2C1, In Pin: lpi2c_scl 00 **GPIO_AD_14_ALT0** — Selecting Pad: GPIO_AD_14 for Mode: ALT0 01 **GPIO_SD_06_ALT1** — Selecting Pad: GPIO_SD_06 for Mode: ALT1 10 **GPIO_12_ALT1** — Selecting Pad: GPIO_12 for Mode: ALT1 11 **GPIO_02_ALT3** — Selecting Pad: GPIO_02 for Mode: ALT3

11.7.110 LPI2C1_SDA_SELECT_INPUT DAISY Register (IOMUXC_LPI2C1_SDA_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 1C4h offset = 401F_81C4h



IOMUXC_LPI2C1_SDA_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. instance: LPI2C1, In Pin: lpi2c_sda 00 **GPIO_AD_13_ALT0** — Selecting Pad: GPIO_AD_13 for Mode: ALT0 01 **GPIO_SD_05_ALT1** — Selecting Pad: GPIO_SD_05 for Mode: ALT1 10 **GPIO_11_ALT1** — Selecting Pad: GPIO_11 for Mode: ALT1 11 **GPIO_01_ALT3** — Selecting Pad: GPIO_01 for Mode: ALT3

11.7.111 LPI2C2_SCL_SELECT_INPUT DAISY Register (IOMUXC_LPI2C2_SCL_SELECT_INPUT)

DAISY 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																

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset																

IOMUXC_LPI2C2_SCL_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. instance: LPI2C2, In Pin: lpi2c_scl 00 **GPIO_AD_08_ALT0** — Selecting Pad: GPIO_AD_08 for Mode: ALT0 01 **GPIO_AD_02_ALT3** — Selecting Pad: GPIO_AD_02 for Mode: ALT3 10 **GPIO_SD_08_ALT1** — Selecting Pad: GPIO_SD_08 for Mode: ALT1 11 **GPIO_10_ALT3** — Selecting Pad: GPIO_10 for Mode: ALT3

11.7.112 LPI2C2_SDA_SELECT_INPUT DAISY Register (IOMUXC_LPI2C2_SDA_SELECT_INPUT)

DAISY 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																

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved														DAISY	
Reset																

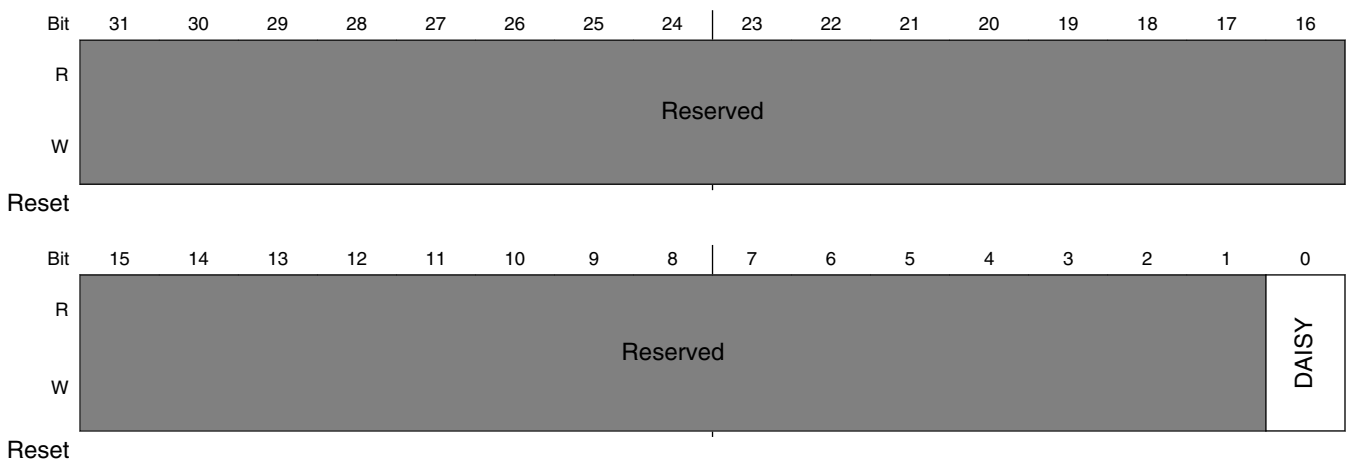
IOMUXC_LPI2C2_SDA_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. instance: LPI2C2, In Pin: lpi2c_sda 00 **GPIO_AD_07_ALT0** — Selecting Pad: GPIO_AD_07 for Mode: ALT0 01 **GPIO_AD_01_ALT3** — Selecting Pad: GPIO_AD_01 for Mode: ALT3 10 **GPIO_SD_07_ALT1** — Selecting Pad: GPIO_SD_07 for Mode: ALT1 11 **GPIO_09_ALT3** — Selecting Pad: GPIO_09 for Mode: ALT3

11.7.113 LPSPI1_PCS_SELECT_INPUT_0 DAISY Register (IOMUXC_LPSPI1_PCS_SELECT_INPUT_0)

DAISY Register

Address: 401F_8000h base + 1D0h offset = 401F_81D0h



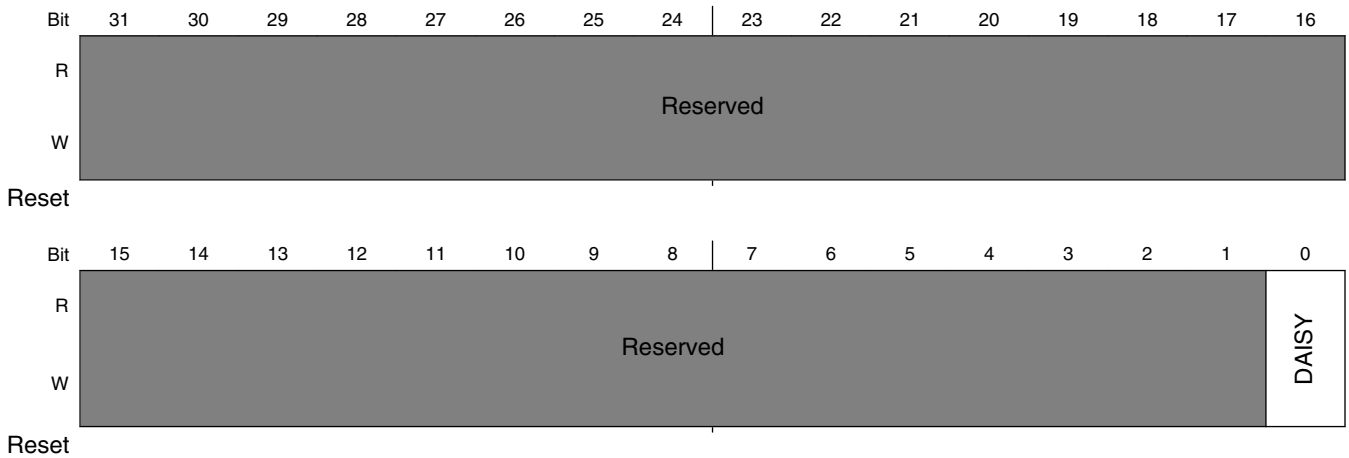
IOMUXC_LPSPI1_PCS_SELECT_INPUT_0 field descriptions

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_AD_05_ALT0** — Selecting Pad: GPIO_AD_05 for Mode: ALT0 1 **GPIO_SD_07_ALT2** — Selecting Pad: GPIO_SD_07 for Mode: ALT2

11.7.114 LPSPI1_SCK_SELECT_INPUT DAISY Register (IOMUXC_LPSPI1_SCK_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 1D4h offset = 401F_81D4h



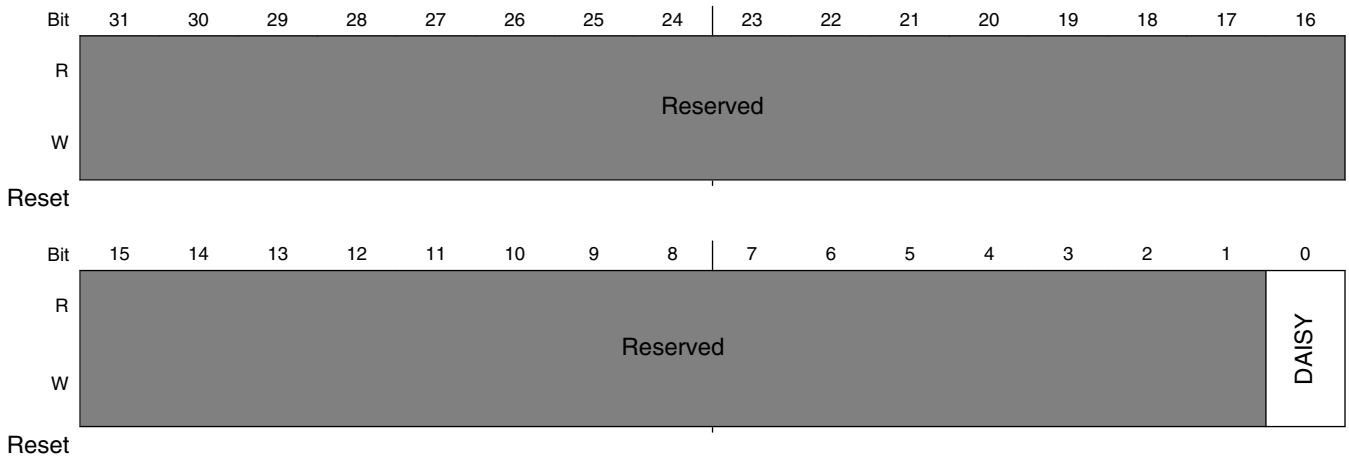
IOMUXC_LPSPI1_SCK_SELECT_INPUT field descriptions

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_AD_06_ALT0** — Selecting Pad: GPIO_AD_06 for Mode: ALT0 1 **GPIO_SD_08_ALT2** — Selecting Pad: GPIO_SD_08 for Mode: ALT2

11.7.115 LPSPI1_SDI_SELECT_INPUT DAISY Register (IOMUXC_LPSPI1_SDI_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 1D8h offset = 401F_81D8h



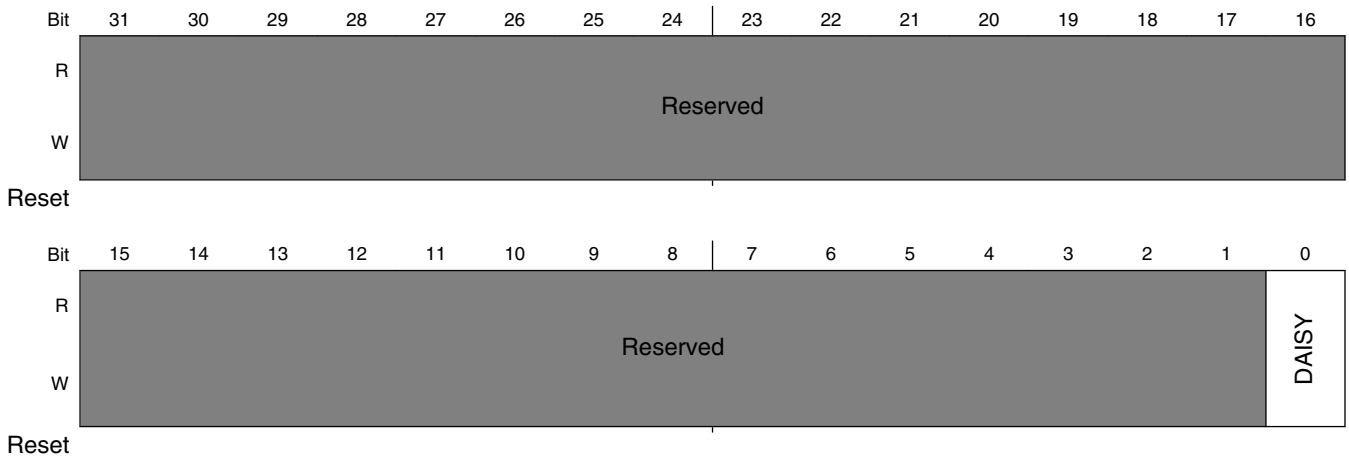
IOMUXC_LPSPI1_SDI_SELECT_INPUT field descriptions

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_AD_03_ALT0** — Selecting Pad: GPIO_AD_03 for Mode: ALT0 1 **GPIO_SD_05_ALT2** — Selecting Pad: GPIO_SD_05 for Mode: ALT2

11.7.116 LPSPI1_SDO_SELECT_INPUT DAISY Register (IOMUXC_LPSPI1_SDO_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 1DCh offset = 401F_81DCh



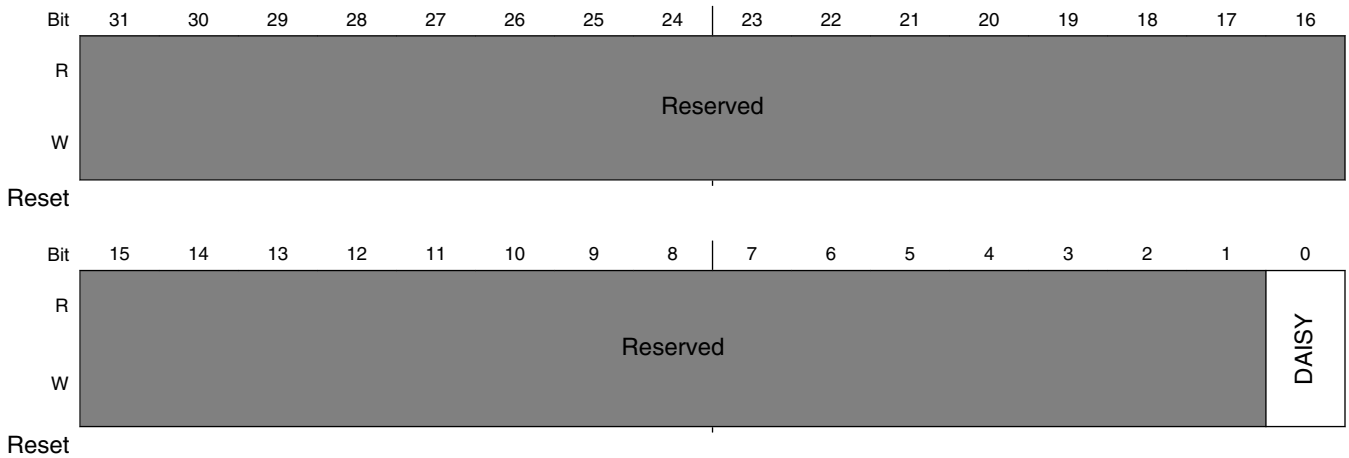
IOMUXC_LPSPI1_SDO_SELECT_INPUT field descriptions

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_AD_04_ALT0** — Selecting Pad: GPIO_AD_04 for Mode: ALT0 1 **GPIO_SD_06_ALT2** — Selecting Pad: GPIO_SD_06 for Mode: ALT2

11.7.117 LPSPI2_PCS_SELECT_INPUT_0 DAISY Register (IOMUXC_LPSPI2_PCS_SELECT_INPUT_0)

DAISY Register

Address: 401F_8000h base + 1E0h offset = 401F_81E0h



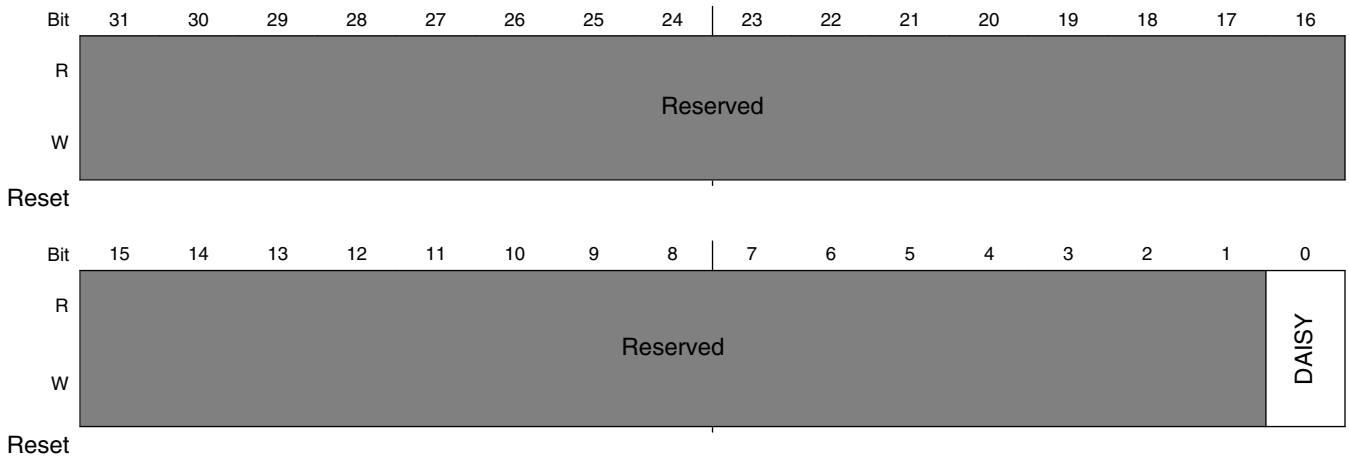
IOMUXC_LPSPI2_PCS_SELECT_INPUT_0 field descriptions

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_AD_11_ALT0** — Selecting Pad: GPIO_AD_11 for Mode: ALT0 1 **GPIO_SD_12_ALT1** — Selecting Pad: GPIO_SD_12 for Mode: ALT1

11.7.118 LPSPI2_SCK_SELECT_INPUT DAISY Register (IOMUXC_LPSPI2_SCK_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 1E4h offset = 401F_81E4h



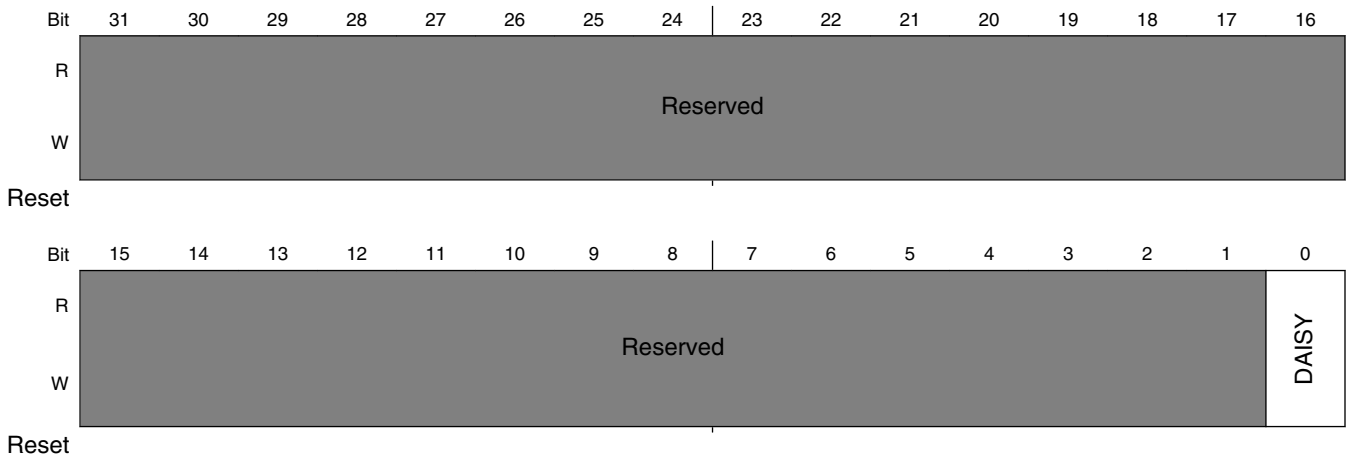
IOMUXC_LPSPI2_SCK_SELECT_INPUT field descriptions

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_AD_12_ALT0** — Selecting Pad: GPIO_AD_12 for Mode: ALT0 1 **GPIO_SD_11_ALT1** — Selecting Pad: GPIO_SD_11 for Mode: ALT1

11.7.119 LPSPI2_SDI_SELECT_INPUT DAISY Register (IOMUXC_LPSPI2_SDI_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 1E8h offset = 401F_81E8h



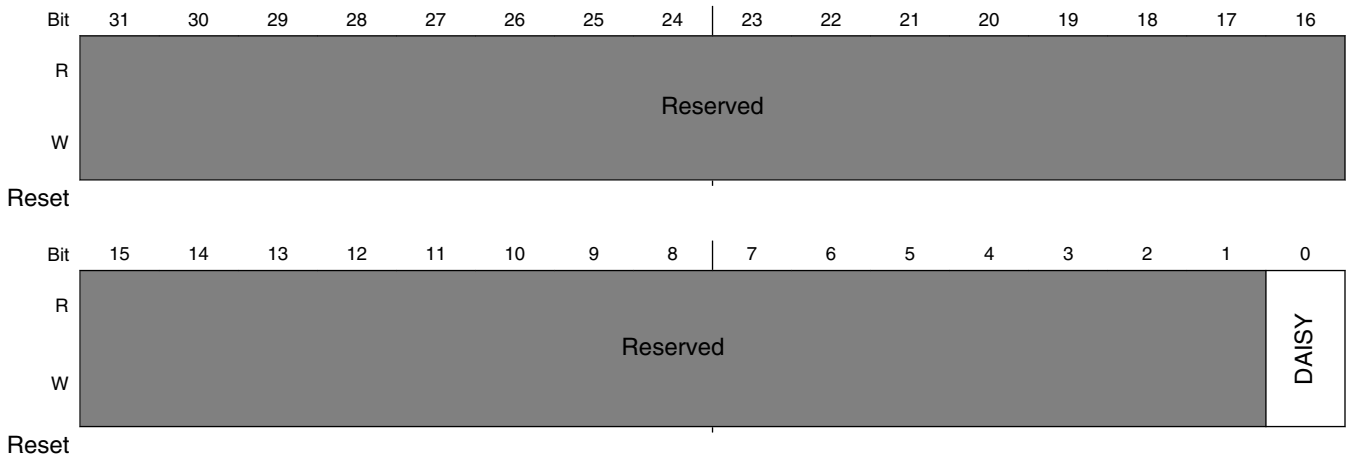
IOMUXC_LPSPI2_SDI_SELECT_INPUT field descriptions

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_AD_09_ALT0** — Selecting Pad: GPIO_AD_09 for Mode: ALT0 1 **GPIO_SD_09_ALT1** — Selecting Pad: GPIO_SD_09 for Mode: ALT1

11.7.120 LPSPI2_SDO_SELECT_INPUT DAISY Register (IOMUXC_LPSPI2_SDO_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 1ECh offset = 401F_81ECh



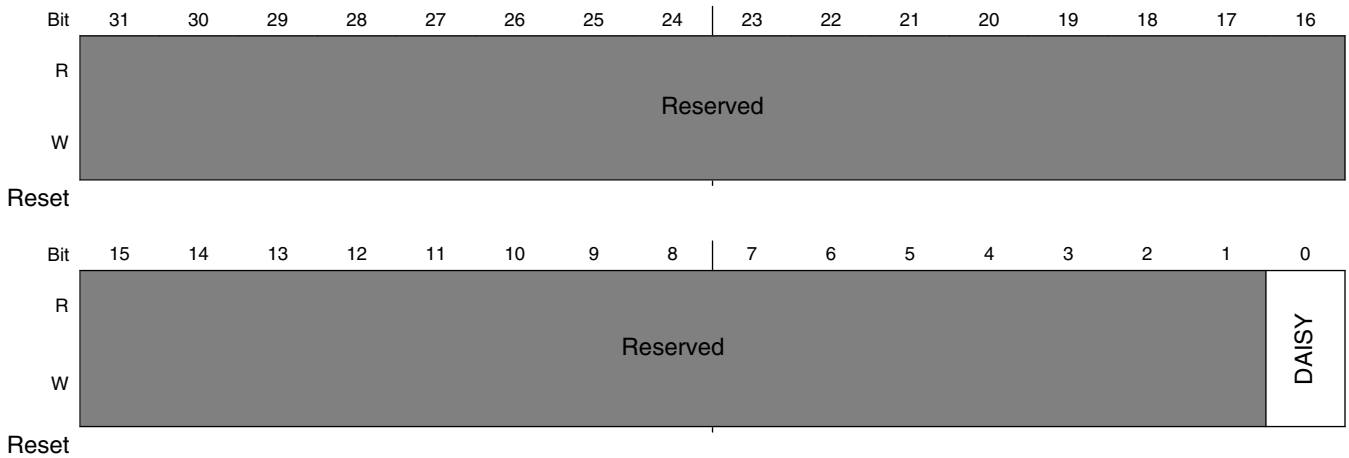
IOMUXC_LPSPI2_SDO_SELECT_INPUT field descriptions

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_AD_10_ALT0** — Selecting Pad: GPIO_AD_10 for Mode: ALT0 1 **GPIO_SD_10_ALT1** — Selecting Pad: GPIO_SD_10 for Mode: ALT1

11.7.121 LPUART1_RXD_SELECT_INPUT DAISY Register (IOMUXC_LPUART1_RXD_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 1F0h offset = 401F_81F0h



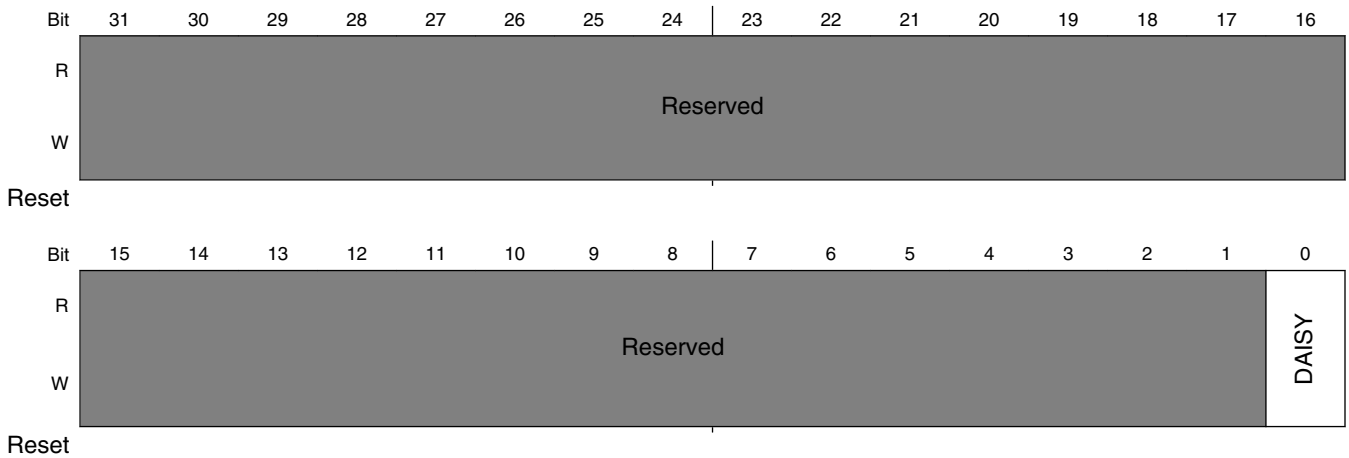
IOMUXC_LPUART1_RXD_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. instance: LPUART1, In Pin: lpuart_rxd 0 **GPIO_SD_11_ALT2** — Selecting Pad: GPIO_SD_11 for Mode: ALT2 1 **GPIO_09_ALT0** — Selecting Pad: GPIO_09 for Mode: ALT0

11.7.122 LPUART1_TXD_SELECT_INPUT DAISY Register (IOMUXC_LPUART1_TXD_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 1F4h offset = 401F_81F4h



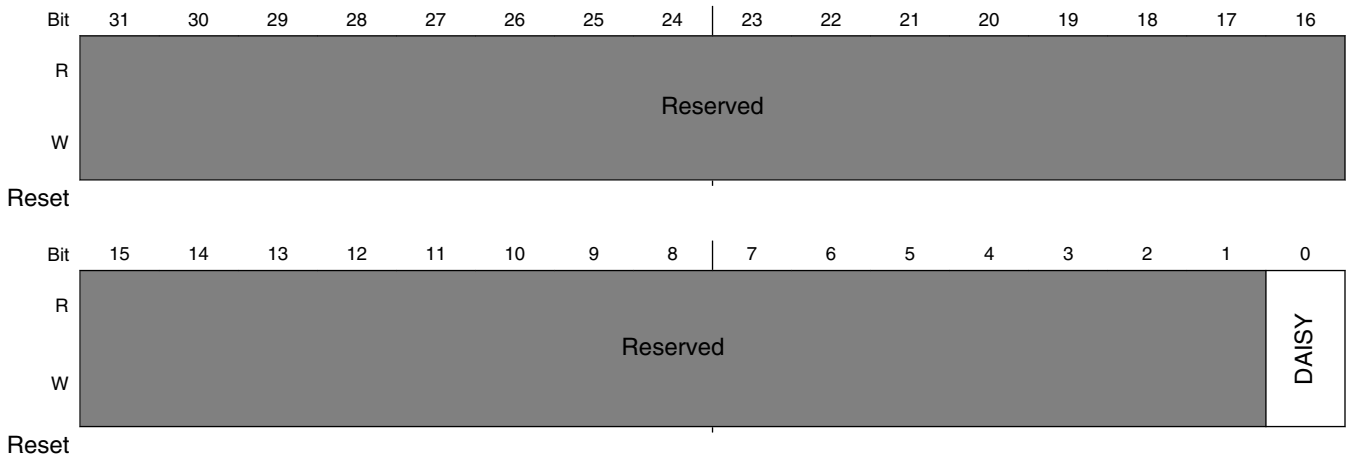
IOMUXC_LPUART1_TXD_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. instance: LPUART1, In Pin: lpuart_txd 0 **GPIO_SD_12_ALT2** — Selecting Pad: GPIO_SD_12 for Mode: ALT2 1 **GPIO_10_ALT0** — Selecting Pad: GPIO_10 for Mode: ALT0

11.7.123 LPUART2_RXD_SELECT_INPUT DAISY Register (IOMUXC_LPUART2_RXD_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 1F8h offset = 401F_81F8h



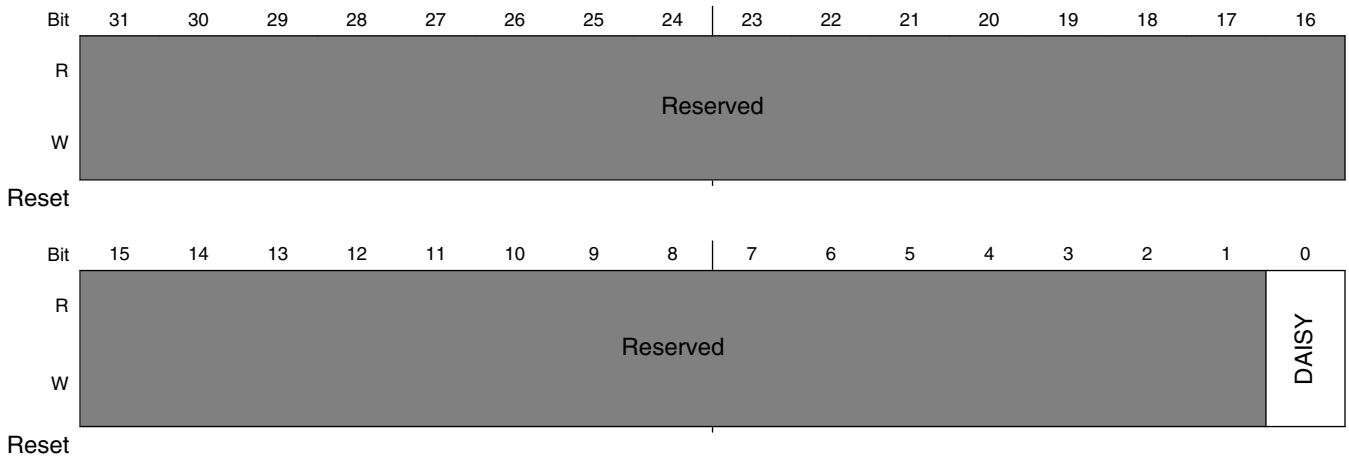
IOMUXC_LPUART2_RXD_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. instance: LPUART2, In Pin: lpuart_rxd 0 **GPIO_SD_09_ALT2** — Selecting Pad: GPIO_SD_09 for Mode: ALT2 1 **GPIO_13_ALT0** — Selecting Pad: GPIO_13 for Mode: ALT0

11.7.124 LPUART2_TXD_SELECT_INPUT DAISY Register (IOMUXC_LPUART2_TXD_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 1FCh offset = 401F_81FCh



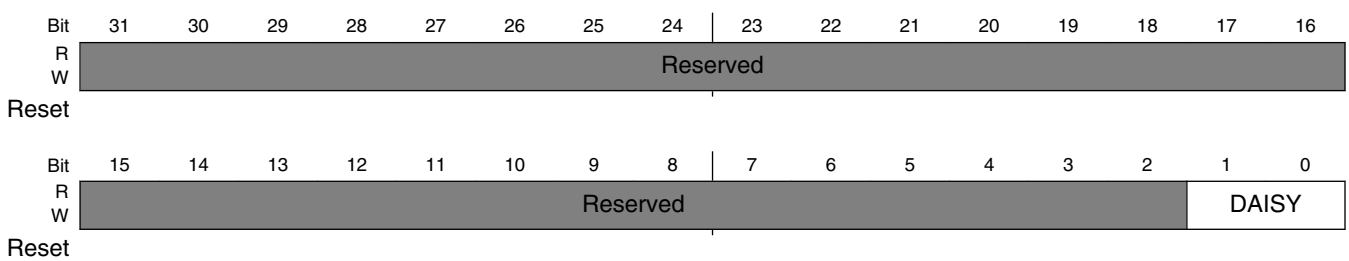
IOMUXC_LPUART2_TXD_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. instance: LPUART2, In Pin: lpuart_txd 0 **GPIO_AD_00_ALT0** — Selecting Pad: GPIO_AD_00 for Mode: ALT0 1 **GPIO_SD_10_ALT2** — Selecting Pad: GPIO_SD_10 for Mode: ALT2

11.7.125 LPUART3_RXD_SELECT_INPUT DAISY Register (IOMUXC_LPUART3_RXD_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 200h offset = 401F_8200h



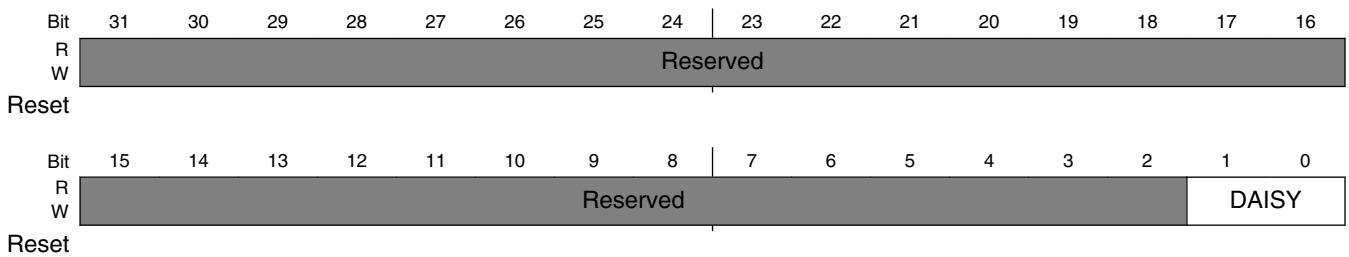
IOMUXC_LPUART3_RXD_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. instance: LPUART3, In Pin: lpuart_rxd 00 **GPIO_AD_07_ALT1** — Selecting Pad: GPIO_AD_07 for Mode: ALT1 01 **GPIO_11_ALT0** — Selecting Pad: GPIO_11 for Mode: ALT0 10 **GPIO_07_ALT3** — Selecting Pad: GPIO_07 for Mode: ALT3

11.7.126 LPUART3_TXD_SELECT_INPUT DAISY Register (IOMUXC_LPUART3_TXD_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 204h offset = 401F_8204h



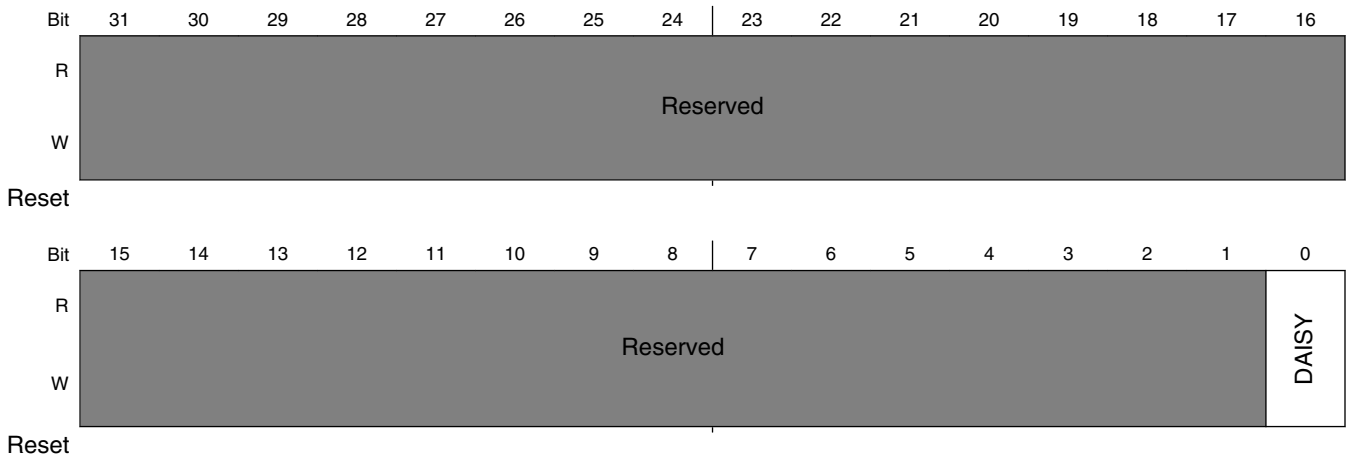
IOMUXC_LPUART3_TXD_SELECT_INPUT field descriptions

Field	Description
31–2 -	This field is reserved. Reserved
DAISY	Selecting Pads Involved in Daisy Chain. instance: LPUART3, In Pin: lpuart_txd 00 **GPIO_AD_08_ALT1** — Selecting Pad: GPIO_AD_08 for Mode: ALT1 01 **GPIO_12_ALT0** — Selecting Pad: GPIO_12 for Mode: ALT0 10 **GPIO_08_ALT3** — Selecting Pad: GPIO_08 for Mode: ALT3

11.7.127 LPUART4_RXD_SELECT_INPUT DAISY Register (IOMUXC_LPUART4_RXD_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 208h offset = 401F_8208h



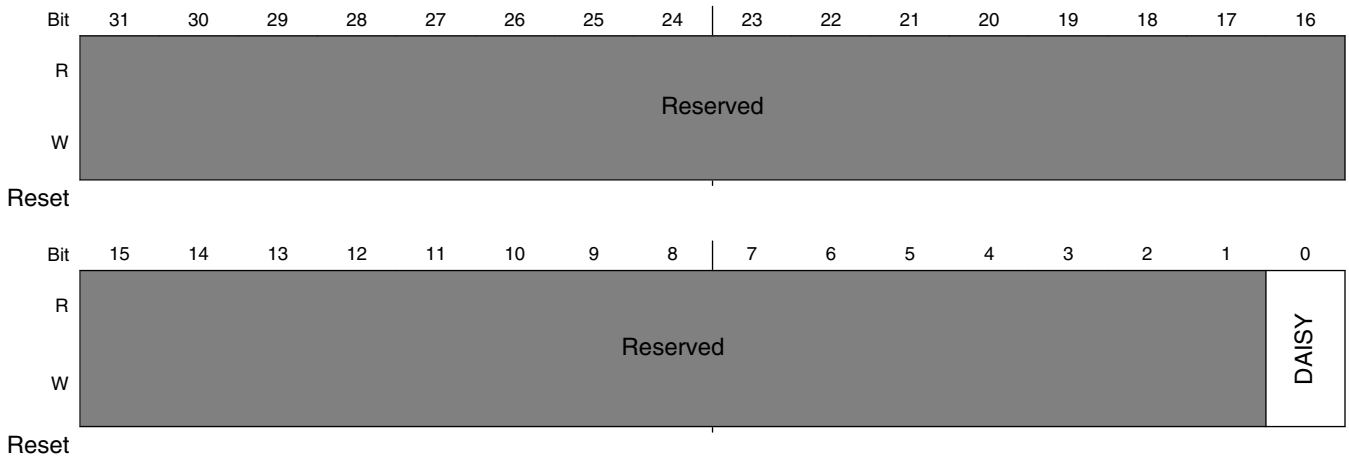
IOMUXC_LPUART4_RXD_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. instance: LPUART4, In Pin: lpuart_rxd 0 **GPIO_AD_01_ALT0** — Selecting Pad: GPIO_AD_01 for Mode: ALT0 1 **GPIO_05_ALT3** — Selecting Pad: GPIO_05 for Mode: ALT3

11.7.128 LPUART4_TXD_SELECT_INPUT DAISY Register (IOMUXC_LPUART4_TXD_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 20Ch offset = 401F_820Ch



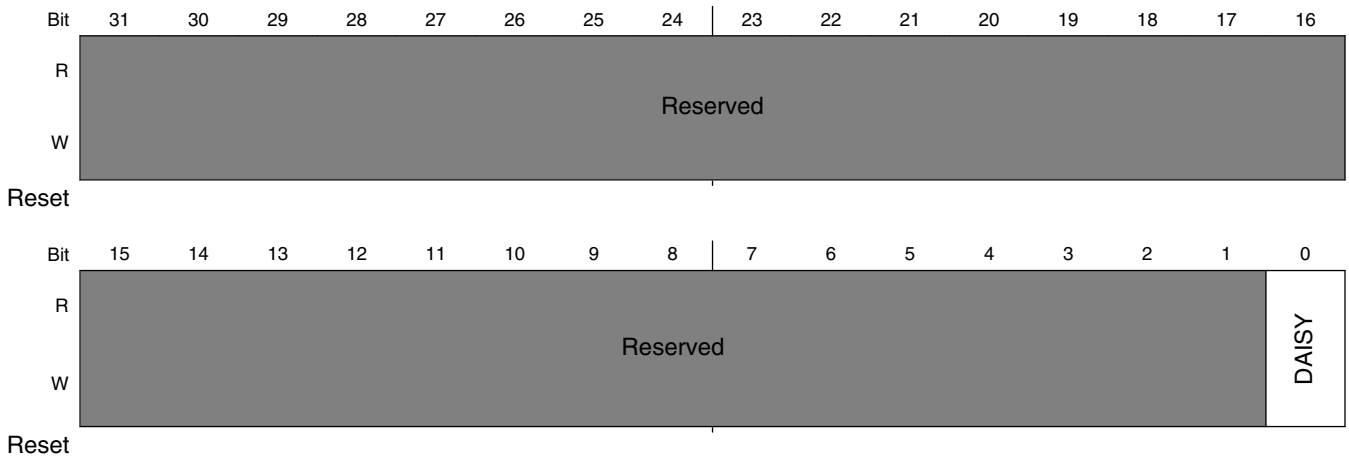
IOMUXC_LPUART4_TXD_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. instance: LPUART4, In Pin: lpuart_txd 0 **GPIO_AD_02_ALT0** — Selecting Pad: GPIO_AD_02 for Mode: ALT0 1 **GPIO_06_ALT3** — Selecting Pad: GPIO_06 for Mode: ALT3

11.7.129 NMI_GLUE_NMI_SELECT_INPUT DAISY Register (IOMUXC_NMI_GLUE_NMI_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 210h offset = 401F_8210h



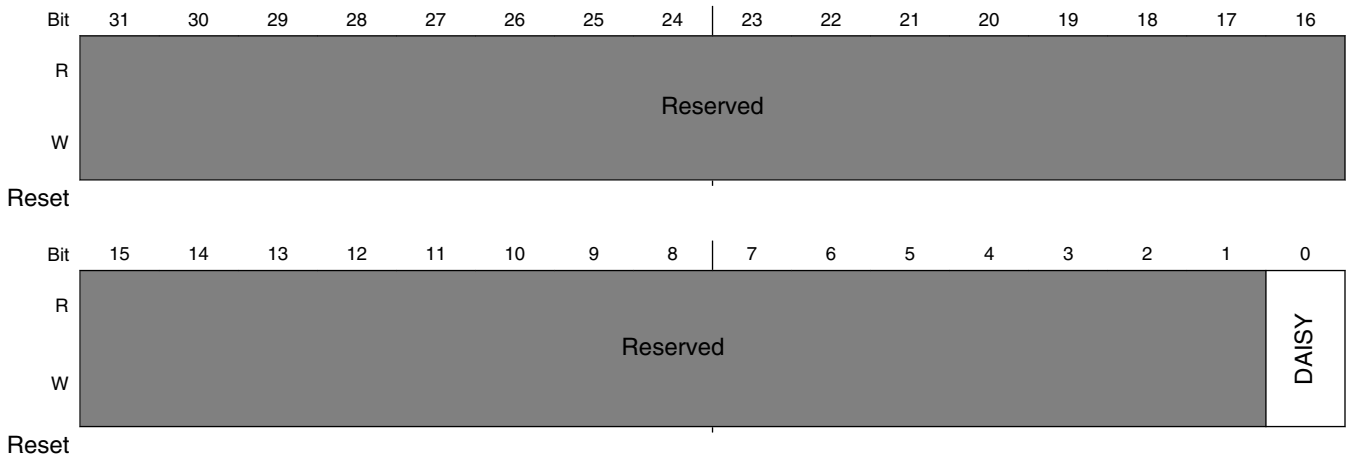
IOMUXC_NMI_GLUE_NMI_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. instance: NMI_GLUE, In Pin: nmi 0 **GPIO_AD_13_ALT6** — Selecting Pad: GPIO_AD_13 for Mode: ALT6 1 **GPIO_AD_00_ALT6** — Selecting Pad: GPIO_AD_00 for Mode: ALT6

11.7.130 SPDIF_IN1_SELECT_INPUT DAISY Register (IOMUXC_SPDIF_IN1_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 214h offset = 401F_8214h



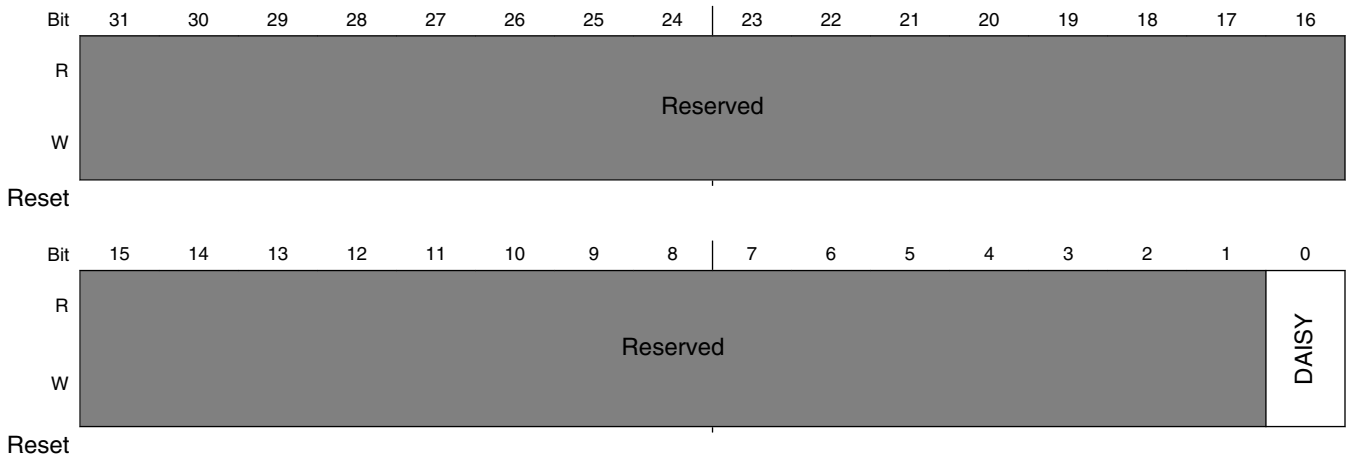
IOMUXC_SPDIF_IN1_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: spdif, In Pin: spdif_in1 0 **GPIO_10_ALT6** — Selecting Pad: GPIO_10 for Mode: ALT6 1 **GPIO_04_ALT4** — Selecting Pad: GPIO_04 for Mode: ALT4

11.7.131 SPDIF_TX_CLK2_SELECT_INPUT DAISY Register (IOMUXC_SPDIF_TX_CLK2_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 218h offset = 401F_8218h



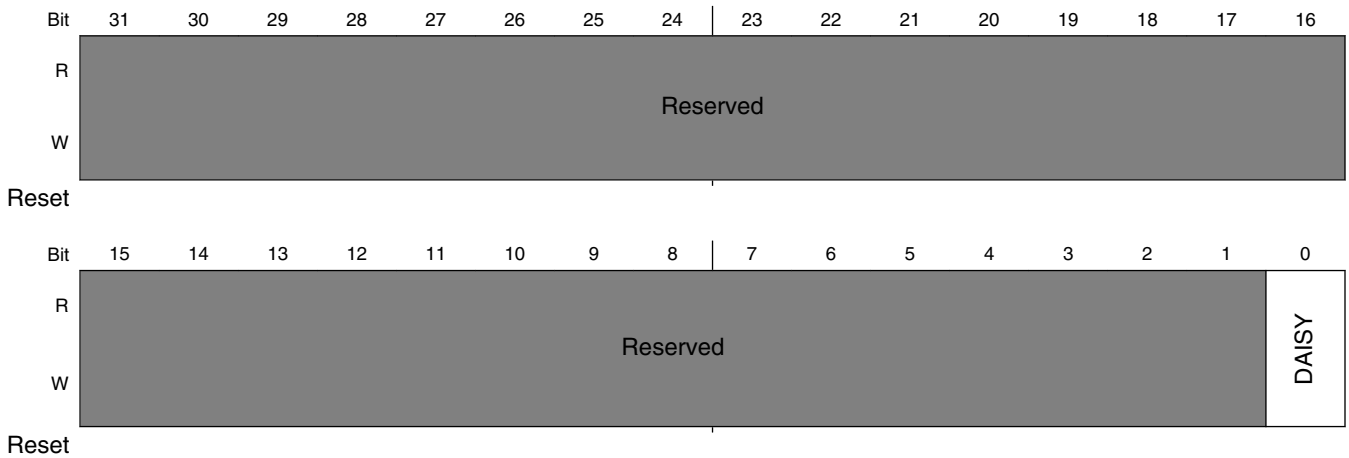
IOMUXC_SPDIF_TX_CLK2_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: spdif, In Pin: tx_clk2 0 **GPIO_12_ALT6** — Selecting Pad: GPIO_12 for Mode: ALT6 1 **GPIO_06_ALT4** — Selecting Pad: GPIO_06 for Mode: ALT4

11.7.132 USB_OTG_OC_SELECT_INPUT DAISY Register (IOMUXC_USB_OTG_OC_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 21Ch offset = 401F_821Ch



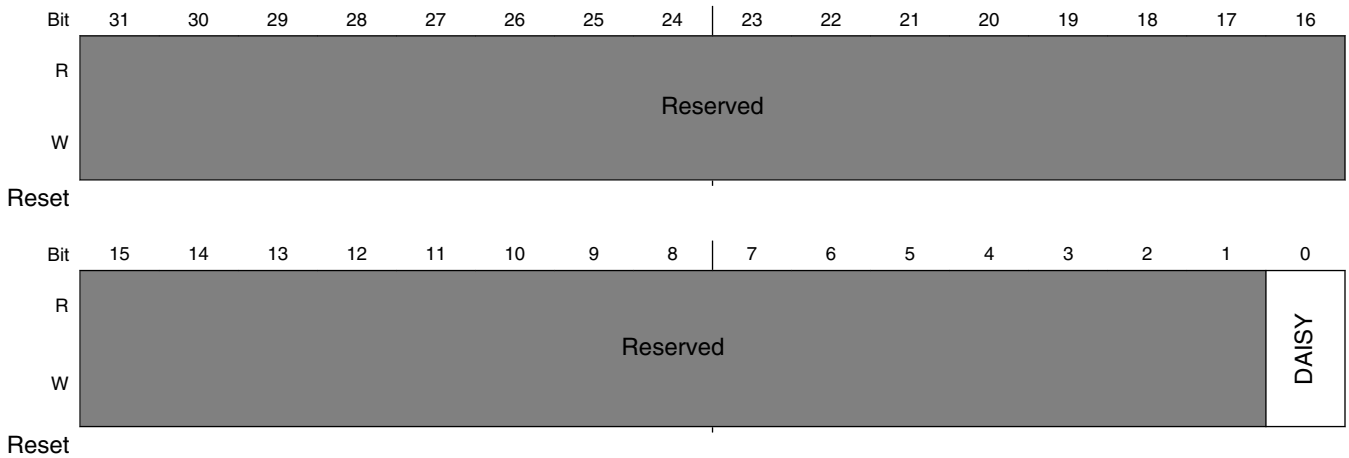
IOMUXC_USB_OTG_OC_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. instance: USB, In Pin: otg_oc 0 **GPIO_AD_01_ALT6** — Selecting Pad: GPIO_AD_01 for Mode: ALT6 1 **GPIO_12_ALT3** — Selecting Pad: GPIO_12 for Mode: ALT3

11.7.133 XEV_GLUE_RXEV_SELECT_INPUT DAISY Register (IOMUXC_XEV_GLUE_RXEV_SELECT_INPUT)

DAISY Register

Address: 401F_8000h base + 220h offset = 401F_8220h



IOMUXC_XEV_GLUE_RXEV_SELECT_INPUT field descriptions

Field	Description
31–1 -	This field is reserved. Reserved
0 DAISY	Selecting Pads Involved in Daisy Chain. Instance: xev_glue, In Pin: rxev 0 **GPIO_AD_07_ALT2** — Selecting Pad: GPIO_AD_07 for Mode: ALT2 1 **GPIO_SD_00_ALT2** — Selecting Pad: GPIO_SD_00 for Mode: ALT2

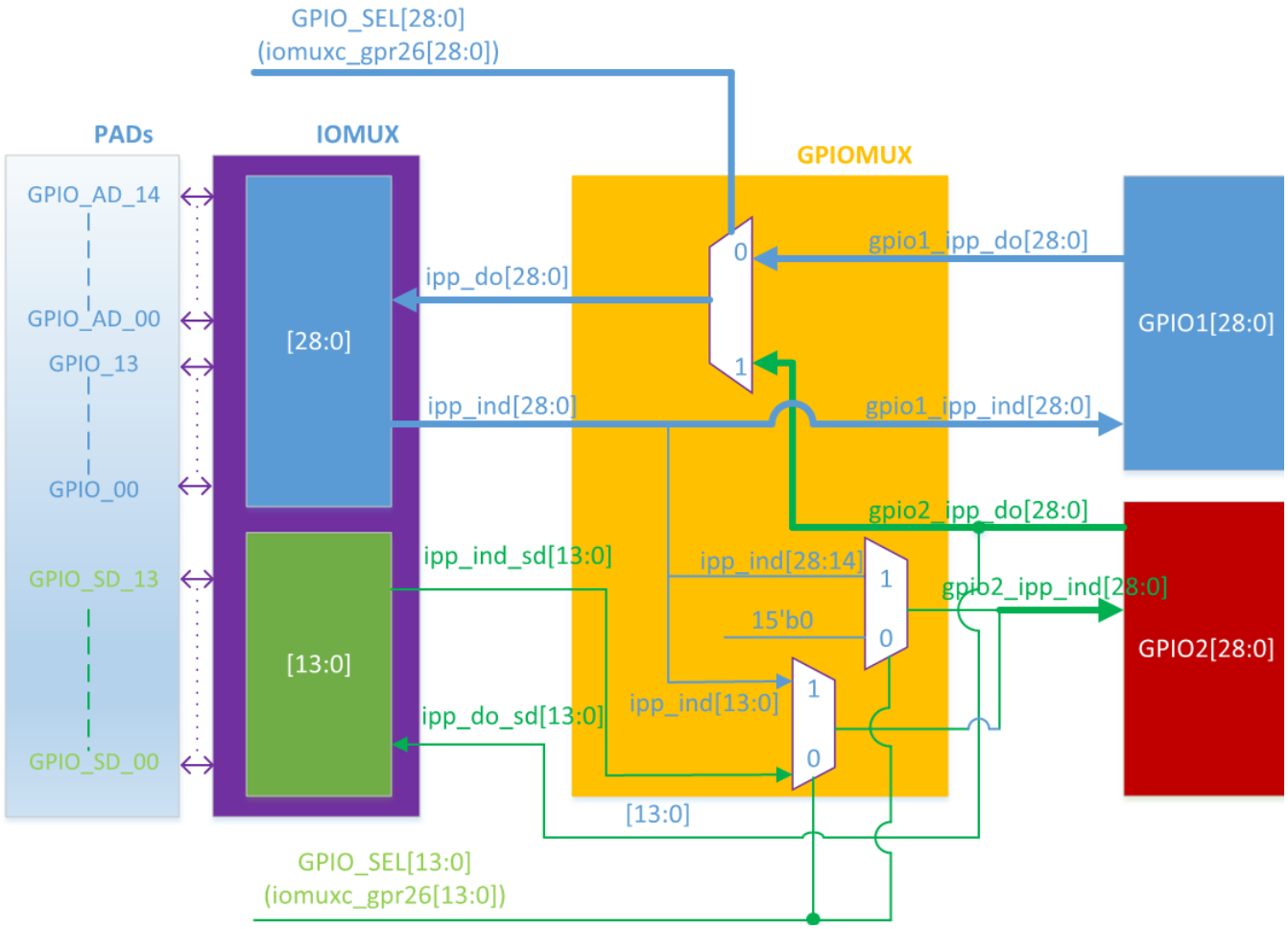
Chapter 12
General Purpose Input/Output (GPIO)

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

The GPIOMUX diagram is as follows:



NOTE

GPIO2 is the on-platform high-speed GPIO, under CM7 power domain. GPIO2 cannot act as wakeup source.

12.2 Overview

The GPIO general-purpose input/output 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.

The GPIO is one of the blocks controlling the IOMUX of the chip.

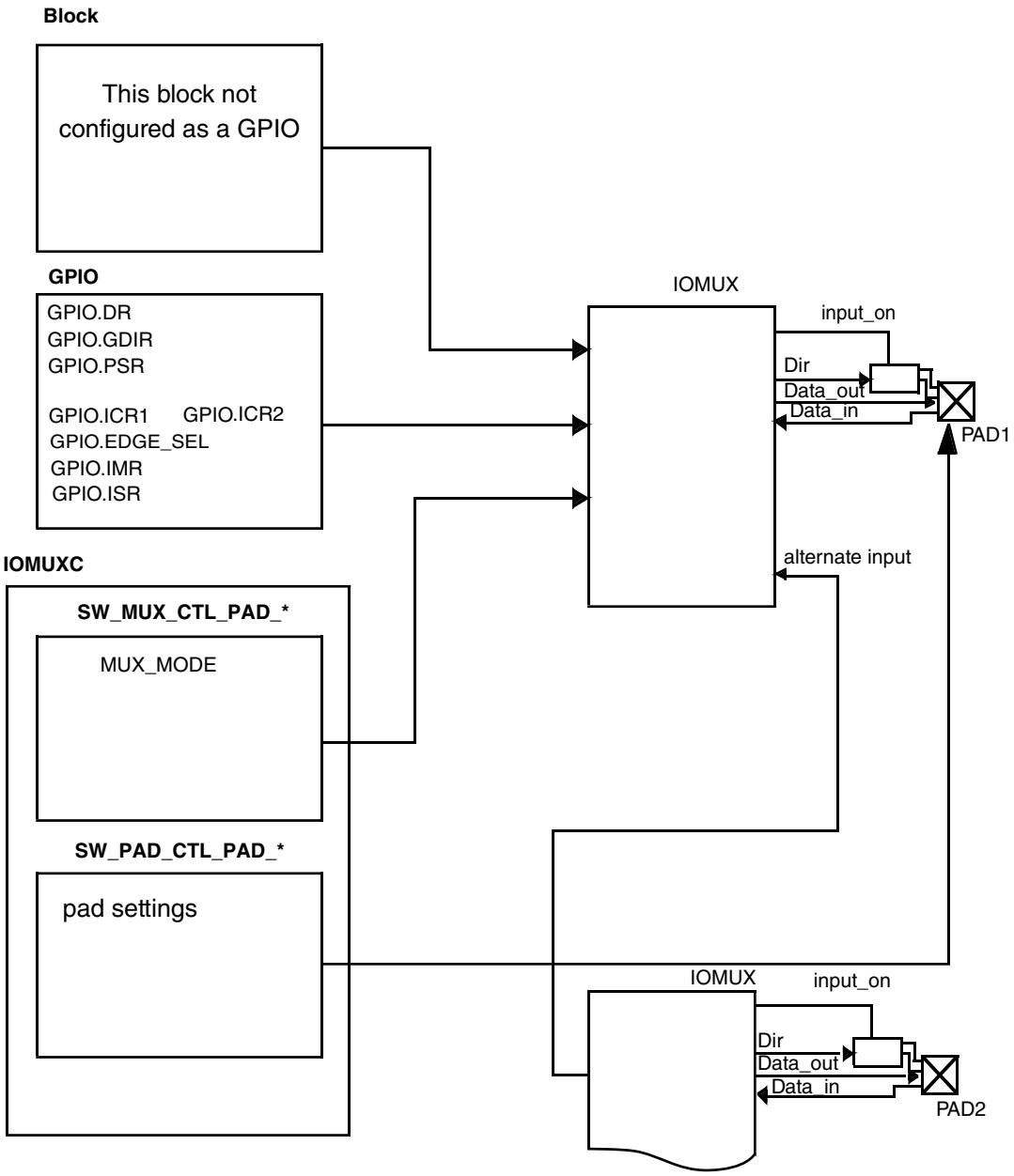


Figure 12-1. Chip IOMUX Scheme

The GPIO functionality is provided through eight registers, an edge-detect circuit, and interrupt generation logic.

The eight registers are:

- Data register (GPIO_DR)
- GPIO direction register (GPIO_GDIR)
- Pad sample register (GPIO_PSR)
- Interrupt control registers (GPIO_ICR1, GPIO_ICR2)

- Edge select register (GPIO_EDGE_SEL)
- Interrupt mask register (GPIO_IMR)
- Interrupt status register (GPIO_ISR)

These 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 (GPIO_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.

The GPIO's overall functionality is described further in GPIO Functional Description.

12.2.1 Block Diagram

The GPIO subsystem contains multiple GPIO blocks, which can generate and control up to 32 signals for general purpose.

A block diagram of the GPIO is shown in Figure 12-2

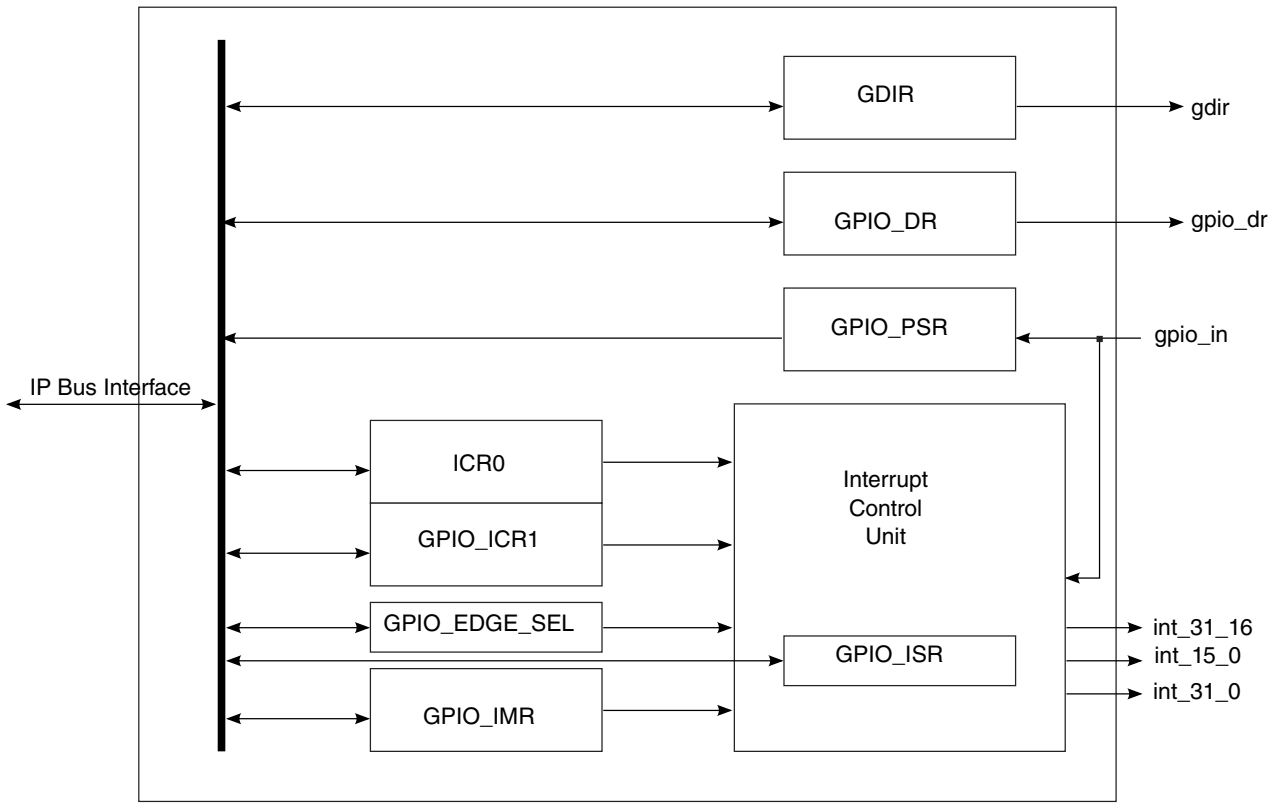


Figure 12-2. GPIO Block Diagram

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 (GPIO_DR)
  - Controls the direction of the signal using the GPIO direction register (GPIO_GDIR)
  - Enables the core to sample the status of the corresponding inputs by reading the pad sample register (GPIO_PSR).
- GPIO interrupt capabilities:
  - Supports up to 32 interrupts
  - Identifies interrupt edges
  - Generates three active-high interrupts to the SoC interrupt controller

12.3 Clocks

The table found here describes the clock sources for GPIO.

Please see the CCM for clock setting, configuration and gating information.

Table 12-2. GPIO Clocks

Clock name	Description
ipg_clk_s	Peripheral access clock

12.4 GPIO Functional Description

This section provides a complete functional description of the block.

12.4.1 GPIO Function

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 (GPIO_GDIR).

When configured as an output (GPIO_GDIR bit = 1), the value in the data bit in the GPIO data register (GPIO_DR) is driven on the corresponding GPIO line. When a signal is configured as an input (GPIO_GDIR bit = 0), the state of the input can be read from the corresponding GPIO_PSR bit.

12.4.2 GPIO pad structure

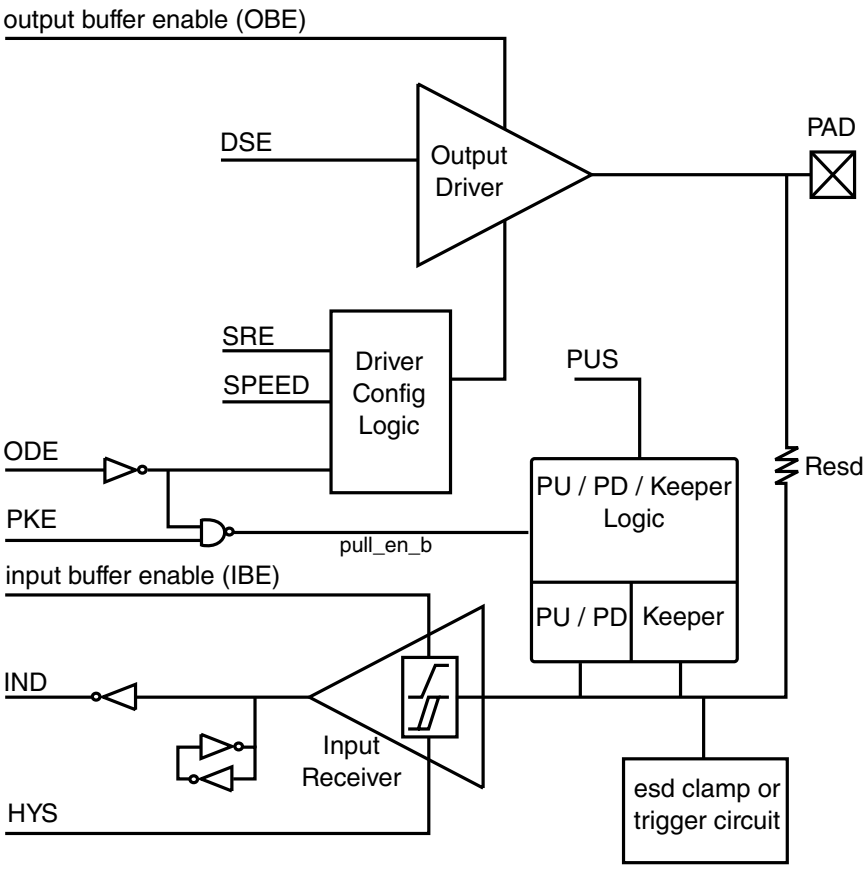


Figure 12-3. GPIO pad functional diagram

12.4.2.1 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. (Keeper at receiver output keeps its previous state).

12.4.2.1.1 Schmitt trigger

The anti-jamming functionality of the Schmitt trigger is illustrated below.

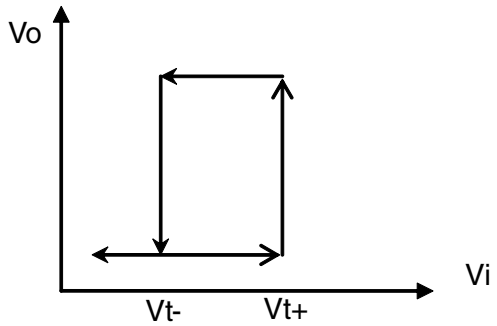


Figure 12-4. Schmitt trigger transfer characteristic

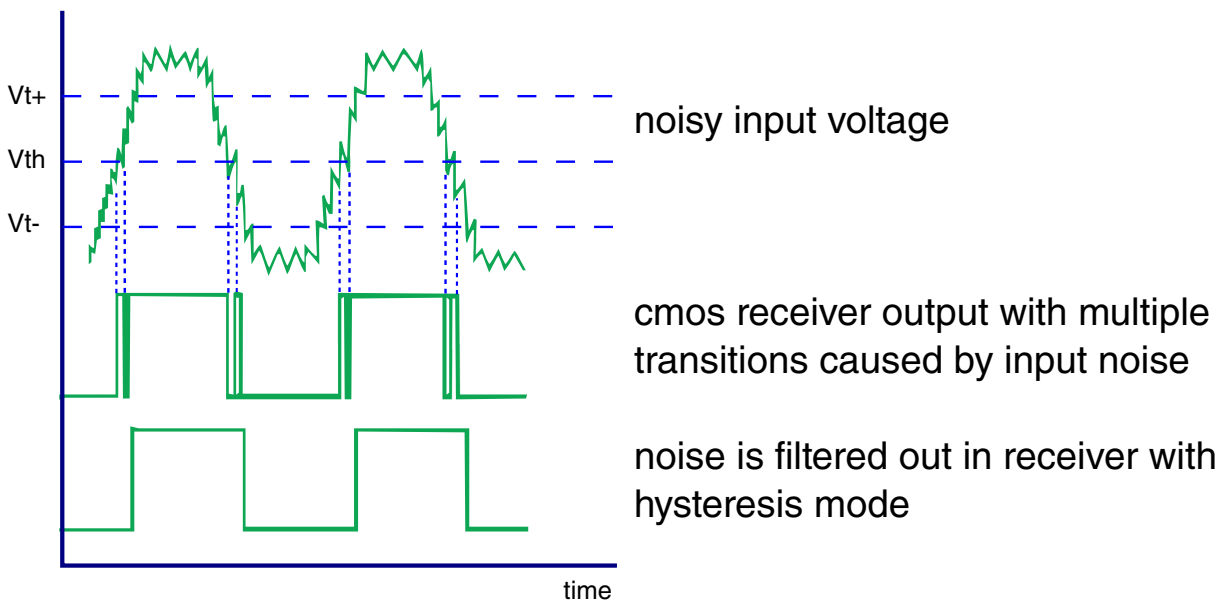


Figure 12-5. Receiver output in CMOS and hysteresis

12.4.2.1.2 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.

12.4.2.2 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

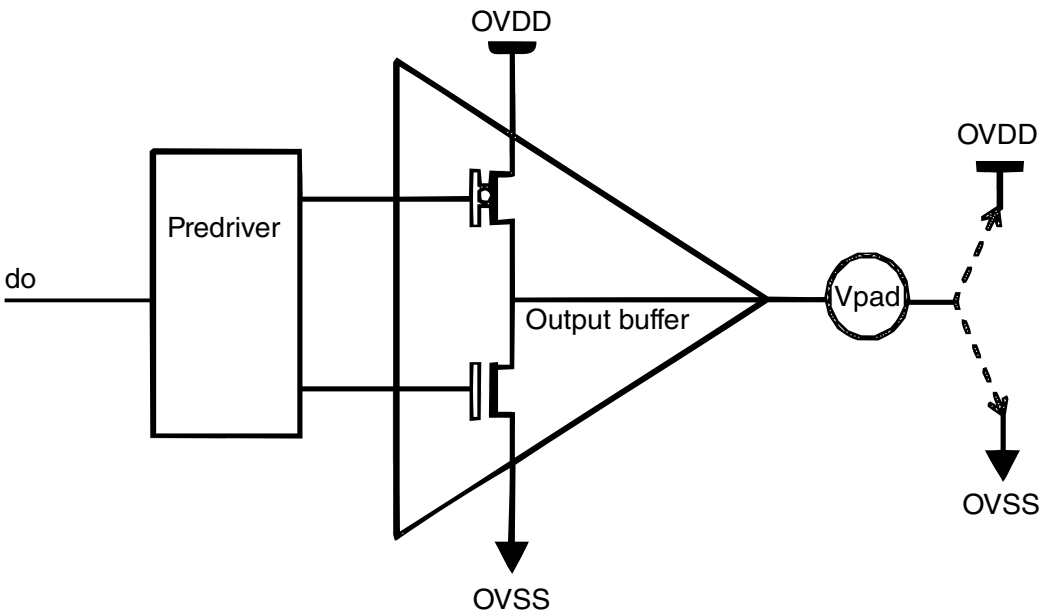


Figure 12-6. Output Driver Functional Diagram

12.4.2.2.1 Drive strength

Drive strength selection can be use to make the impedance matched and get better signal integrity.

12.4.2.2.2 Output keeper

A simple latch to hold the input value.

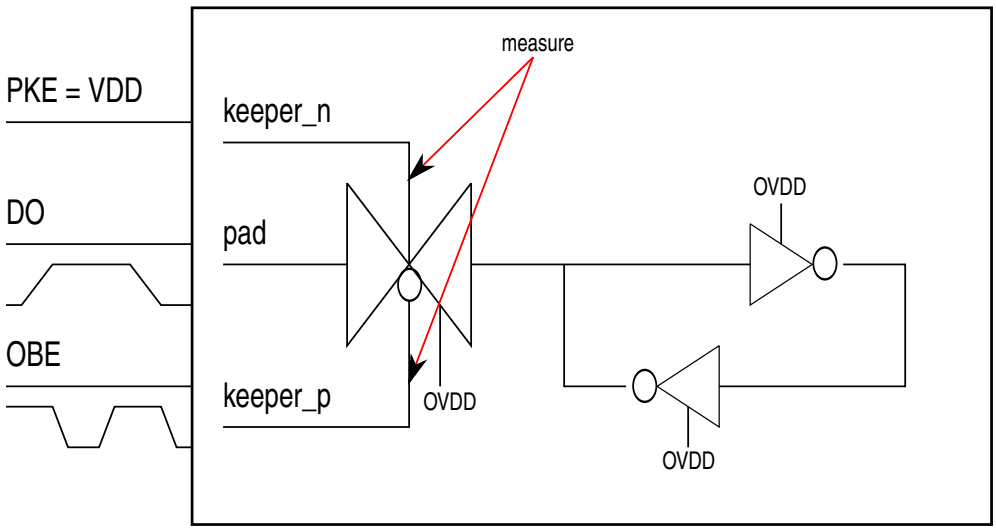


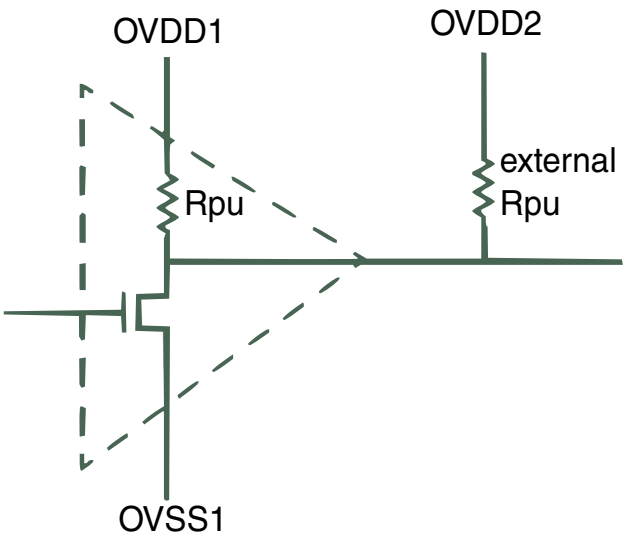
Figure 12-7. Keeper functional diagram

12.4.2.2.3 PU / 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.

12.4.2.2.4 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

12.4.2.3 Operating Frequency

Table 12-3. 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 12-3. 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 12-3. 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

12.4.3 GPIO Programming

12.4.3.1 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 (GPIO_GDIR[GDIR] set to 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 (GPIO_GDIR = 0), a read access to GPIO_DR does not return GPIO_DR data. Instead, it returns the GPIO_PSR data, which is the corresponding input signal value.

12.4.3.2 GPIO Write Mode

The programming sequence for driving output signals should be as follows:

1. Configure IOMUX to select GPIO mode (Via IOMUXC), also enable SION if need to read loopback pad value through PSR
2. Configure GPIO direction register to output (GPIO_GDIR[GDIR] set to 1b).
3. Write value to data register (GPIO_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 so we can 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
```

12.4.4 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 bit = 0). The interrupt control registers (GPIO_ICR1 and GPIO_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 GPIO_ICR1 and GPIO_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.

12.5 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.

12.5.1 GPIO Memory map

GPIO1 base address: 401B_8000h.

GPIO2 base address: 4200_0000h.

GPIO5 base address: 400C_0000h.

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
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

12.5.2 GPIO data register (DR)

12.5.2.1 Offset

Register	Offset
DR	0h

12.5.2.2 Function

The 32-bit GPIO_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 GPIO_DR 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.

12.5.2.3 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

12.5.2.4 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 GPIO_DR 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 I/O multiplexer must be configured to GPIO mode for the GPIO_DR value to connect with the signal. Reading the data register with the input path disabled always returns a zero value.

12.5.3 GPIO direction register (GDIR)

12.5.3.1 Offset

Register	Offset
GDIR	4h

12.5.3.2 Function

GPIO_GDIR 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.

12.5.3.3 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

12.5.3.4 Fields

Field	Function
31-0 GDIR	GDIR GPIO direction bits. Bit n of this register defines the direction of the GPIO[n] signal. **NOTE:** GPIO_GDIR affects only the direction of the I/O signal when the corresponding bit in the I/O MUX is configured for GPIO. 0b - GPIO is configured as input. 1b - GPIO is configured as output.

12.5.4 GPIO pad status register (PSR)

12.5.4.1 Offset

Register	Offset
PSR	8h

12.5.4.2 Function

GPIO_PSR 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.

12.5.4.3 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

12.5.4.4 Fields

Field	Function
31-0 PSR	PSR GPIO pad status bits (status bits). Reading GPIO_PSR returns the state of the corresponding input signal. Settings: **NOTE:** The IOMUXC must be configured to GPIO mode for GPIO_PSR to reflect the state of the corresponding signal.

12.5.5 GPIO interrupt configuration register1 (ICR1)

12.5.5.1 Offset

Register	Offset
ICR1	Ch

12.5.5.2 Function

GPIO_ICR1 contains 16 two-bit fields, where each field specifies the interrupt configuration for a different input signal.

12.5.5.3 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

12.5.5.4 Fields

Field	Function
31-30 ICR15	ICR15 Interrupt configuration 1 fields. This register controls the active condition of the interrupt function for GPIO interrupt 15. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00b - Interrupt n is low-level sensitive. 01b - Interrupt n is high-level sensitive. 10b - Interrupt n is rising-edge sensitive. 11b - Interrupt n is falling-edge sensitive.
29-28 ICR14	ICR14 Interrupt configuration 1 fields. This register controls the active condition of the interrupt function for GPIO interrupt 14.

Table continues on the next page...

Field	Function
	Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00b - Interrupt n is low-level sensitive. 01b - Interrupt n is high-level sensitive. 10b - Interrupt n is rising-edge sensitive. 11b - Interrupt n is falling-edge sensitive.
27-26 ICR13	ICR13 Interrupt configuration 1 fields. This register controls the active condition of the interrupt function for GPIO interrupt 13. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00b - Interrupt n is low-level sensitive. 01b - Interrupt n is high-level sensitive. 10b - Interrupt n is rising-edge sensitive. 11b - Interrupt n is falling-edge sensitive.
25-24 ICR12	ICR12 Interrupt configuration 1 fields. This register controls the active condition of the interrupt function for GPIO interrupt 12. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00b - Interrupt n is low-level sensitive. 01b - Interrupt n is high-level sensitive. 10b - Interrupt n is rising-edge sensitive. 11b - Interrupt n is falling-edge sensitive.
23-22 ICR11	ICR11 Interrupt configuration 1 fields. This register controls the active condition of the interrupt function for GPIO interrupt 11. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00b - Interrupt n is low-level sensitive. 01b - Interrupt n is high-level sensitive. 10b - Interrupt n is rising-edge sensitive. 11b - Interrupt n is falling-edge sensitive.
21-20 ICR10	ICR10 Interrupt configuration 1 fields. This register controls the active condition of the interrupt function for GPIO interrupt 10. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00b - Interrupt n is low-level sensitive. 01b - Interrupt n is high-level sensitive. 10b - Interrupt n is rising-edge sensitive. 11b - Interrupt n is falling-edge sensitive.
19-18 ICR9	ICR9 Interrupt configuration 1 fields. This register controls the active condition of the interrupt function for GPIO interrupt 9.

Table continues on the next page...

Field	Function
	Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00b - Interrupt n is low-level sensitive. 01b - Interrupt n is high-level sensitive. 10b - Interrupt n is rising-edge sensitive. 11b - Interrupt n is falling-edge sensitive.
17-16 ICR8	ICR8 Interrupt configuration 1 fields. This register controls the active condition of the interrupt function for GPIO interrupt 8. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00b - Interrupt n is low-level sensitive. 01b - Interrupt n is high-level sensitive. 10b - Interrupt n is rising-edge sensitive. 11b - Interrupt n is falling-edge sensitive.
15-14 ICR7	ICR7 Interrupt configuration 1 fields. This register controls the active condition of the interrupt function for GPIO interrupt 7. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00b - Interrupt n is low-level sensitive. 01b - Interrupt n is high-level sensitive. 10b - Interrupt n is rising-edge sensitive. 11b - Interrupt n is falling-edge sensitive.
13-12 ICR6	ICR6 Interrupt configuration 1 fields. This register controls the active condition of the interrupt function for GPIO interrupt 6. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00b - Interrupt n is low-level sensitive. 01b - Interrupt n is high-level sensitive. 10b - Interrupt n is rising-edge sensitive. 11b - Interrupt n is falling-edge sensitive.
11-10 ICR5	ICR5 Interrupt configuration 1 fields. This register controls the active condition of the interrupt function for GPIO interrupt 5. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00b - Interrupt n is low-level sensitive. 01b - Interrupt n is high-level sensitive. 10b - Interrupt n is rising-edge sensitive. 11b - Interrupt n is falling-edge sensitive.
9-8 ICR4	ICR4 Interrupt configuration 1 fields. This register controls the active condition of the interrupt function for GPIO interrupt 4.

Table continues on the next page...

Field	Function
	Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00b - Interrupt n is low-level sensitive. 01b - Interrupt n is high-level sensitive. 10b - Interrupt n is rising-edge sensitive. 11b - Interrupt n is falling-edge sensitive.
7-6 ICR3	ICR3 Interrupt configuration 1 fields. This register controls the active condition of the interrupt function for GPIO interrupt 3. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00b - Interrupt n is low-level sensitive. 01b - Interrupt n is high-level sensitive. 10b - Interrupt n is rising-edge sensitive. 11b - Interrupt n is falling-edge sensitive.
5-4 ICR2	ICR2 Interrupt configuration 1 fields. This register controls the active condition of the interrupt function for GPIO interrupt 2. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00b - Interrupt n is low-level sensitive. 01b - Interrupt n is high-level sensitive. 10b - Interrupt n is rising-edge sensitive. 11b - Interrupt n is falling-edge sensitive.
3-2 ICR1	ICR1 Interrupt configuration 1 fields. This register controls the active condition of the interrupt function for GPIO interrupt 1. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00b - Interrupt n is low-level sensitive. 01b - Interrupt n is high-level sensitive. 10b - Interrupt n is rising-edge sensitive. 11b - Interrupt n is falling-edge sensitive.
1-0 ICR0	ICR0 Interrupt configuration 1 fields. This register controls the active condition of the interrupt function for GPIO interrupt 0. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00b - Interrupt n is low-level sensitive. 01b - Interrupt n is high-level sensitive. 10b - Interrupt n is rising-edge sensitive. 11b - Interrupt n is falling-edge sensitive.

12.5.6 GPIO interrupt configuration register2 (ICR2)

12.5.6.1 Offset

Register	Offset
ICR2	10h

12.5.6.2 Function

GPIO_ICR2 contains 16 two-bit fields, where each field specifies the interrupt configuration for a different input signal.

12.5.6.3 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

12.5.6.4 Fields

Field	Function
31-30 ICR31	ICR31 Interrupt configuration 2 fields. This register controls the active condition of the interrupt function for GPIO interrupt 31. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00b - Interrupt n is low-level sensitive. 01b - Interrupt n is high-level sensitive. 10b - Interrupt n is rising-edge sensitive. 11b - Interrupt n is falling-edge sensitive.

Table continues on the next page...

Field	Function
29-28 ICR30	ICR30 Interrupt configuration 2 fields. This register controls the active condition of the interrupt function for GPIO interrupt 30. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00b - Interrupt n is low-level sensitive. 01b - Interrupt n is high-level sensitive. 10b - Interrupt n is rising-edge sensitive. 11b - Interrupt n is falling-edge sensitive.
27-26 ICR29	ICR29 Interrupt configuration 2 fields. This register controls the active condition of the interrupt function for GPIO interrupt 29. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00b - Interrupt n is low-level sensitive. 01b - Interrupt n is high-level sensitive. 10b - Interrupt n is rising-edge sensitive. 11b - Interrupt n is falling-edge sensitive.
25-24 ICR28	ICR28 Interrupt configuration 2 fields. This register controls the active condition of the interrupt function for GPIO interrupt 28. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00b - Interrupt n is low-level sensitive. 01b - Interrupt n is high-level sensitive. 10b - Interrupt n is rising-edge sensitive. 11b - Interrupt n is falling-edge sensitive.
23-22 ICR27	ICR27 Interrupt configuration 2 fields. This register controls the active condition of the interrupt function for GPIO interrupt 27. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00b - Interrupt n is low-level sensitive. 01b - Interrupt n is high-level sensitive. 10b - Interrupt n is rising-edge sensitive. 11b - Interrupt n is falling-edge sensitive.
21-20 ICR26	ICR26 Interrupt configuration 2 fields. This register controls the active condition of the interrupt function for GPIO interrupt 26. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00b - Interrupt n is low-level sensitive. 01b - Interrupt n is high-level sensitive. 10b - Interrupt n is rising-edge sensitive. 11b - Interrupt n is falling-edge sensitive.

Table continues on the next page...

Field	Function
19-18 ICR25	ICR25 Interrupt configuration 2 fields. This register controls the active condition of the interrupt function for GPIO interrupt 25. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00b - Interrupt n is low-level sensitive. 01b - Interrupt n is high-level sensitive. 10b - Interrupt n is rising-edge sensitive. 11b - Interrupt n is falling-edge sensitive.
17-16 ICR24	ICR24 Interrupt configuration 2 fields. This register controls the active condition of the interrupt function for GPIO interrupt 24. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00b - Interrupt n is low-level sensitive. 01b - Interrupt n is high-level sensitive. 10b - Interrupt n is rising-edge sensitive. 11b - Interrupt n is falling-edge sensitive.
15-14 ICR23	ICR23 Interrupt configuration 2 fields. This register controls the active condition of the interrupt function for GPIO interrupt 23. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00b - Interrupt n is low-level sensitive. 01b - Interrupt n is high-level sensitive. 10b - Interrupt n is rising-edge sensitive. 11b - Interrupt n is falling-edge sensitive.
13-12 ICR22	ICR22 Interrupt configuration 2 fields. This register controls the active condition of the interrupt function for GPIO interrupt 22. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00b - Interrupt n is low-level sensitive. 01b - Interrupt n is high-level sensitive. 10b - Interrupt n is rising-edge sensitive. 11b - Interrupt n is falling-edge sensitive.
11-10 ICR21	ICR21 Interrupt configuration 2 fields. This register controls the active condition of the interrupt function for GPIO interrupt 21. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00b - Interrupt n is low-level sensitive. 01b - Interrupt n is high-level sensitive. 10b - Interrupt n is rising-edge sensitive. 11b - Interrupt n is falling-edge sensitive.

Table continues on the next page...

Field	Function
9-8 ICR20	ICR20 Interrupt configuration 2 fields. This register controls the active condition of the interrupt function for GPIO interrupt 20. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00b - Interrupt n is low-level sensitive. 01b - Interrupt n is high-level sensitive. 10b - Interrupt n is rising-edge sensitive. 11b - Interrupt n is falling-edge sensitive.
7-6 ICR19	ICR19 Interrupt configuration 2 fields. This register controls the active condition of the interrupt function for GPIO interrupt 19. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00b - Interrupt n is low-level sensitive. 01b - Interrupt n is high-level sensitive. 10b - Interrupt n is rising-edge sensitive. 11b - Interrupt n is falling-edge sensitive.
5-4 ICR18	ICR18 Interrupt configuration 2 fields. This register controls the active condition of the interrupt function for GPIO interrupt 18. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00b - Interrupt n is low-level sensitive. 01b - Interrupt n is high-level sensitive. 10b - Interrupt n is rising-edge sensitive. 11b - Interrupt n is falling-edge sensitive.
3-2 ICR17	ICR17 Interrupt configuration 2 fields. This register controls the active condition of the interrupt function for GPIO interrupt 17. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00b - Interrupt n is low-level sensitive. 01b - Interrupt n is high-level sensitive. 10b - Interrupt n is rising-edge sensitive. 11b - Interrupt n is falling-edge sensitive.
1-0 ICR16	ICR16 Interrupt configuration 2 fields. This register controls the active condition of the interrupt function for GPIO interrupt 16. Settings: Bits ICRn[1:0] determine the interrupt condition for signal n as follows: 00b - Interrupt n is low-level sensitive. 01b - Interrupt n is high-level sensitive. 10b - Interrupt n is rising-edge sensitive. 11b - Interrupt n is falling-edge sensitive.

12.5.7 GPIO interrupt mask register (IMR)

12.5.7.1 Offset

Register	Offset
IMR	14h

12.5.7.2 Function

GPIO_IMR contains masking bits for each interrupt line.

12.5.7.3 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

12.5.7.4 Fields

Field	Function
31-0 IMR	IMR Interrupt Mask bits. This register is used to enable or disable the interrupt function on each of the 32 GPIO signals. Settings: Bit IMR[n] (n=0...31) controls interrupt n as follows: 0b - Interrupt n is disabled. 1b - Interrupt n is enabled.

12.5.8 GPIO interrupt status register (ISR)

12.5.8.1 Offset

Register	Offset
ISR	18h

12.5.8.2 Function

The GPIO_ISR 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.

12.5.8.3 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

12.5.8.4 Fields

Field	Function
31-0 ISR	ISR Interrupt status bits - 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 GPIO_IMR.

Field	Function
	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.

12.5.9 GPIO edge select register (EDGE_SEL)

12.5.9.1 Offset

Register	Offset
EDGE_SEL	1Ch

12.5.9.2 Function

GPIO_EDGE_SEL 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).

12.5.9.3 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

12.5.9.4 Fields

Field	Function
31-0	GPIO_EDGE_SEL

Field	Function
GPIO_EDGE_SEL	Edge select. When GPIO_EDGE_SEL[*n*] is set, the GPIO disregards the ICR[*n*] setting, and detects any edge on the corresponding input signal.

12.5.10 GPIO data register SET (DR_SET)

12.5.10.1 Offset

Register	Offset
DR_SET	84h

12.5.10.2 Function

The SET register of GPIO_DR.

12.5.10.3 Diagram

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

12.5.10.4 Fields

Field	Function
31-0 DR_SET	DR_SET The SET register of GPIO_DR.

12.5.11 GPIO data register CLEAR (DR_CLEAR)

12.5.11.1 Offset

Register	Offset
DR_CLEAR	88h

12.5.11.2 Function

The CLEAR register of GPIO_DR.

12.5.11.3 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

12.5.11.4 Fields

Field	Function
31-0 DR_CLEAR	DR_CLEAR The CLEAR register of GPIO_DR.

12.5.12 GPIO data register TOGGLE (DR_TOGGLE)

12.5.12.1 Offset

Register	Offset
DR_TOGGLE	8Ch

12.5.12.2 Function

The TOGGLE register of GPIO_DR.

12.5.12.3 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

12.5.12.4 Fields

Field	Function
31-0 DR_TOGGLE	DR_TOGGLE The TOGGLE register of GPIO_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.

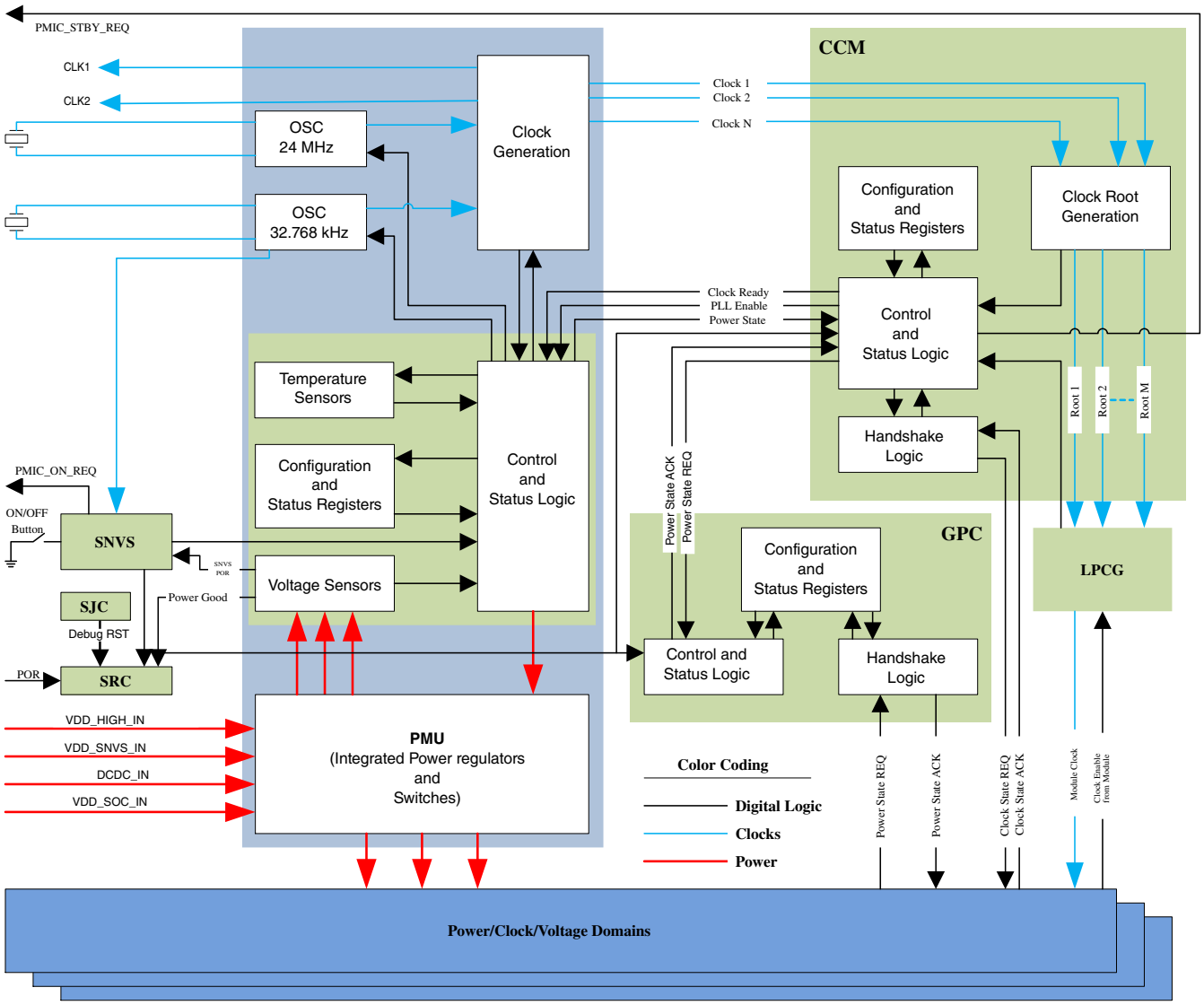


Figure 13-1. Power and clock management framework

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.

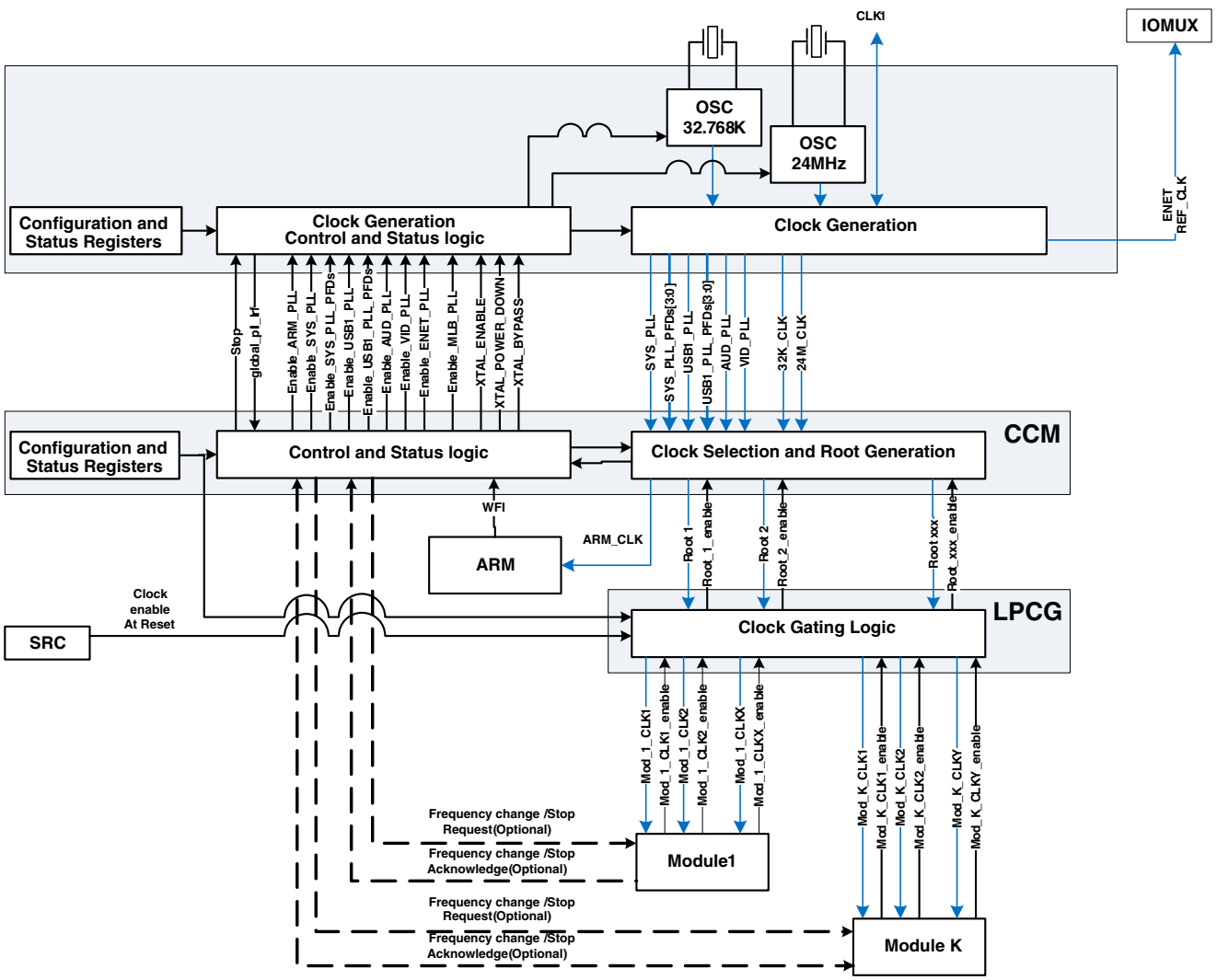


Figure 13-2. Clock Management System

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

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.

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.

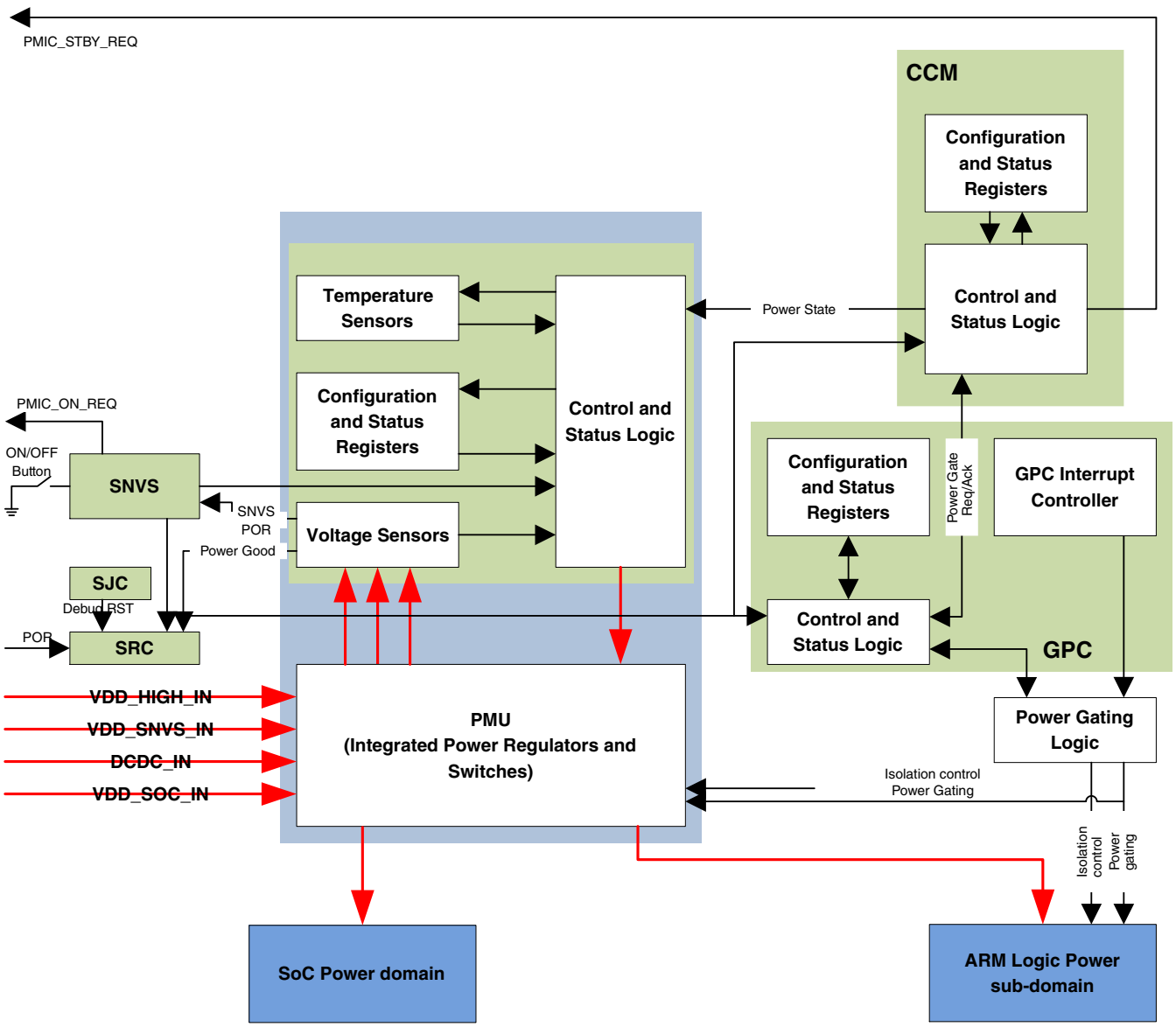


Figure 13-3. Power Management System

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.

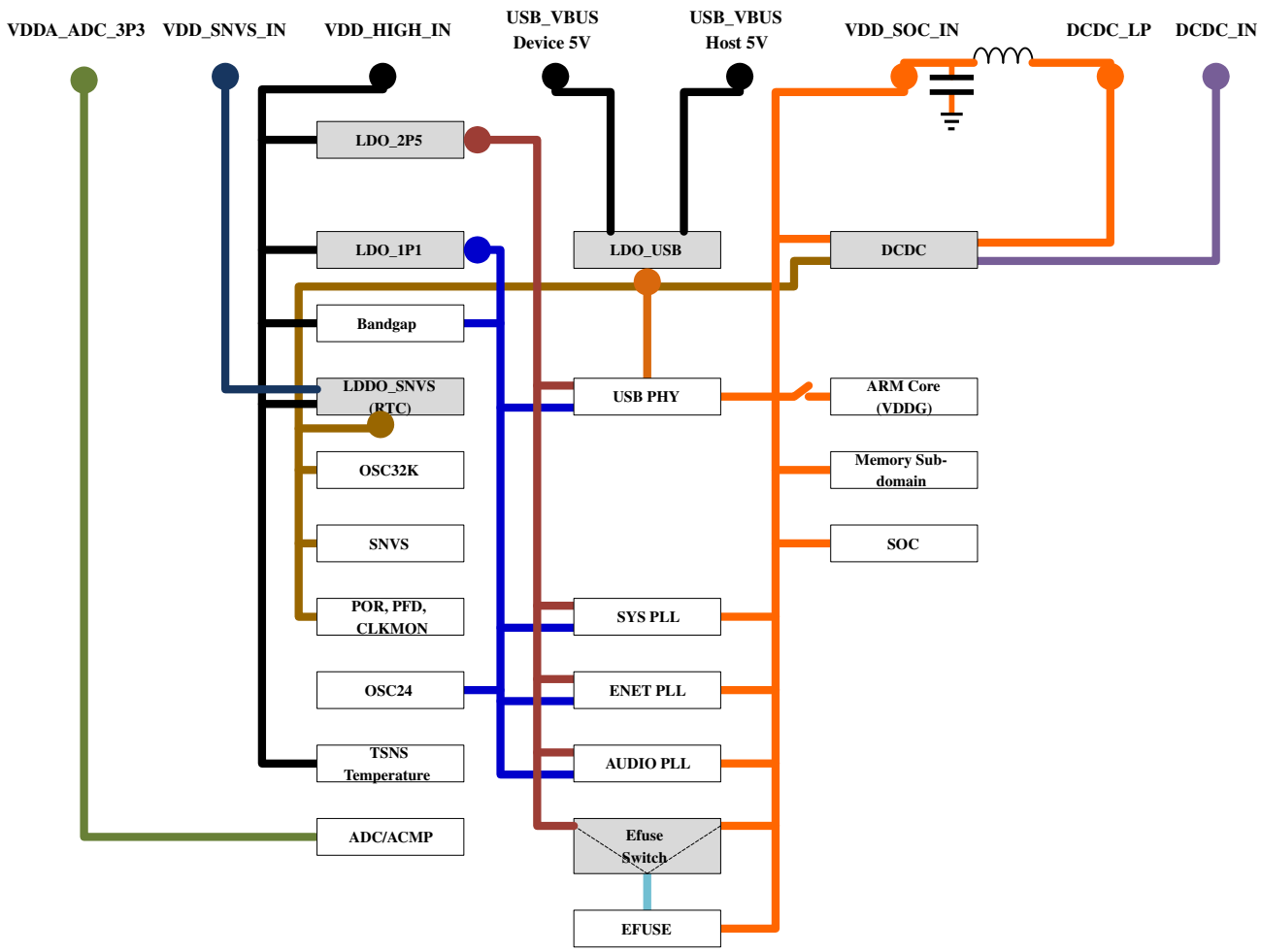


Figure 13-4. SOC Power Tree

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.

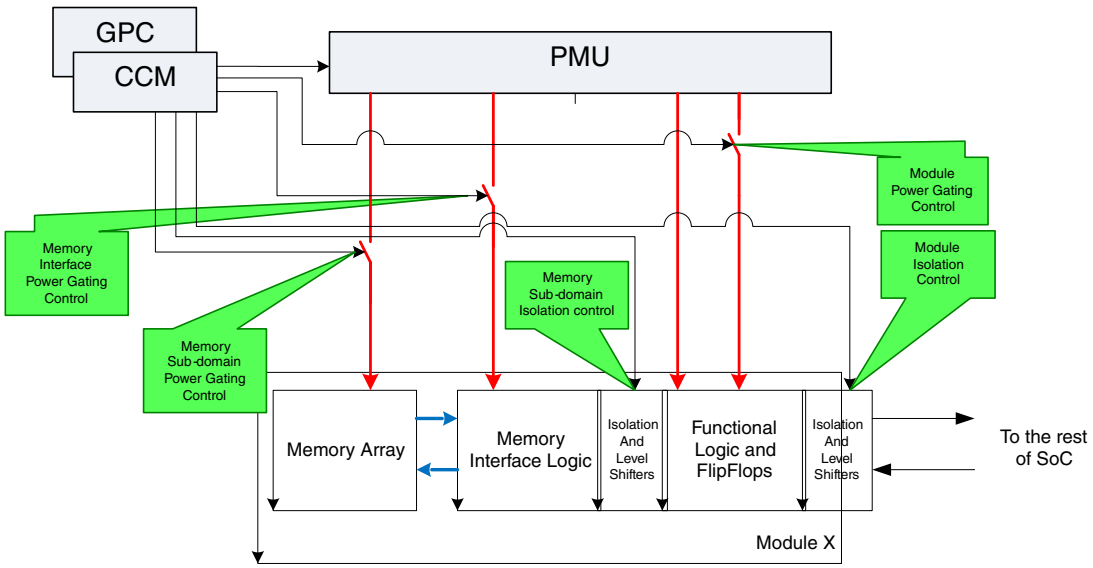


Figure 13-5. Power domain interface

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 (except for TCM memory arrays and controller). This domain can be supplied from internal or external controllable regulator.
- FlexRAM memory array - 128 KB always-on 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.
- 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.
- 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.

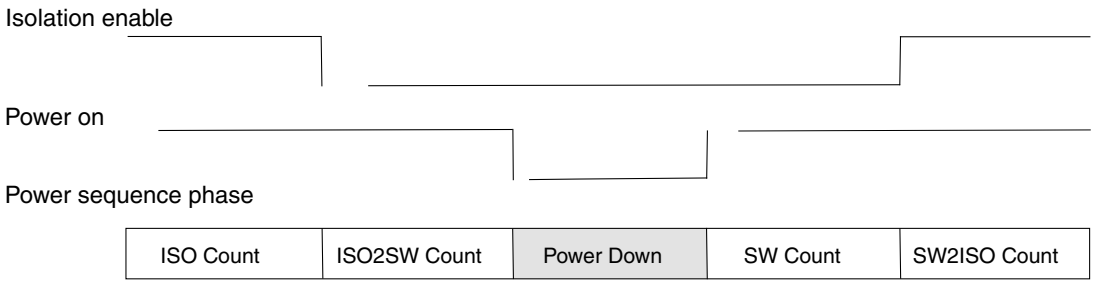


Figure 13-6. Arm Core Platform isolation and power on switch flow

13.4.1.4.3.2 FlexRAM PDRAMx

1. Configure the GPC/PGC Registers (GPC Memory Map/Register Definition) as follows to power-down isolate the PDRAM*x* logic from the rest of the SoC logic (PGC_MEGA_x Registers controls PDRAM1 domain, while PDRAM0 domain follows PGC_CPU_x Register configuration):

Configure the GPC/PGC PDNSCR Register ISO bits. These bits determine the delay between the power-down request to enabling the LDO domain isolation.

Configure the GPC/PGC PUPSCR Register SW2ISO bits. These bits determine the power-up request to the LDO domain isolation disabling.

Configure the GPC/PGC CTRL[PCR] bit to allow the power down of the block.

Configure the GPC/PGC GPC_CNTR to power down PDRAM*x*

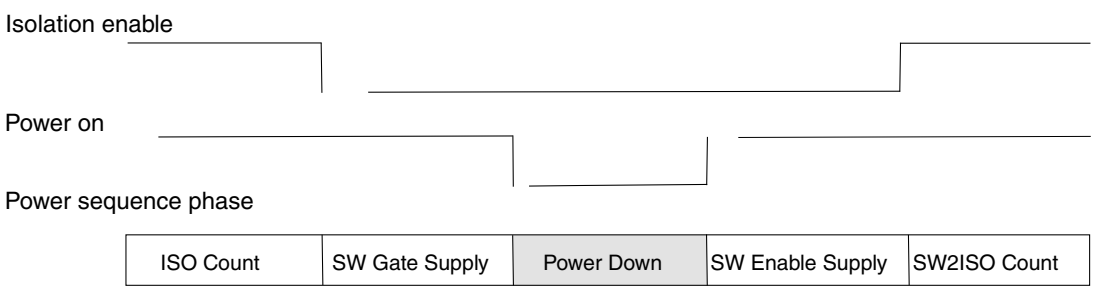


Figure 13-7. PDRAMx isolation and power on switch flow

13.4.1.4.3.3 SoC

For additional power reduction it is possible to do the following:

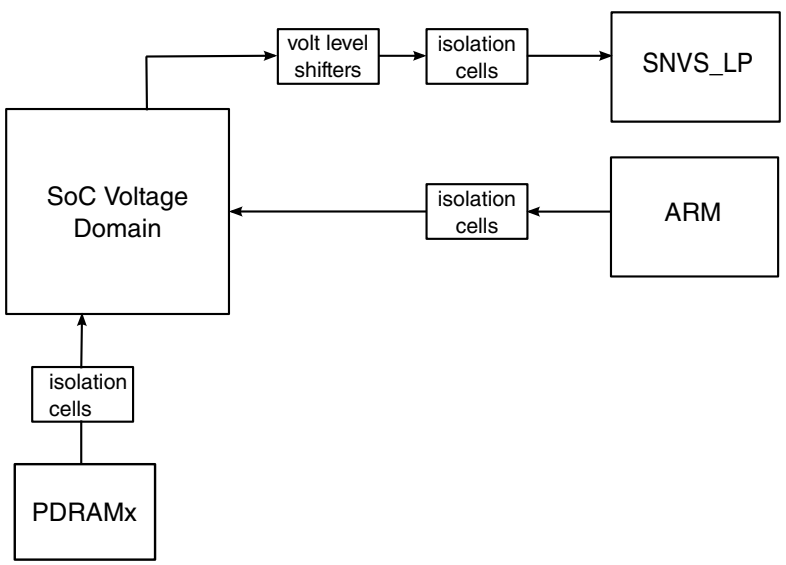
- Power-down the internal oscillator by configuring the following bits CCM_CCR[COSC_EN] (CCM Memory Map/Register Definition). This can be done only in case there is no dependency on 24 MHz XTAL for wake-up.
- It is possible to turn off and turn on the PMIC supplies to the SoC even when the SoC supplies 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 13-8. 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

Table continues on the next page...

Table 13-1. Low Power Mode Definition (continued)

	Overdrive Run	Full Speed Run	Low Power Run	System Idle	Low Power Idle	Suspend	SNVS
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
FlexRAM	ON	ON	ON	ON	ON	ON	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.

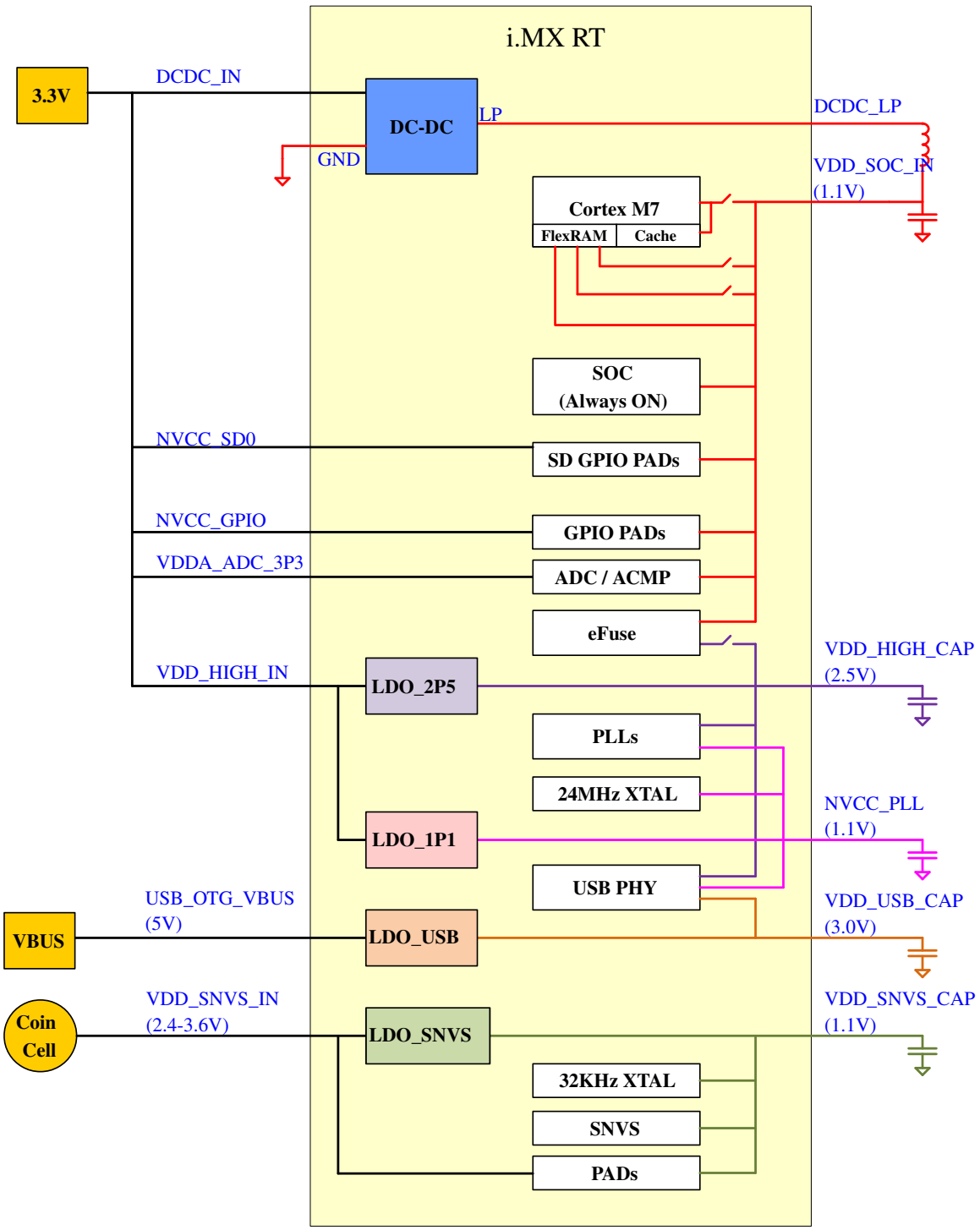


Figure 13-9. SOC Power Architecture

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.

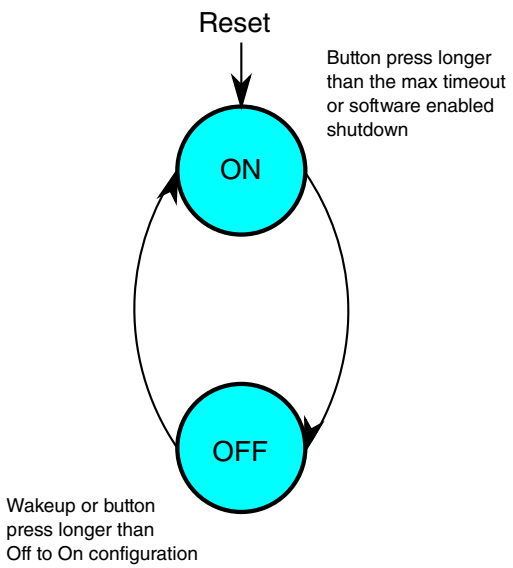


Figure 13-10. Dumb PMIC Mode State Machine

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.

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:
  - PLL2 also referenced as System PLL
  - PLL3 also referenced as USB1 PLL
  - PLL4 also referenced as Audio PLL
  - PLL6 also referenced as ENET PLL

NOTE

ENET PLL is also used as the Arm reference clock.

- 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 (CORE, PERIPH, 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

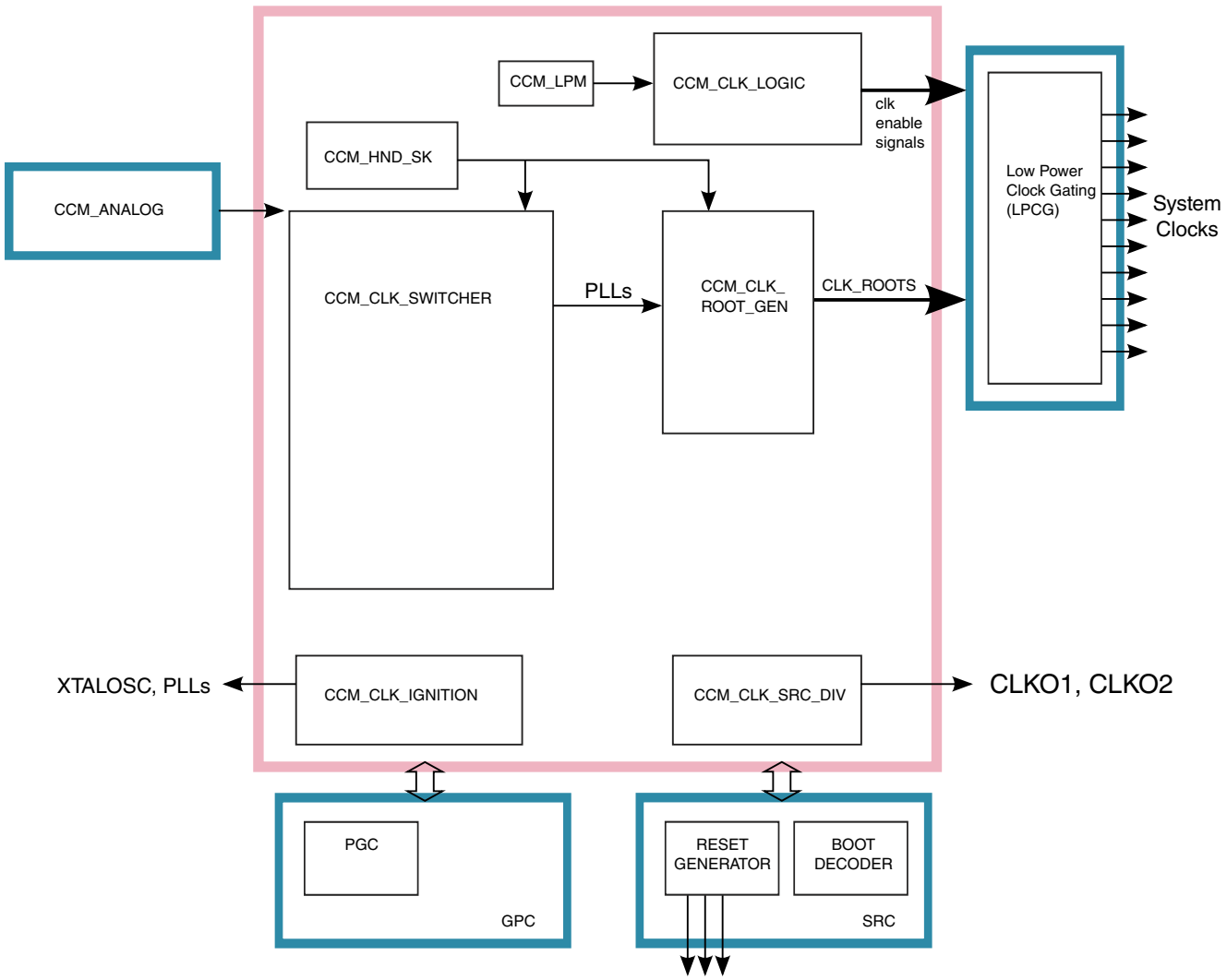


Figure 14-1. Block Diagram

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_CLKO1	Observability clock 1 output	GPIO_SD_02	ALT3	O
CCM_CLKO2	Observability clock 2 output	GPIO_SD_01	ALT3	O
CCM_PMIC_READY	Signal goes to STANBY_REQ pin, which notifies external power management IC to move from functional voltage to standby voltage.	GPIO_SD_13	ALT3	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.

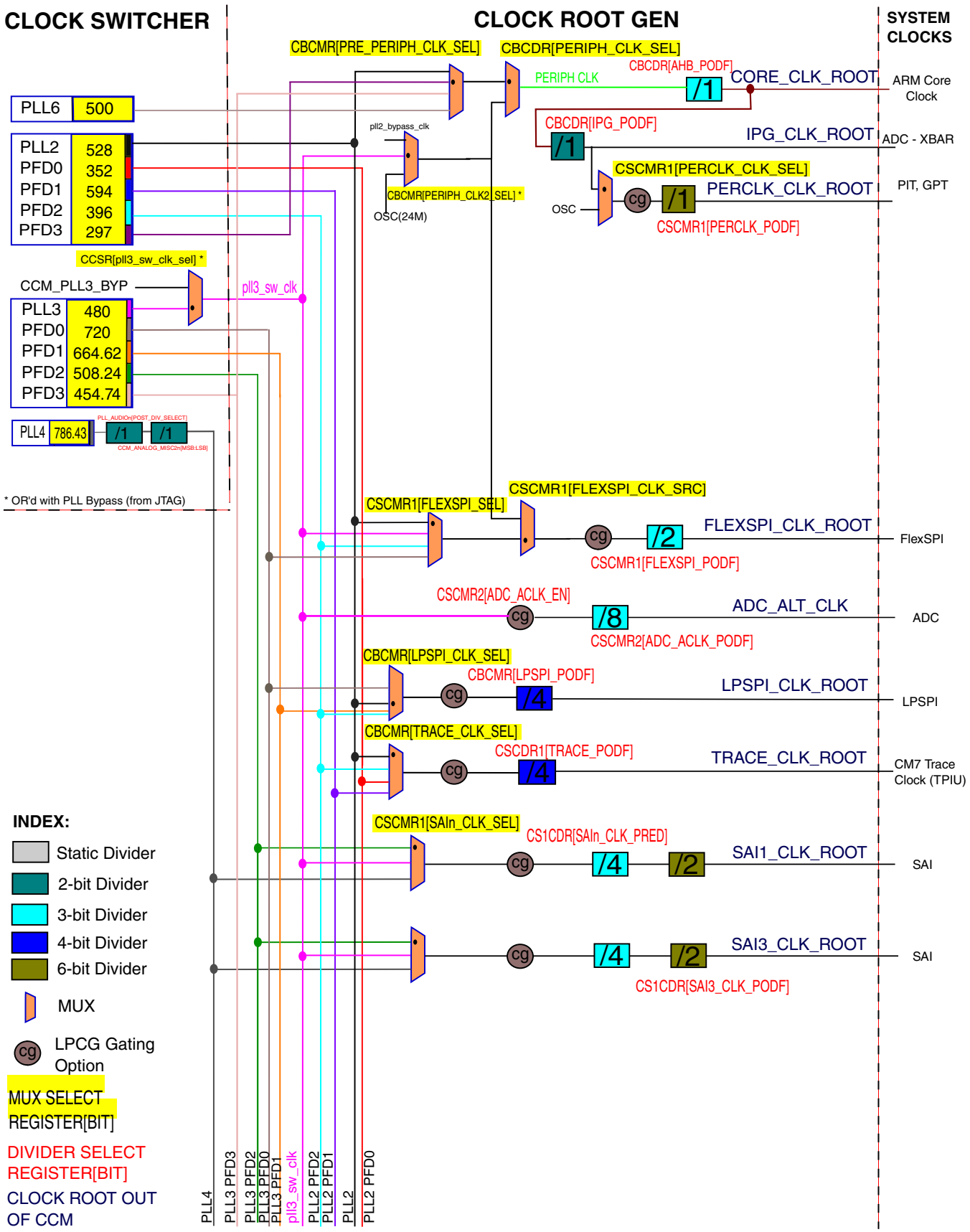


Figure 14-2. Clock Tree - Part 1

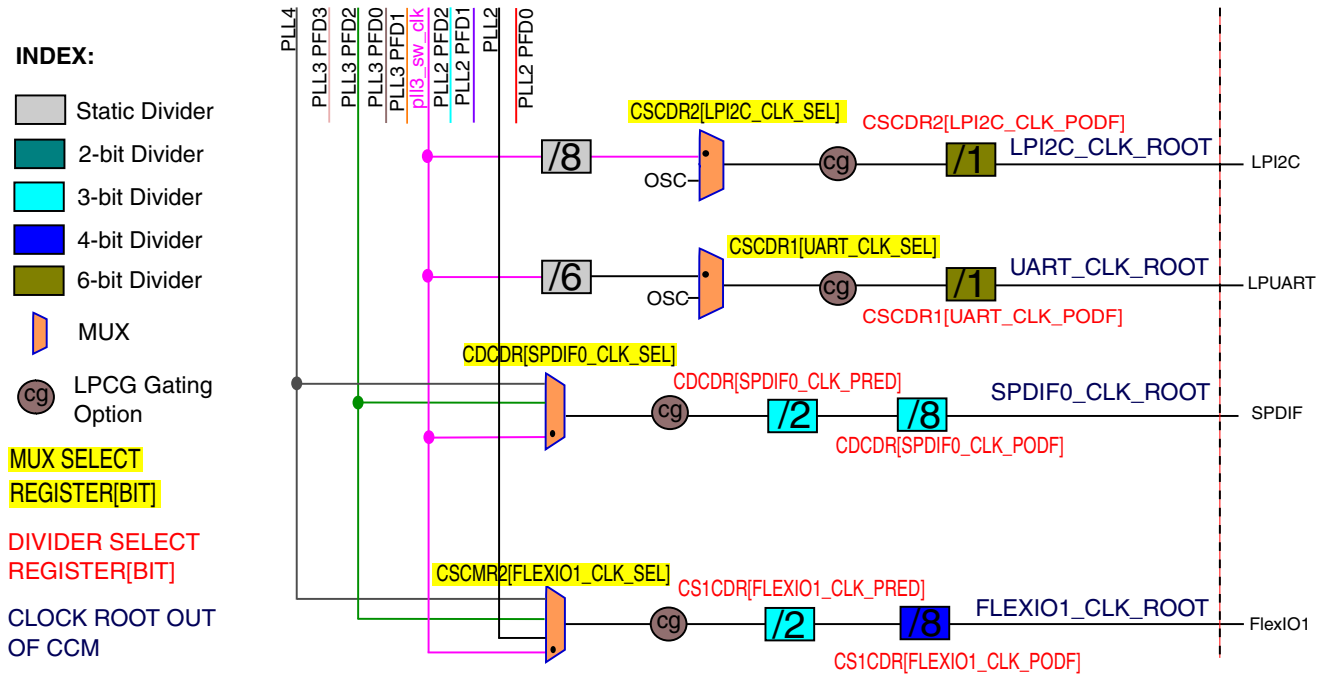


Figure 14-3. Clock Tree - Part 2

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
ADC*n*	adc1_ipg_clk	ipg_clk_root	CCGR1[CG8] (adc1_clk_enable)	
ADC_ETC	adc_etc_ipg_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
AIPS	aips_tz1_hclk	ipg_clk_root	CCGR0[CG0] (aips_tz1_clk_enable)	
	aips_tz2_hclk	ipg_clk_root	CCGR0[CG1] (aips_tz2_clk_enable)	
AOI*n*	aoi1_ipg_clk	ipg_clk_root	CCGR3[CG4] (aoi1_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	core_clk_root		
	cm7_mxrt_axi_clk	ipg_clk_root		
CCM	ccm_ccm_ipg_clk_s	ipg_clk_root		
	ccm_ipg_clk	ipg_clk_root		
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)	
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)	
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		
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)	

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
	flexio1_flexio_clk	flexio1_clk_root	CCGR5[CG1] (flexio1_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)	
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)	
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		
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)	
	gpio5_ipg_clk_s	ipg_clk_root	CCGR1[CG15] (gpio5_clk_enable)	
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]

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
	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)	
LPI2C*n*	lpi2c1_ipg_clk	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	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)	
LPSPI*n*	lpspi1_ipg_clk	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)	
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)	
	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)	
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
ROMCP	rom_64k_rom_CLK	ipg_clk_root	CCGR5[CG0] (rom_clk_enable)	
	romcp_hclk	ipg_clk_root	CCGR5[CG0] (rom_clk_enable)	
	romcp_hclk_reg	ipg_clk_root	CCGR5[CG0] (rom_clk_enable)	
	romcp_sec_mst_hclk	ipg_clk_root	CCGR5[CG0] (rom_clk_enable)	
SAI*n*	sai1_ipg_clk	ipg_clk_root	CCGR5[CG9] (sai1_clk_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)	
	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 IOMUXC_GPR_GPR1[SAI3_MCLK2_SEL]	CCGR5[CG11] (sai3_clk_enable)	
	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)	

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
	sai3_ipt_clk_sai_mclk	sai3_clk_root	CCGR5[CG11] (sai3_clk_enable)	
SIM	sim_ems_mainclk	ipg_clk_root	CCGR4[CG7] (sim_ems_clk_enable)	
	sim_m_mainclk	ipg_clk_root	CCGR4[CG6] (sim_m_clk_enable)	
	sim_m_mainclk_r	ipg_clk_root	CCGR0[CG4] (sim_m_mainclk_r_enable)	
	sim_m7_mainclk	ipg_clk_root	CCGR4[CG4] (sim_m7_clk_enable)	
	sim_m7_mainclk_r	ipg_clk_root	CCGR4[CG0] (sim_m7_mainclk_r_enable)	
	sim_main_mainclk	ipg_clk_root	CCGR5[CG8] (sim_main_clk_enable)	
	sim_main_mainclk_r	ipg_clk_root	CCGR0[CG4] (sim_m_mainclk_r_enable)	
	sim_per_mainclk	ipg_clk_root	CCGR6[CG10] (sim_per_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]
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
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)	

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**
CORE_CLK_ROOT	12	500				
IPG_CLK_ROOT	3	150	132	24	132	24
PERCLK_CLK_ROOT	6	75	72	24	72	24
FLEXSPI_CLK_ROOT	2	332	332	24	332	24
LPSPI_CLK_ROOT	6	132	132	24	132	24
TRACE_CLK_ROOT	6	132	132	24	132	24
SAI1_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
UART_CLK_ROOT	4	80	80	24	80	24
SPDIF0_CLK_ROOT	1.5	66	66	24	66	24
FLEXIO1_CLK_ROOT	1.5	120	120	24	120	24
ADC_ALT_CLK	2	60				

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.

PLL2 - System PLL (functional frequency 528 MHz)

PLL3 - USB1 PLL (functional frequency 480 MHz)

PLL4 - Audio PLL

PLL6 - ENET PLL

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 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.

14.6.1.3.2 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.

14.6.1.3.3 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)

14.6.1.3.4 Ethernet PLL (PLL6)

This PLL synthesizes a low jitter clock from the 24 MHz reference clock.

The reference clocks generated by this PLL are:

- fixed 500 MHz

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.

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.

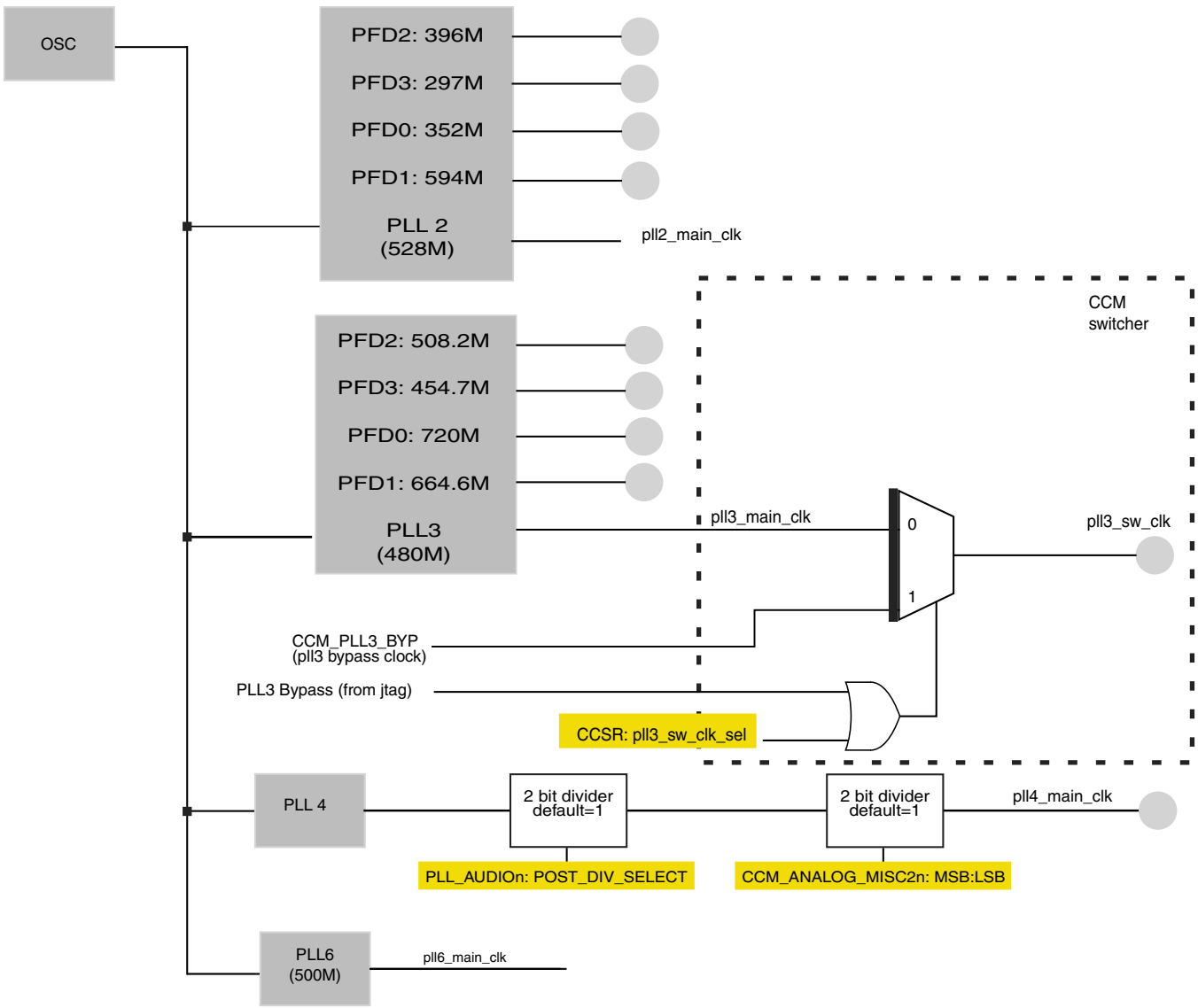


Figure 14-4. Switcher clock generation

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.

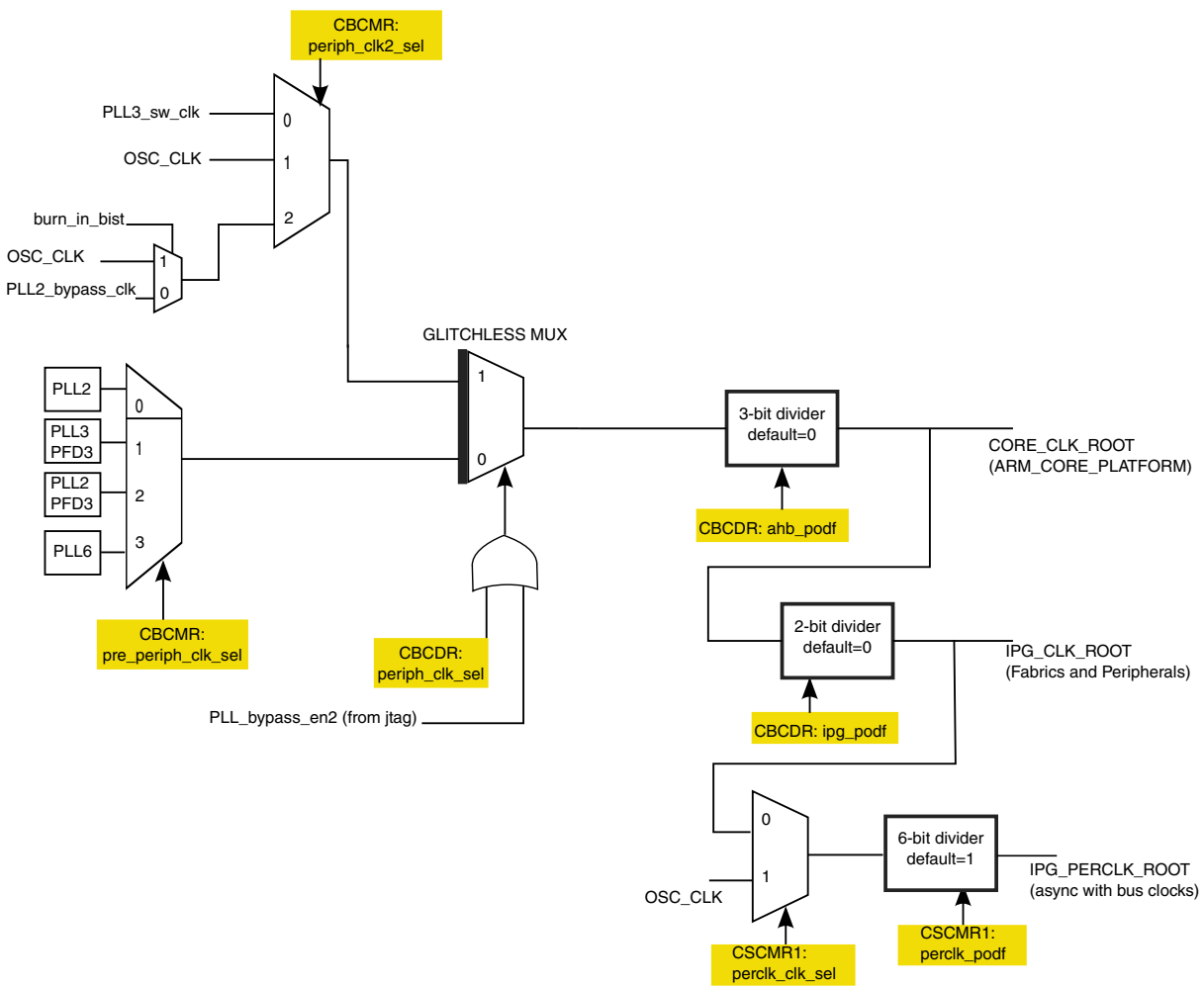


Figure 14-5. BUS clock generation

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
- ahb_podf
- perclk_podf
- flexspi_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]
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 CORE_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.

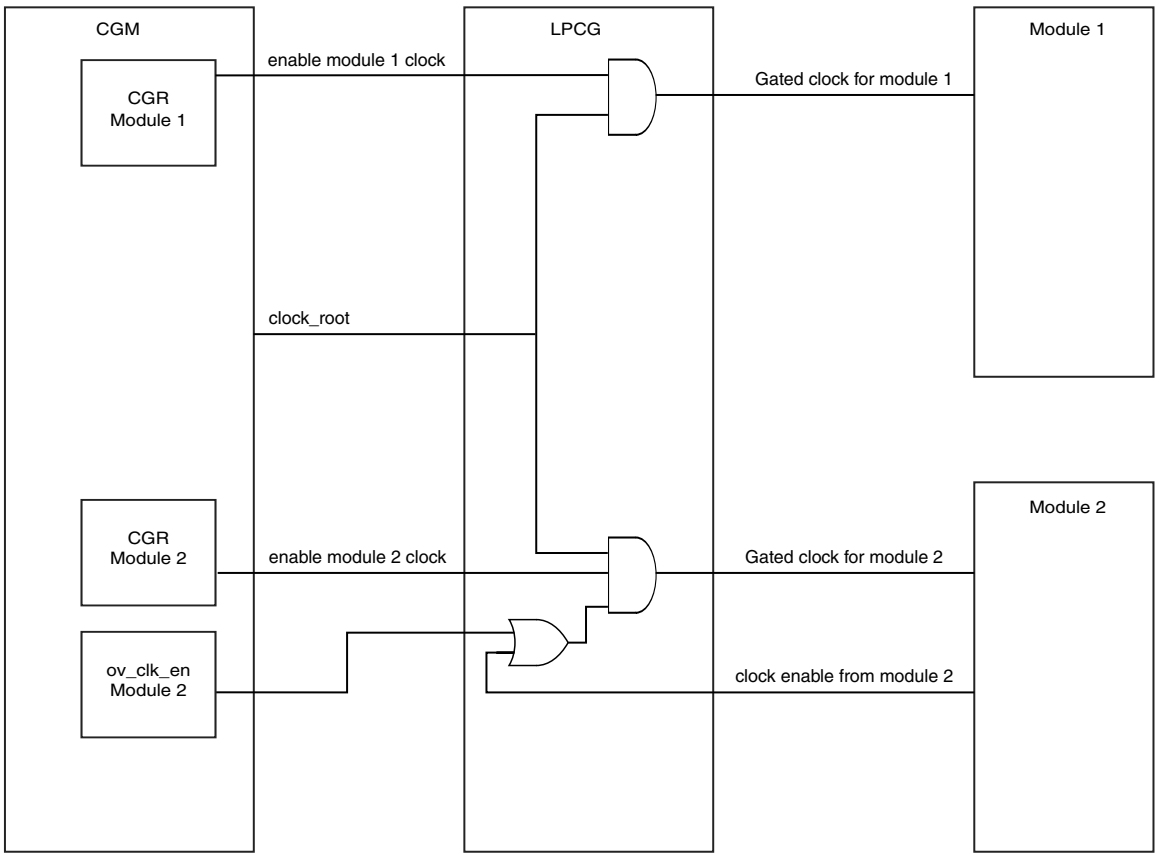


Figure 14-6. Clock split in LPCG

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	14.7.1/539
400F_C008	CCM Status Register (CCM_CSR)	32	R	0000_0010h	14.7.2/541
400F_C00C	CCM Clock Switcher Register (CCM_CCSR)	32	R/W	0000_0000h	14.7.3/543
400F_C014	CCM Bus Clock Divider Register (CCM_CBCDR)	32	R/W	0002_8000h	14.7.4/544
400F_C018	CCM Bus Clock Multiplexer Register (CCM_CBCMR)	32	R/W	0C08_8020h	14.7.5/546
400F_C01C	CCM Serial Clock Multiplexer Register 1 (CCM_CSCMR1)	32	R/W	0080_FC00h	14.7.6/548
400F_C020	CCM Serial Clock Multiplexer Register 2 (CCM_CSCMR2)	32	R/W	0018_0000h	14.7.7/550
400F_C024	CCM Serial Clock Divider Register 1 (CCM_CSCDR1)	32	R/W	0600_0000h	14.7.8/551

Table continues on the next page...

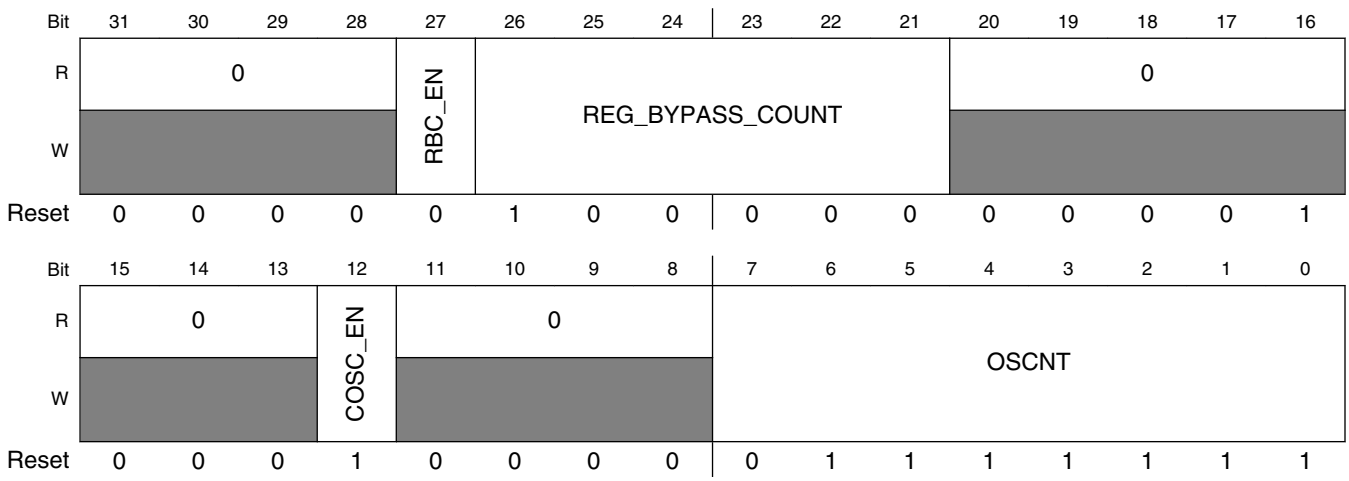
CCM memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400F_C028	CCM Clock Divider Register (CCM_CS1CDR)	32	R/W	0EC1_02C1h	14.7.9/553
400F_C030	CCM D1 Clock Divider Register (CCM_CDCDR)	32	R/W	03F0_0000h	14.7.10/ 555
400F_C038	CCM Serial Clock Divider Register 2 (CCM_CSCDR2)	32	R/W	0003_9000h	14.7.11/ 556
400F_C048	CCM Divider Handshake In-Process Register (CCM_CDHIPR)	32	R	0000_0000h	14.7.12/ 558
400F_C054	CCM Low Power Control Register (CCM_CLPCR)	32	R/W	0000_0079h	14.7.13/ 561
400F_C058	CCM Interrupt Status Register (CCM_CISR)	32	w1c	0000_0000h	14.7.14/ 563
400F_C05C	CCM Interrupt Mask Register (CCM_CIMR)	32	R/W	FFFF_FFFFh	14.7.15/ 566
400F_C060	CCM Clock Output Source Register (CCM_CCOSR)	32	R/W	000A_0001h	14.7.16/ 568
400F_C064	CCM General Purpose Register (CCM_CGPR)	32	R/W	0000_C000h	14.7.17/ 570
400F_C068	CCM Clock Gating Register 0 (CCM_CCGR0)	32	R/W	FFFF_FFFFh	14.7.18/ 571
400F_C06C	CCM Clock Gating Register 1 (CCM_CCGR1)	32	R/W	FFFF_FFFFh	14.7.19/ 573
400F_C070	CCM Clock Gating Register 2 (CCM_CCGR2)	32	R/W	FFFF_FFFFh	14.7.20/ 574
400F_C074	CCM Clock Gating Register 3 (CCM_CCGR3)	32	R/W	FFFF_FFFFh	14.7.21/ 575
400F_C078	CCM Clock Gating Register 4 (CCM_CCGR4)	32	R/W	FFFF_FFFFh	14.7.22/ 577
400F_C07C	CCM Clock Gating Register 5 (CCM_CCGR5)	32	R/W	FFFF_FFFFh	14.7.23/ 578
400F_C080	CCM Clock Gating Register 6 (CCM_CCGR6)	32	R/W	FFFF_FFFFh	14.7.24/ 579
400F_C088	CCM Module Enable Overide Register (CCM_CMEOR)	32	R/W	FFFF_FFFFh	14.7.25/ 581

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. 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...

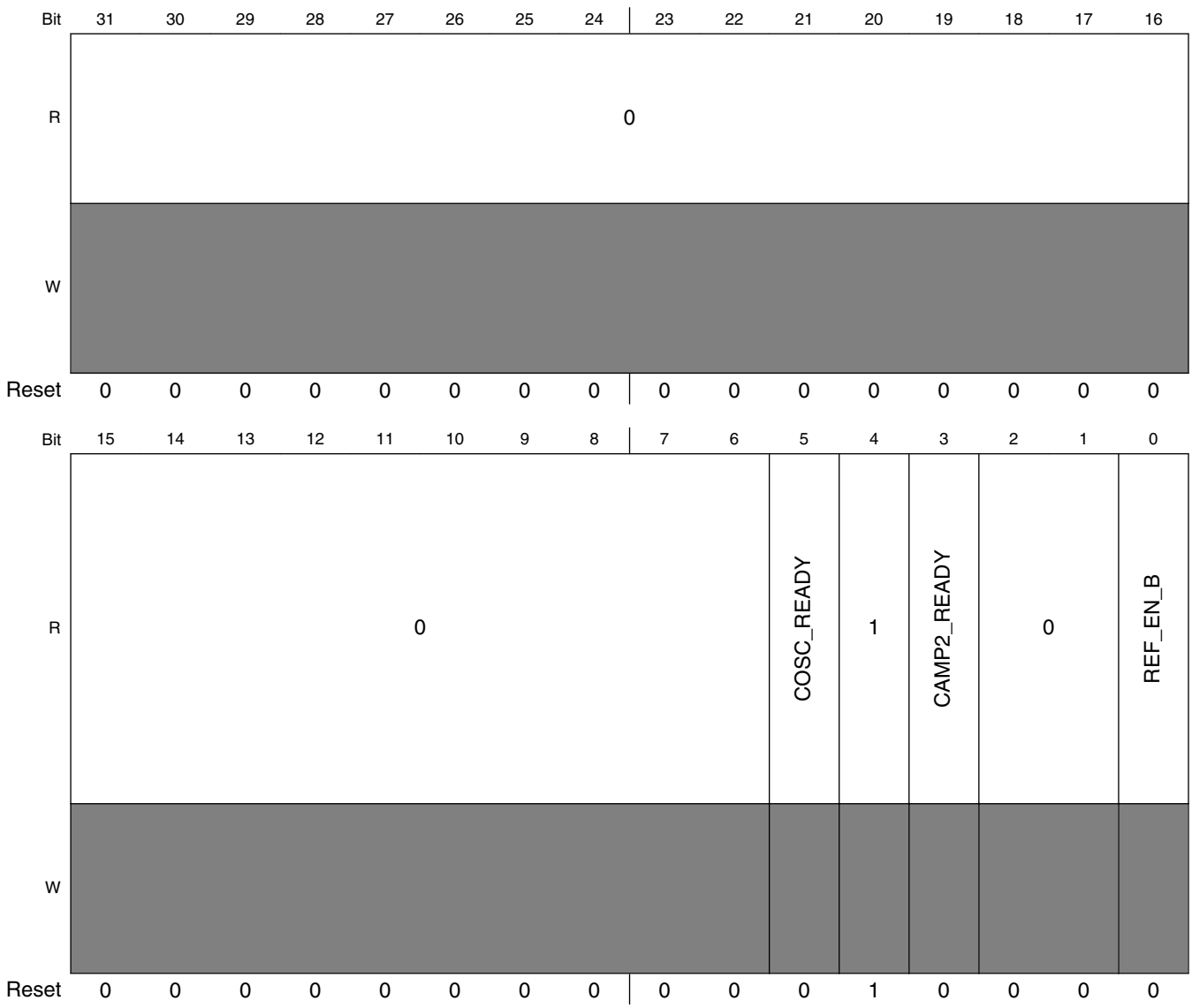
CCM_CCR field descriptions (continued)

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...

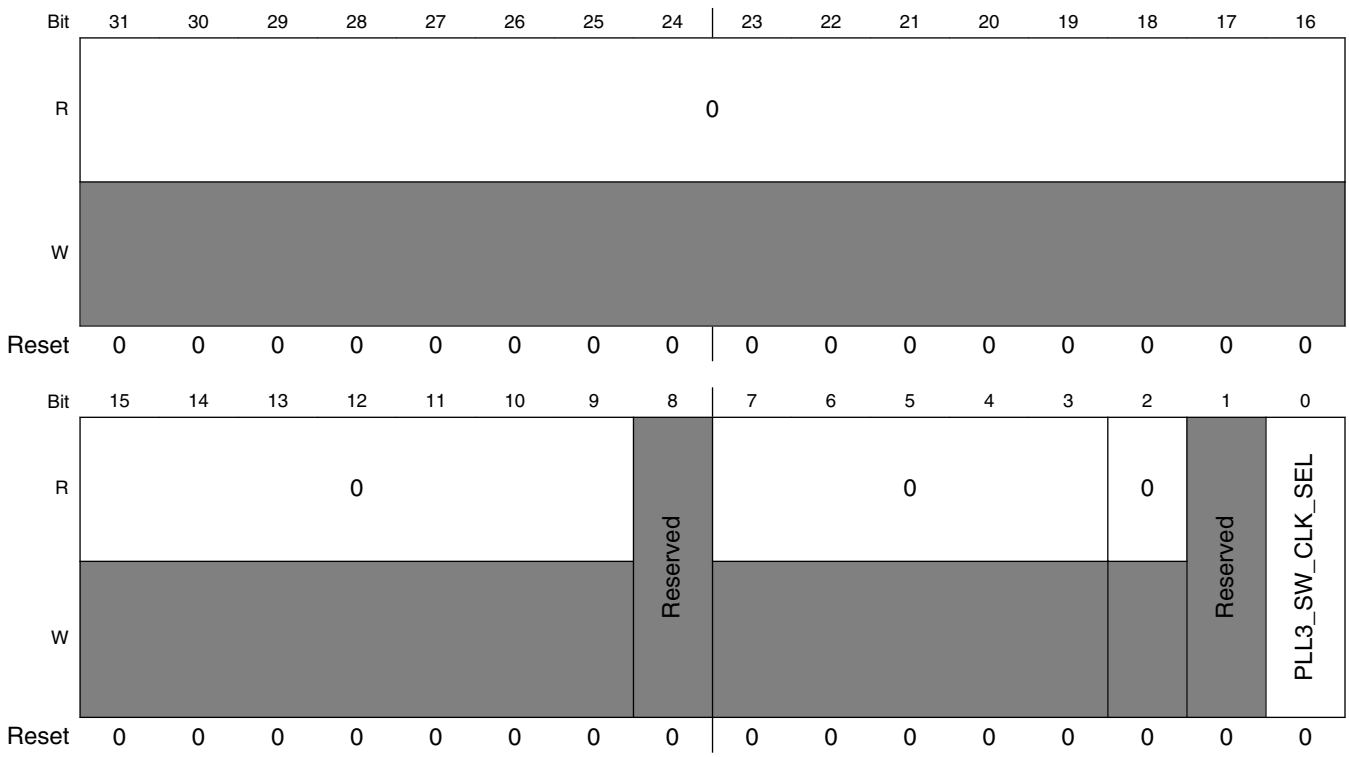
CCM_CSR field descriptions (continued)

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 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

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved		Reserved			Reserved	PERIPH_CLK_SEL		Reserved					Reserved		
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserved			AHB_PODF			IPG_PODF		Reserved		Reserved			Reserved		
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CCM_CBCDR field descriptions

Field	Description
31–30 -	This field is reserved. Reserved
29–27 -	This field is reserved. Reserved

Table continues on the next page...

CCM_CBCDR field descriptions (continued)

Field	Description
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 selected by CCM_CBCMR[CORE_CLK_PRE_SEL] 1 derive clock selected by CCM_CBCMR[PERIPH_CLK2_SEL]
24–19 -	This field is reserved. Reserved
18–16 -	This field is reserved. Reserved
15–13 -	This field is reserved. 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 4 100 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
7–6 -	This field is reserved. Reserved
5–3 -	This field is reserved. Reserved
-	This field is reserved. Reserved

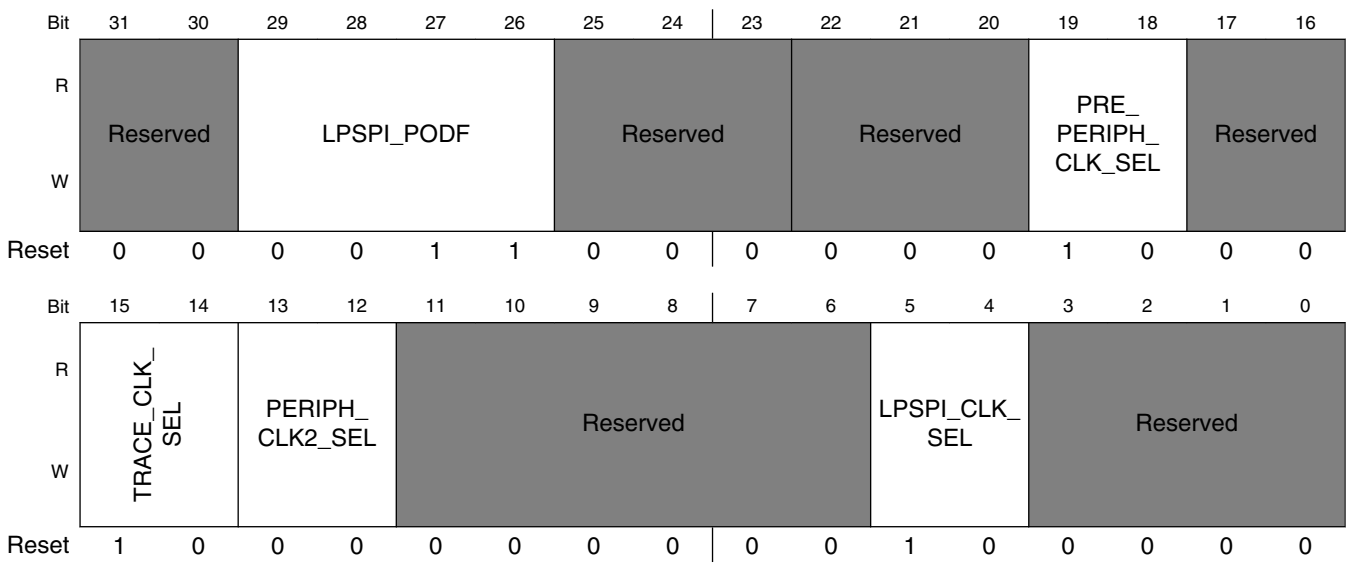
14.7.5 CCM Bus Clock Multiplexer Register (CCM_CBCMR)

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
31–30 -	This field is reserved. Reserved
29–26 LPSPI_PODF	Divider for LPSPI. **NOTE:** Divider should be updated when output clock is gated. 0000 divide by 1 0001 divide by 2 0010 divide by 3 0011 divide by 4 0100 divide by 5 0101 divide by 6 0110 divide by 7

Table continues on the next page...

CCM_CBCMR field descriptions (continued)

Field	Description
	0111 divide by 8 1000 divide by 9 1001 divide by 10 1010 divide by 11 1011 divide by 12 1100 divide by 13 1101 divide by 14 1110 divide by 15 1111 divide by 16
25–23 -	This field is reserved. Reserved
22–20 -	This field is reserved. Reserved
19–18 PRE_PERIPH_CLK_SEL	Selector for pre_periph clock multiplexer 00 derive clock from PLL2 01 derive clock from PLL3 PFD3 10 derive clock from PLL2 PFD3 11 derive clock from PLL6
17–16 -	This field is reserved. Reserved
15–14 TRACE_CLK_SEL	Selector for Trace clock multiplexer 00 derive clock from PLL2 01 derive clock from PLL2 PFD2 10 derive clock from PLL2 PFD0 11 derive clock from PLL2 PFD1
13–12 PERIPH_CLK2_SEL	Selector for peripheral clk2 clock multiplexer 00 derive clock from pll3_sw_clk 01 derive clock from osc_clk 10 derive clock from pll2_bypass_clk 11 reserved
11–6 -	This field is reserved. Reserved
5–4 LPSPI_CLK_SEL	Selector for lpspi clock multiplexer 00 derive clock from PLL3 PFD1 clk 01 derive clock from PLL3 PFD0 10 derive clock from PLL2 11 derive clock from PLL2 PFD2
-	This field is reserved. Reserved

14.7.6 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

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	FLEXSPI_CLK_SRC	FLEXSPI_CLK_SEL		Reserved			FLEXSPI_PODF			Reserved					Reserved	
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	SAI3_CLK_SEL		Reserved		SAI1_CLK_SEL		Reserved			PERCLK_CLK_SEL	PERCLK_PODF					
Reset	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0

CCM_CSCMR1 field descriptions

Field	Description
31 FLEXSPI_CLK_SRC	Select for source of flexspi_clk_root 0 derive clock selected by CCM_CSCMR1[FLEXSPI_CLK_SEL] 1 derive clock selected by CCM_CBCMR[PERIPH_CLK2_ SEL]
30–29 FLEXSPI_CLK_SEL	Selector for flexspi clock multiplexer 00 derive clock from PLL2 01 derive clock from pll3_sw_clk 10 derive clock from PLL2 PFD2 11 derive clock from PLL3 PFD0
28–26 -	This field is reserved. Reserved

Table continues on the next page...

CCM_CSCMR1 field descriptions (continued)

Field	Description
25–23 FLEXSPI_PODF	Divider for flexspi 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
22–18 -	This field is reserved. Reserved
17–16 -	This field is reserved. Reserved
15–14 SAI3_CLK_SEL	Selector for sai3 clock multiplexer 00 derive clock from PLL3 PFD2 01 derive from pll3_sw_clk 10 derive clock from PLL4 11 Reserved
13–12 -	This field is reserved. Reserved
11–10 SAI1_CLK_SEL	Selector for sai1 clock multiplexer 00 derive clock from PLL3 PFD2 01 derive from pll3_sw_clk 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 000100 divide by 5 000101 divide by 6 000110 divide by 7 111111 divide by 64

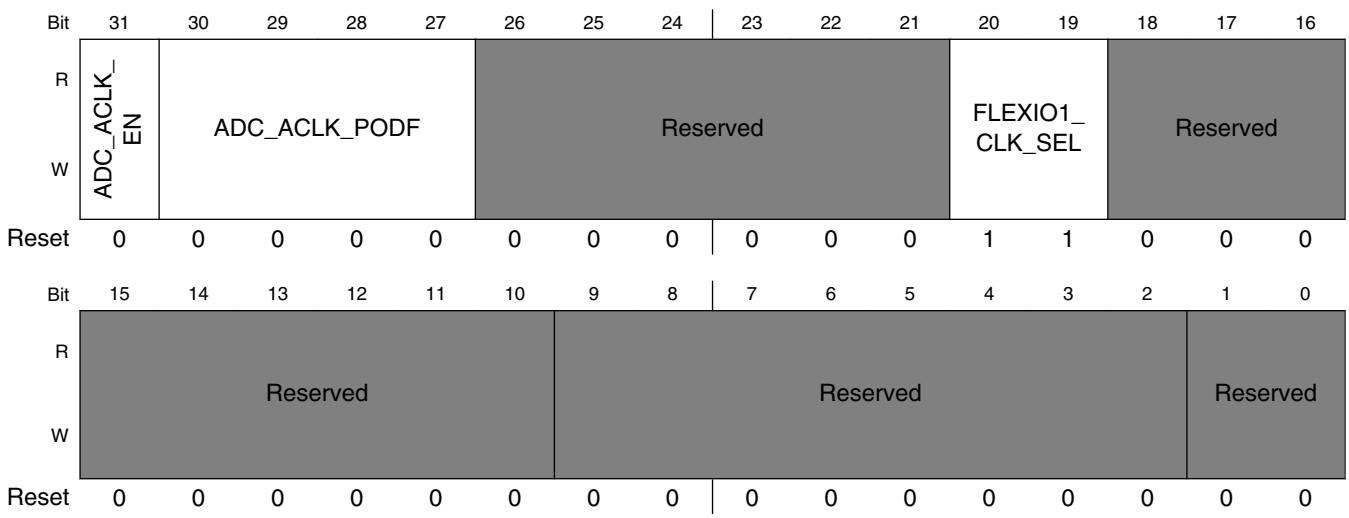
14.7.7 CCM Serial Clock Multiplexer Register 2 (CCM_CSCMR2)

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



CCM_CSCMR2 field descriptions

Field	Description
31 ADC_ACLK_EN	Enable ADC alt_clk, so that ADC alt_clk can be driven be divided pll3_sw_clk. 0 ADC alt_clk source is disabled 1 ADC alt_clk source is enabled
30–27 ADC_ACLK_PODF	Divider for ADC alt_clk, as the list below (other values reserved). **NOTE:** ADC_ACLK_PODF should NOT be updated when ADC_ACLK_EN = 1 0111 pll3_sw_clk / 8 1011 pll3_sw_clk / 12 1111 pll3_sw_clk / 16
26–21 -	This field is reserved. Reserved

Table continues on the next page...

CCM_CSCMR2 field descriptions (continued)

Field	Description
20–19 FLEXIO1_CLK_SEL	Selector for flexio1 clock multiplexer 00 derive clock from PLL4 divided clock 01 derive clock from PLL3 PFD2 clock 10 derive from PLL2 11 derive clock from pll3_sw_clk
18–10 -	This field is reserved. Reserved
9–2 -	This field is reserved. Reserved
-	This field is reserved. Reserved

14.7.8 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

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0			TRACE_PODF					Reserved					Reserved		
W																
Reset	0	0	0	0	0	1	1	0	0	0	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		Reserved				UART_CLK_SEL	UART_CLK_PODF						
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

CCM_CSCDR1 field descriptions

Field	Description
31–29 Reserved	This read-only field is reserved and always has the value 0.

Table continues on the next page...

CCM_CSCDR1 field descriptions (continued)

Field	Description
28–25 TRACE_PODF	Divider for trace 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 1000 divide by 9 1001 divide by 10 1010 divide by 11 1011 divide by 12 1100 divide by 13 1101 divide by 14 1110 divide by 15 1111 divide by 16
24–19 -	This field is reserved. Reserved
18–16 -	This field is reserved. Reserved
15–14 -	This field is reserved. Reserved
13–11 -	This field is reserved. Reserved
10–8 -	This field is reserved. Reserved.
7–6 UART_CLK_SEL	Selector for the UART clock multiplexor 00 derive clock from pll3_80m 01 derive clock from osc_clk 10 derive clock from per_clk_root 11 Reserved
UART_CLK_ PODF	Divider for uart clock podf. 000000 divide by 1 111111 divide by 2^6

14.7.9 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			FLEXIO1_CLK_PODF				SAI3_CLK_PRED			SAI3_CLK_PODF							0				FLEXIO1_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
31–29 Reserved	This read-only field is reserved and always has the value 0.
28–25 FLEXIO1_CLK_PODF	Divider for flexio1 clock. 0000 divide by 1 0001 divide by 2 0010 divide by 3 0011 divide by 4 0100 divide by 5 0101 divide by 6 0110 divide by 7 0111 divide by 8 1000 divide by 9 1001 divide by 10 1010 divide by 11 1011 divide by 12 1100 divide by 13 1101 divide by 14 1110 divide by 15 1111 divide by 16
24–22 SAI3_CLK_PRED	Divider for sai3 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

Table continues on the next page...

CCM_CS1CDR field descriptions (continued)

Field	Description
21–16 SAI3_CLK_ PODF	Divider for sai3 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 FLEXIO1_CLK_ PRED	Divider for flexio1 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

14.7.10 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	0	0	0	0	1	1	1	1	1	1	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	Reserv ed	Reserved							Reserved							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	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
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 3 111 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 1 111 divide by 8
21–20 SPDIF0_CLK_ SEL	Selector for spdif0 clock multiplexer 00 derive clock from PLL4 01 derive clock from PLL3 PFD2 10 Reserved 11 derive clock from pll3_sw_clk
19–15 -	This field is reserved. Reserved
14–7 -	This field is reserved. Reserved

Table continues on the next page...

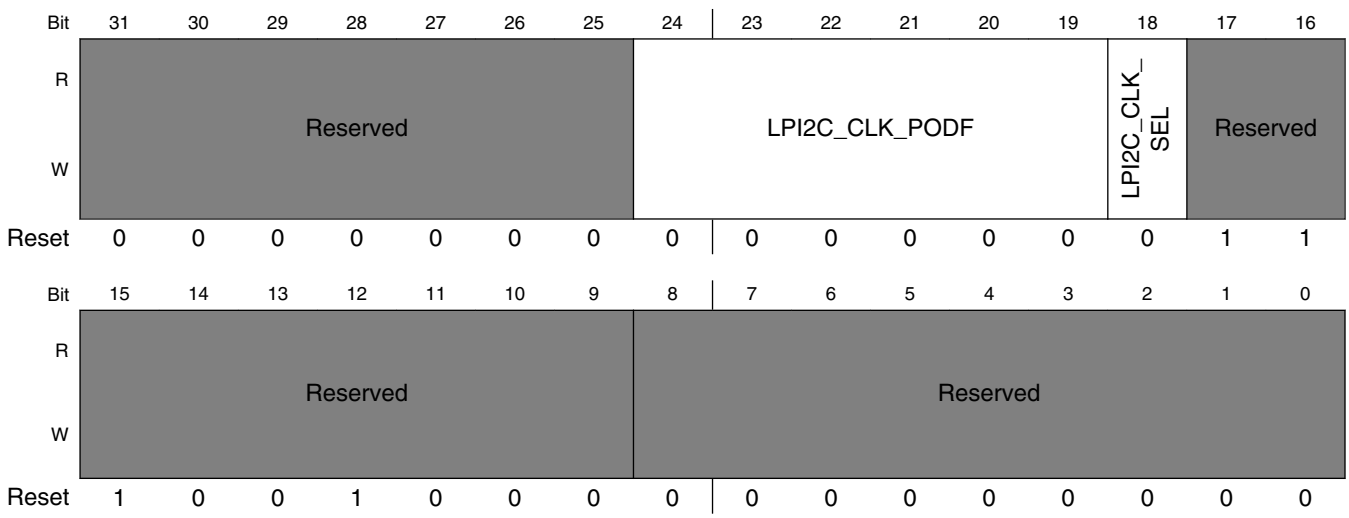
CCM_CDCDR field descriptions (continued)

Field	Description
-	This field is reserved. Reserved

14.7.11 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 -	This field is reserved. 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

Table continues on the next page...

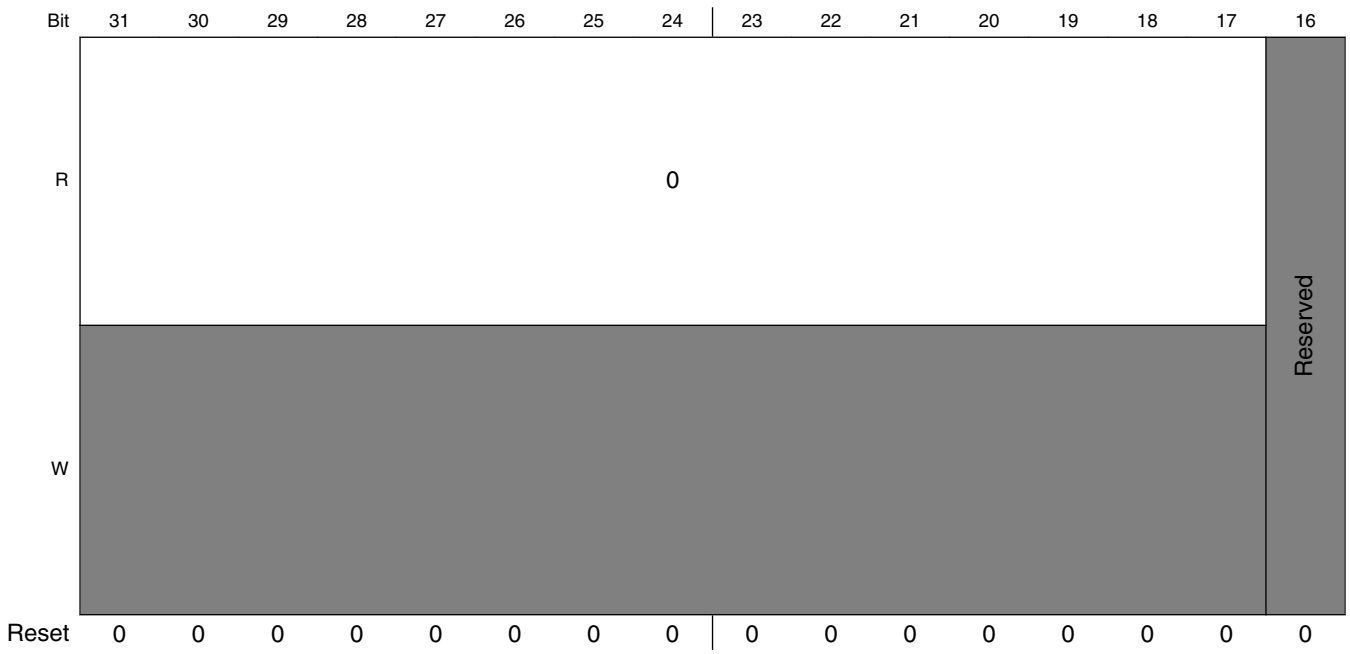
CCM_CSCDR2 field descriptions (continued)

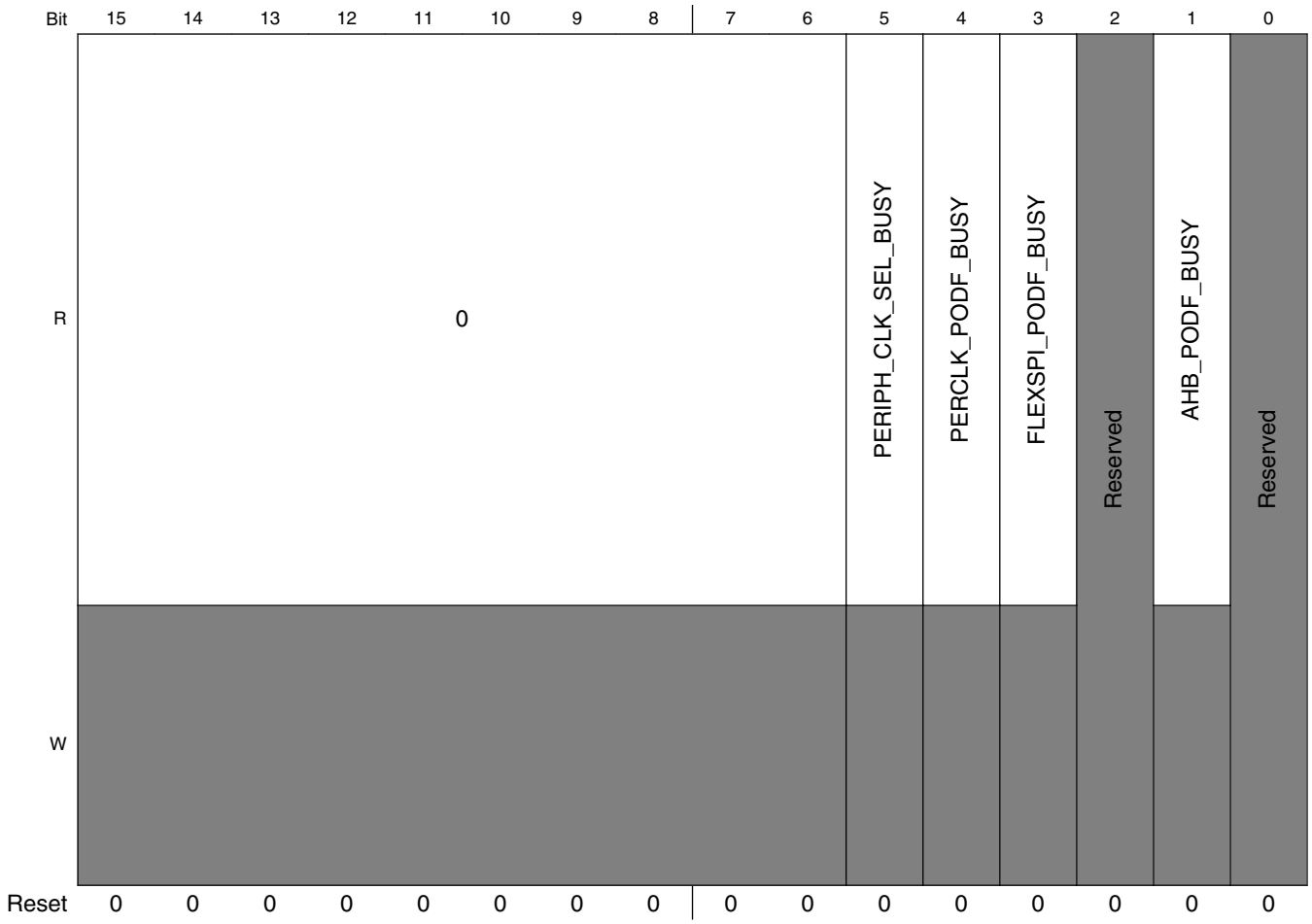
Field	Description
17–9 -	This field is reserved. Reserved
-	This field is reserved. Reserved

14.7.12 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





CCM_CDHIPR 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
15–6 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 PERCLK_PODF_BUSY	Busy indicator for perclk_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 perclk_podf will be applied.

Table continues on the next page...

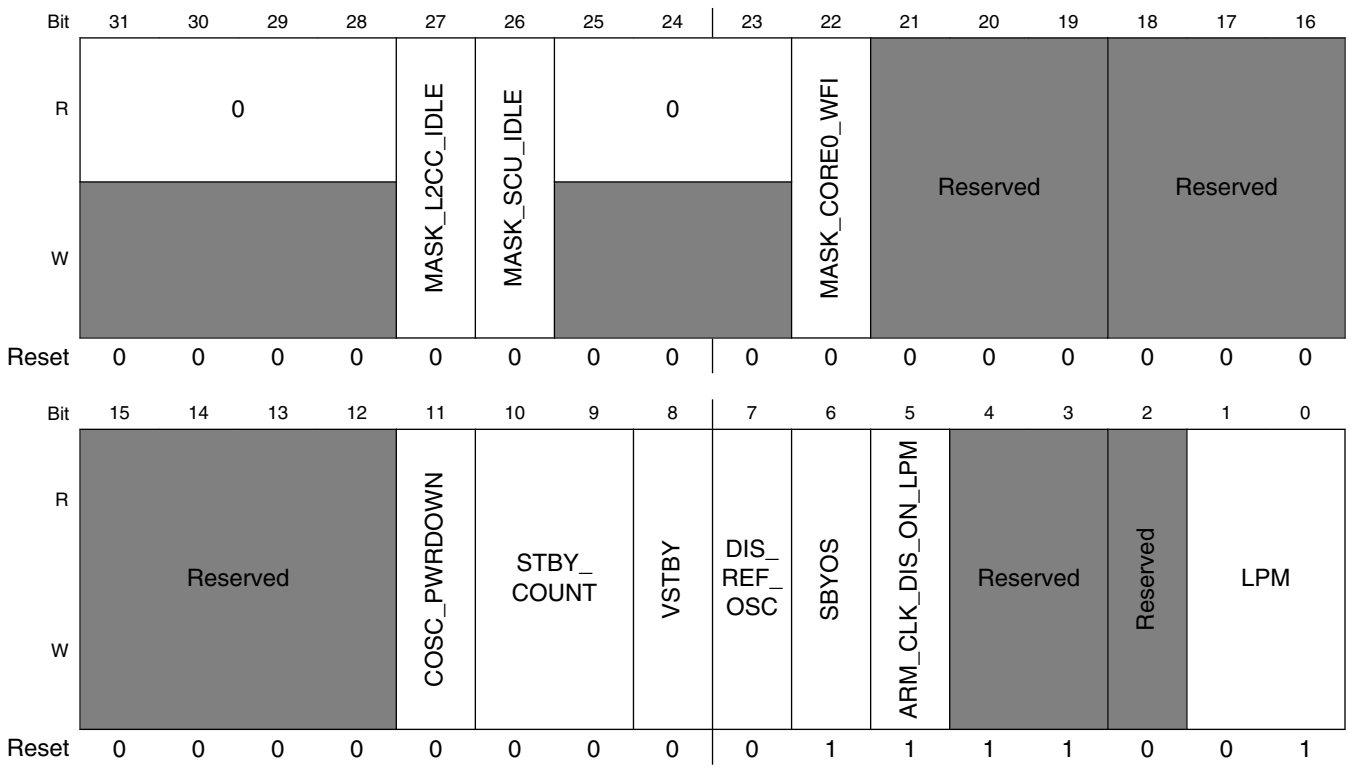
CCM_CDHIPR field descriptions (continued)

Field	Description
3 FLEXSPI_ PODF_BUSY	Busy indicator for flexspi_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 flexspi_podf 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 -	This field is reserved. Reserved

14.7.13 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
21–19 -	This field is reserved. Reserved
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. 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.

Table continues on the next page...

CCM_CLPCR field descriptions (continued)

Field	Description
	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

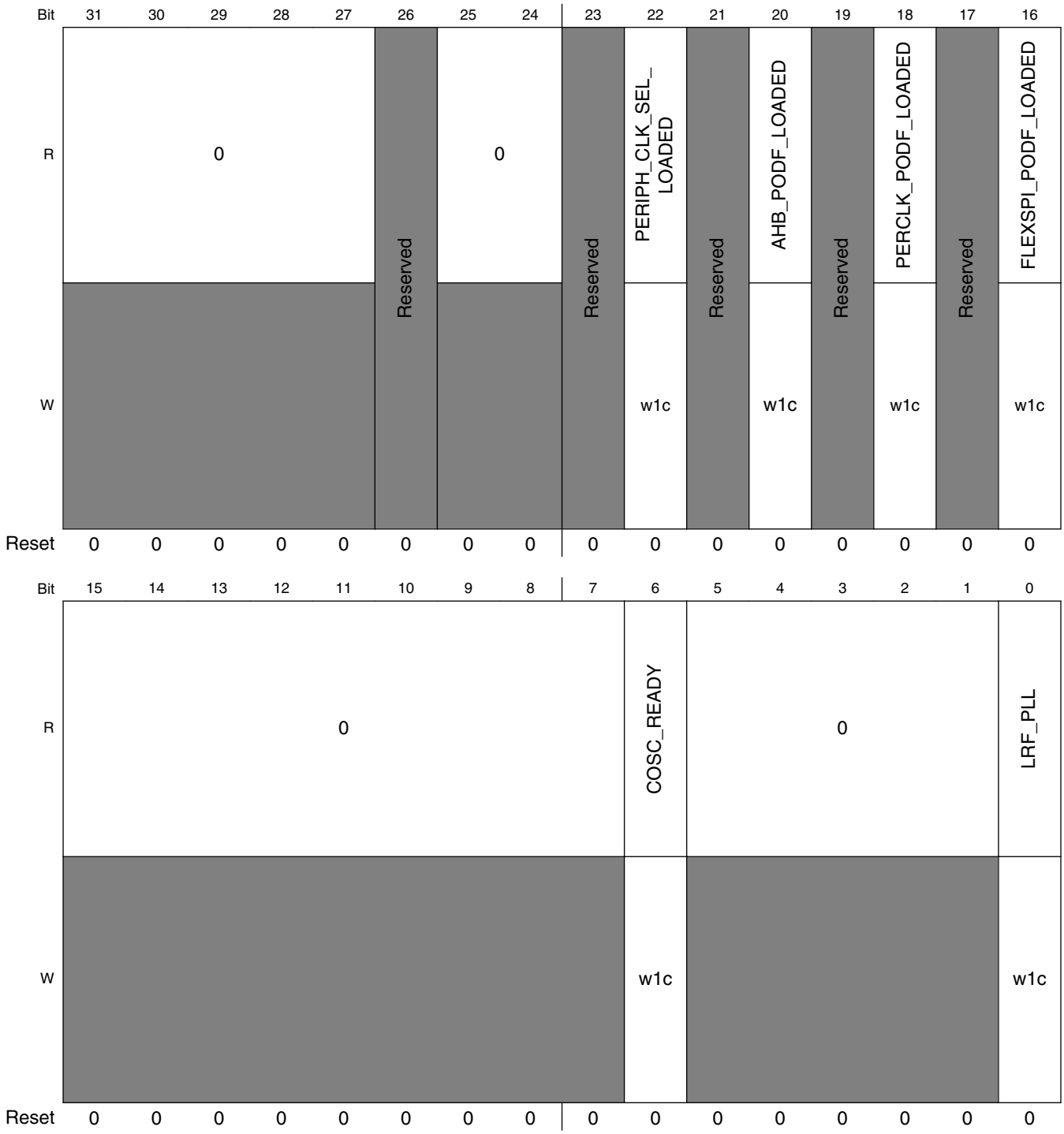
14.7.14 CCM Interrupt Status Register (CCM_CISR)

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...

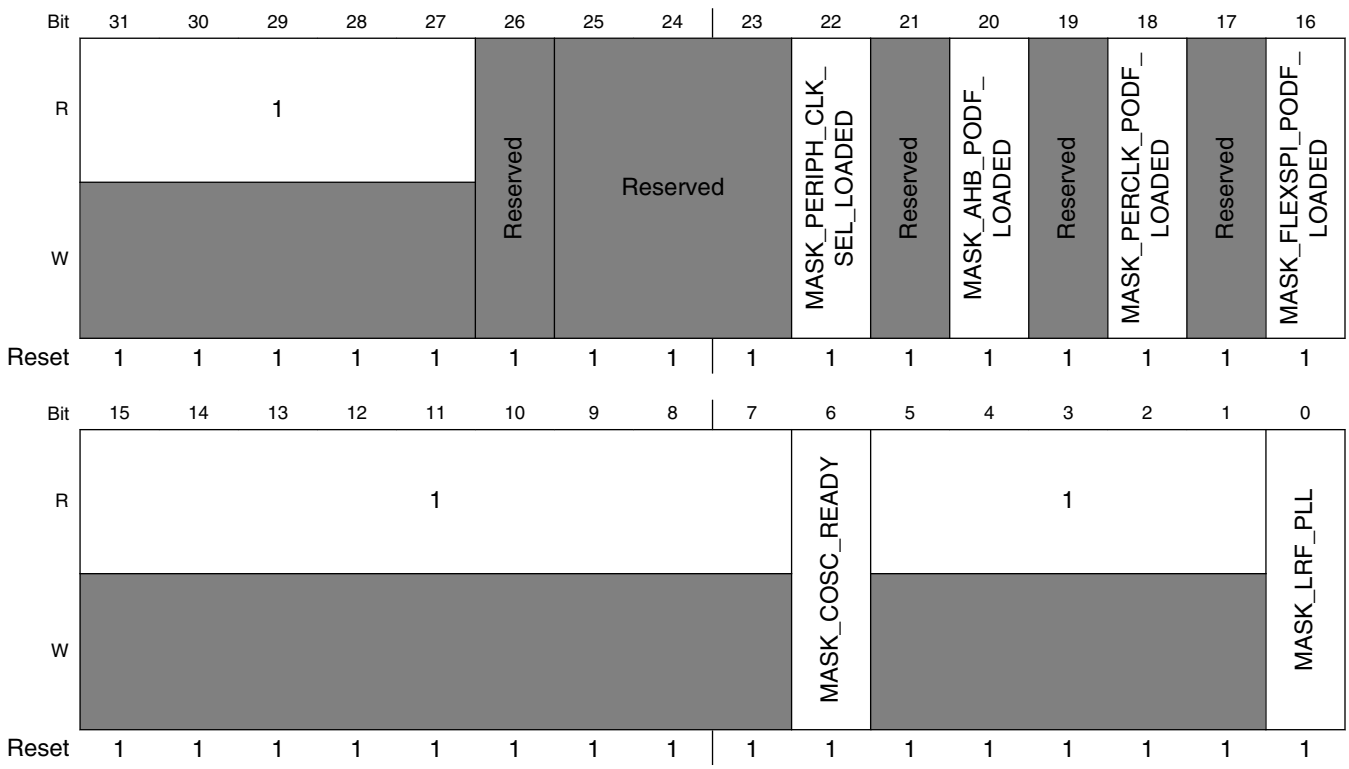
CCM_CISR field descriptions (continued)

Field	Description
26 -	This field is reserved. Reserved
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 -	This field is reserved. Reserved
18 PERCLK_PODF_ LOADED	CCM interrupt request 1 generated due to frequency change of perclk_podf 0 interrupt is not generated due to frequency change of perclk_podf 1 interrupt generated due to frequency change of perclk_podf
17 -	This field is reserved. Reserved
16 FLEXSPI_ PODF_LOADED	CCM interrupt request 1 generated due to frequency change of flexspi_podf 0 interrupt is not generated due to frequency change of flexspi_podf 1 interrupt generated due to frequency change of flexspi_podf
15–7 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

14.7.15 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 -	This field is reserved. Reserved
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...

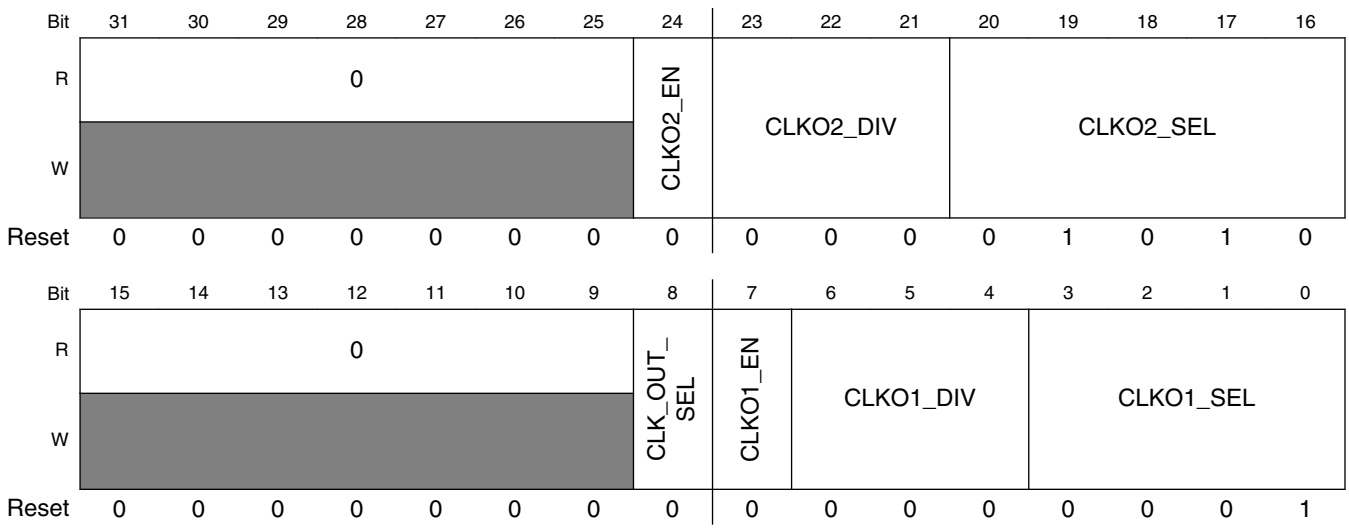
CCM_CIMR field descriptions (continued)

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 -	This field is reserved. Reserved
18 MASK_ PERCLK_PODF_ LOADED	mask interrupt generation due to update of perclk_podf 0 don't mask interrupt due to update of perclk_podf 1 mask interrupt due to update of perclk_podf
17 -	This field is reserved. Reserved
16 MASK_ FLEXSPI_ PODF_LOADED	mask interrupt generation due to update of flexspi_podf 0 don't mask interrupt due to update of flexspi_podf 1 mask interrupt due to update of flexspi_podf
15–7 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

14.7.16 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 00101 Reserved 00110 lpi2c_clk_root

Table continues on the next page...

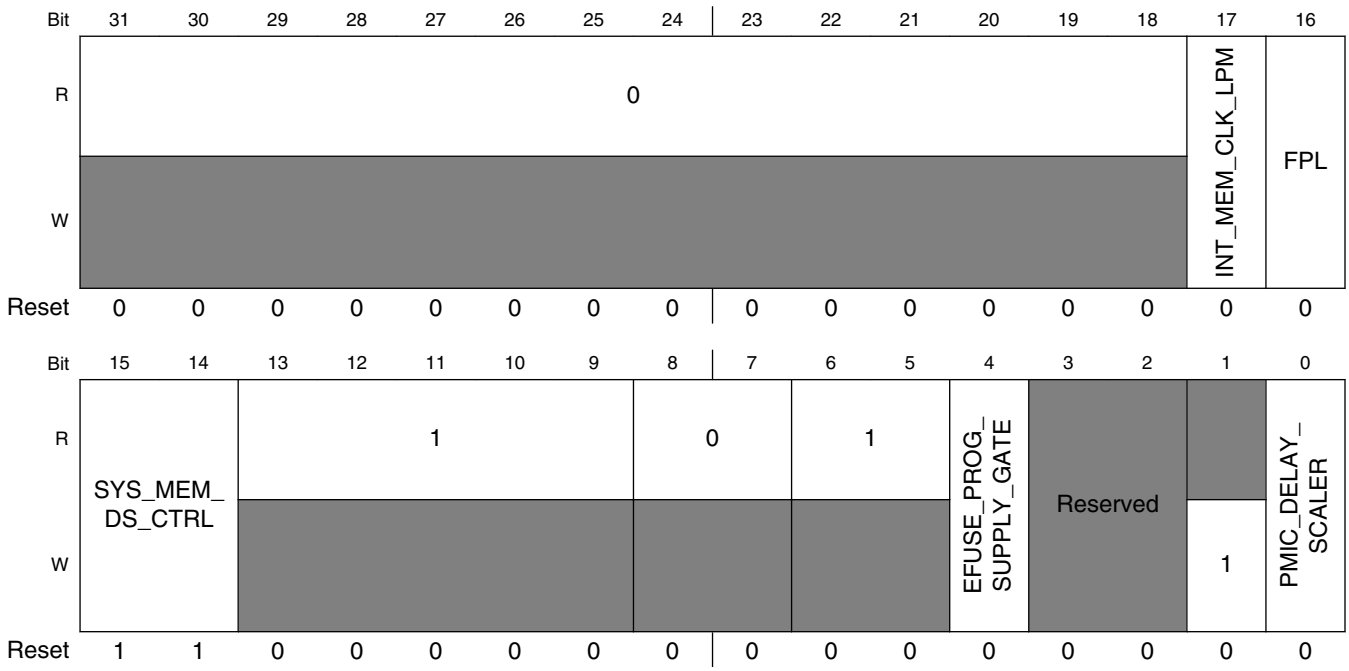
CCM_CCOSR field descriptions (continued)

Field	Description
	01110 osc_clk 10000 lpspi_clk_root 10010 sai1_clk_root 10100 sai3_clk_root 10110 trace_clk_root 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 pll3_sw_clk (divided by 2) 0001 PLL2 (divided by 2) 0010 ENET PLL (divided by 2) 0011 Reserved 0101 Reserved 0110 Reserved 1010 Reserved 1011 core_clk_root 1100 ipg_clk_root 1101 perclk_root 1110 Reserved 1111 pll4_main_clk

14.7.17 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

14.7.18 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	Reserved
19–18 CG9	Reserved
17–16 CG8	Reserved
15–14 CG7	Reserved
13–12 CG6	lpuart3 clock (lpuart3_clk_enable)
11–10 CG5	dcp clock (dcp_clk_enable)
9–8 CG4	sim_m_clk_r_clk_enable
7–6 CG3	flexspi_exsc clock (flexspi_exsc_clk_enable)
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)

14.7.19 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	Reserved
17–16 CG8	adc1 clock (adc1_clk_enable)
15–14 CG7	Reserved
13–12 CG6	pit clocks (pit_clk_enable)
11–10 CG5	Reserved
9–8 CG4	Reserved

Table continues on the next page...

CCM_CCGR1 field descriptions (continued)

Field	Description
7–6 CG3	Reserved
5–4 CG2	Reserved
3–2 CG1	lpspi2 clocks (lpspi2_clk_enable)
CG0	lpspi1 clocks (lpspi1_clk_enable)

14.7.20 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	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_CCGR2 field descriptions

Field	Description
31–30 CG15	Reserved
29–28 CG14	Reserved
27–26 CG13	Reserved
25–24 CG12	Reserved
23–22 CG11	xbar1 clock (xbar1_clk_enable)
21–20 CG10	Reserved
19–18 CG9	Reserved

Table continues on the next page...

CCM_CCGR2 field descriptions (continued)

Field	Description
17–16 CG8	Reserved
15–14 CG7	Reserved
13–12 CG6	OCOTP_CTRL clock (iim_clk_enable)
11–10 CG5	Reserved
9–8 CG4	lpi2c2 clock (lpi2c2_clk_enable)
7–6 CG3	lpi2c1 clock (lpi2c1_clk_enable)
5–4 CG2	iomuxc_snvs clock (iomuxc_snvs_clk_enable)
3–2 CG1	Reserved
CG0	ocram_exsc clock (ocram_exsc_clk_enable)

14.7.21 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	1	1	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. Reserved
27–26 CG13	Reserved
25–24 CG12	Reserved
23–22 CG11	Reserved
21–20 CG10	Reserved
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	Reserved
11–10 CG5	Reserved
9–8 CG4	aoi1 clock (aoi1_clk_enable)
7–6 CG3	Reserved
5–4 CG2	Reserved
3–2 CG1	Reserved
CG0	Reserved

14.7.22 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	dma_ps clocks (dma_ps_clk_enable)
29–28 CG14	Reserved
27–26 CG13	Reserved
25–24 CG12	Reserved
23–22 CG11	Reserved
21–20 CG10	Reserved
19–18 CG9	Reserved
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	Reserved
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	Reserved
5–4 CG2	iomuxc gpr clock (iomuxc_gpr_clk_enable)
3–2 CG1	iomuxc clock (iomuxc_clk_enable)
CG0	sim_m7_clk_r_enable

14.7.23 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	Reserved
25–24 CG12	lpuart1 clock (lpuart1_clk_enable)
23–22 CG11	sai3 clock (sai3_clk_enable)
21–20 CG10	Reserved
19–18 CG9	sai1 clock (sai1_clk_enable)

Table continues on the next page...

CCM_CCGR5 field descriptions (continued)

Field	Description
17–16 CG8	Reserved
15–14 CG7	spdif clock (spdif_clk_enable)
13–12 CG6	Reserved
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)

14.7.24 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	Reserved
29–28 CG14	Reserved

Table continues on the next page...

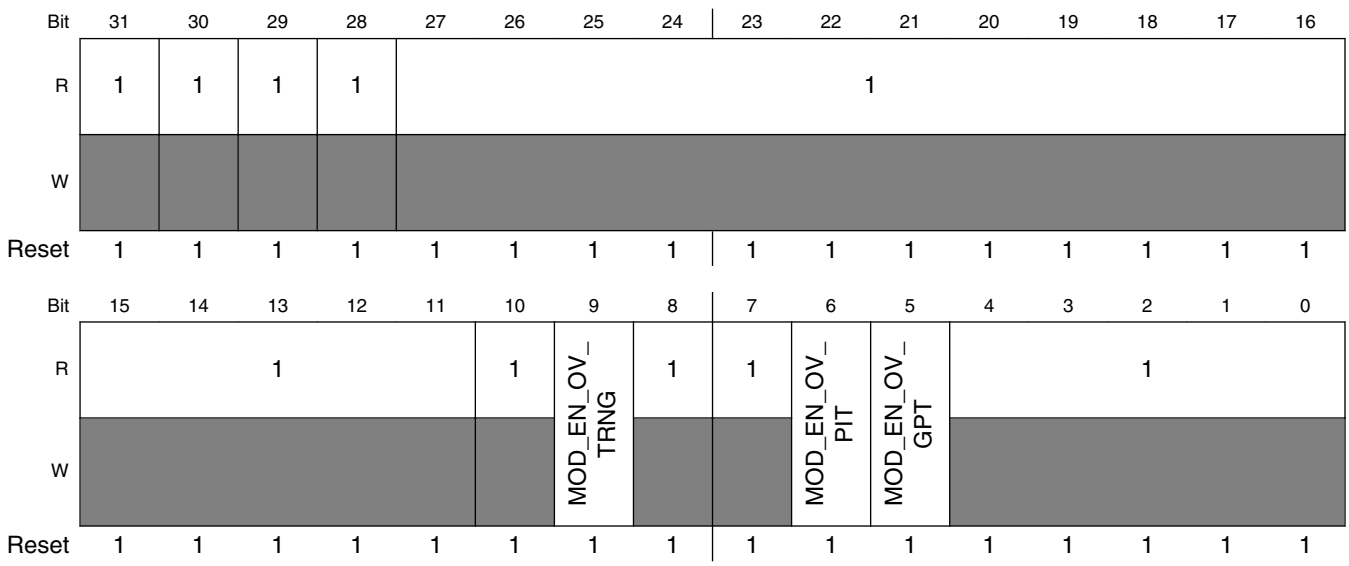
CCM_CCGR6 field descriptions (continued)

Field	Description
27–26 CG13	Reserved
25–24 CG12	Reserved
23–22 CG11	anadig clocks (anadig_clk_enable)
21–20 CG10	sim_per clock (sim_per_clk_enable)
19–18 CG9	Reserved
17–16 CG8	Reserved
15–14 CG7	Reserved
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	Reserved
7–6 CG3	dcdc clocks (dcdc_clk_enable)
5–4 CG2	Reserved
3–2 CG1	Reserved
CG0	usboh3 clock (usboh3_clk_enable)

14.7.25 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.
30 Reserved	This read-only field is reserved and always has the value 1.
29 Reserved	This read-only field is reserved and always has the value 1.
28 Reserved	This read-only field is reserved and always has the value 1.
27–11 Reserved	This read-only field is reserved and always has the value 1.
10 Reserved	This read-only field is reserved and always has the value 1.
9 MOD_EN_OV_TRNG	Overide clock enable signal from TRNG 0 don't override module enable signal 1 override module enable signal

Table continues on the next page...

CCM_CMEOR field descriptions (continued)

Field	Description
8 Reserved	This read-only field is reserved and always has the value 1.
7 Reserved	This read-only field is reserved and always has the value 1.
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:

- USB1 PLL: {0h010, 0h014, 0h018, 0h01C}, {0h0F0, 0h0F4, 0h0F8, 0h0FC}.
- System PLL: {0h030, 0h034, 0h038, 0h03C}, 0h040, 0h050, 0h060, {0h100, 0h104, 0h108, 0h10C}.
- Audio PLL : {0h070, 0h074, 0h078, 0h07C}, 0h080

CCM_ANALOG memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400D_8010	Analog USB1 480MHz PLL Control Register (CCM_ANALOG_PLL_USB1)	32	R/W	0001_2000h	14.8.1/585
400D_8014	Analog USB1 480MHz PLL Control Register (CCM_ANALOG_PLL_USB1_SET)	32	R/W	0001_2000h	14.8.1/585
400D_8018	Analog USB1 480MHz PLL Control Register (CCM_ANALOG_PLL_USB1_CLR)	32	R/W	0001_2000h	14.8.1/585
400D_801C	Analog USB1 480MHz PLL Control Register (CCM_ANALOG_PLL_USB1_TOG)	32	R/W	0001_2000h	14.8.1/585
400D_8030	Analog System PLL Control Register (CCM_ANALOG_PLL_SYS)	32	R/W	0001_3001h	14.8.2/587
400D_8034	Analog System PLL Control Register (CCM_ANALOG_PLL_SYS_SET)	32	R/W	0001_3001h	14.8.2/587

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_8038	Analog System PLL Control Register (CCM_ANALOG_PLL_SYS_CLR)	32	R/W	0001_3001h	14.8.2/587
400D_803C	Analog System PLL Control Register (CCM_ANALOG_PLL_SYS_TOG)	32	R/W	0001_3001h	14.8.2/587
400D_8040	528MHz System PLL Spread Spectrum Register (CCM_ANALOG_PLL_SYS_SS)	32	R/W	0000_0000h	14.8.3/588
400D_8050	Numerator of 528MHz System PLL Fractional Loop Divider Register (CCM_ANALOG_PLL_SYS_NUM)	32	R/W	0000_0000h	14.8.4/589
400D_8060	Denominator of 528MHz System PLL Fractional Loop Divider Register (CCM_ANALOG_PLL_SYS_DENOM)	32	R/W	0000_0012h	14.8.5/589
400D_8070	Analog Audio PLL control Register (CCM_ANALOG_PLL_AUDIO)	32	R/W	0001_1006h	14.8.6/591
400D_8074	Analog Audio PLL control Register (CCM_ANALOG_PLL_AUDIO_SET)	32	R/W	0001_1006h	14.8.6/591
400D_8078	Analog Audio PLL control Register (CCM_ANALOG_PLL_AUDIO_CLR)	32	R/W	0001_1006h	14.8.6/591
400D_807C	Analog Audio PLL control Register (CCM_ANALOG_PLL_AUDIO_TOG)	32	R/W	0001_1006h	14.8.6/591
400D_8080	Numerator of Audio PLL Fractional Loop Divider Register (CCM_ANALOG_PLL_AUDIO_NUM)	32	R/W	05F5_E100h	14.8.7/593
400D_8090	Denominator of Audio PLL Fractional Loop Divider Register (CCM_ANALOG_PLL_AUDIO_DENOM)	32	R/W	2964_619Ch	14.8.8/593
400D_80E0	Analog ENET PLL Control Register (CCM_ANALOG_PLL_ENET)	32	R/W	0001_1001h	14.8.9/595
400D_80E4	Analog ENET PLL Control Register (CCM_ANALOG_PLL_ENET_SET)	32	R/W	0001_1001h	14.8.9/595
400D_80E8	Analog ENET PLL Control Register (CCM_ANALOG_PLL_ENET_CLR)	32	R/W	0001_1001h	14.8.9/595
400D_80EC	Analog ENET PLL Control Register (CCM_ANALOG_PLL_ENET_TOG)	32	R/W	0001_1001h	14.8.9/595
400D_80F0	480MHz Clock (PLL3) Phase Fractional Divider Control Register (CCM_ANALOG_PFD_480)	32	R/W	1311_100Ch	14.8.10/ 597
400D_80F4	480MHz Clock (PLL3) Phase Fractional Divider Control Register (CCM_ANALOG_PFD_480_SET)	32	R/W	1311_100Ch	14.8.10/ 597
400D_80F8	480MHz Clock (PLL3) Phase Fractional Divider Control Register (CCM_ANALOG_PFD_480_CLR)	32	R/W	1311_100Ch	14.8.10/ 597
400D_80FC	480MHz Clock (PLL3) Phase Fractional Divider Control Register (CCM_ANALOG_PFD_480_TOG)	32	R/W	1311_100Ch	14.8.10/ 597
400D_8100	528MHz Clock (PLL2) Phase Fractional Divider Control Register (CCM_ANALOG_PFD_528)	32	R/W	1018_101Bh	14.8.11/ 599
400D_8104	528MHz Clock (PLL2) Phase Fractional Divider Control Register (CCM_ANALOG_PFD_528_SET)	32	R/W	1018_101Bh	14.8.11/ 599
400D_8108	528MHz Clock (PLL2) Phase Fractional Divider Control Register (CCM_ANALOG_PFD_528_CLR)	32	R/W	1018_101Bh	14.8.11/ 599

Table continues on the next page...

CCM_ANALOG memory map (continued)

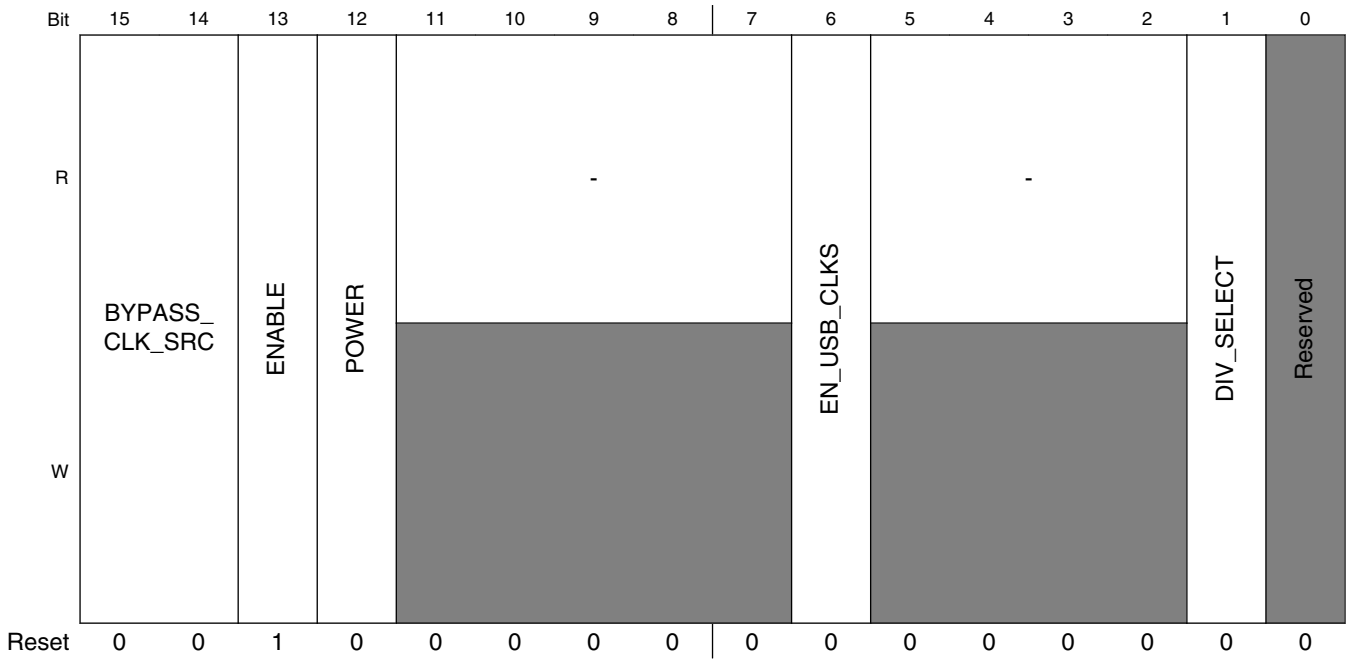
Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400D_810C	528MHz Clock (PLL2) Phase Fractional Divider Control Register (CCM_ANALOG_PFD_528_TOG)	32	R/W	1018_101Bh	14.8.11/ 599
400D_8150	Miscellaneous Register 0 (CCM_ANALOG_MISC0)	32	R/W	0400_0000h	14.8.12/ 602
400D_8154	Miscellaneous Register 0 (CCM_ANALOG_MISC0_SET)	32	R/W	0400_0000h	14.8.12/ 602
400D_8158	Miscellaneous Register 0 (CCM_ANALOG_MISC0_CLR)	32	R/W	0400_0000h	14.8.12/ 602
400D_815C	Miscellaneous Register 0 (CCM_ANALOG_MISC0_TOG)	32	R/W	0400_0000h	14.8.12/ 602
400D_8160	Miscellaneous Register 1 (CCM_ANALOG_MISC1)	32	R/W	0000_0000h	14.8.13/ 606
400D_8164	Miscellaneous Register 1 (CCM_ANALOG_MISC1_SET)	32	R/W	0000_0000h	14.8.13/ 606
400D_8168	Miscellaneous Register 1 (CCM_ANALOG_MISC1_CLR)	32	R/W	0000_0000h	14.8.13/ 606
400D_816C	Miscellaneous Register 1 (CCM_ANALOG_MISC1_TOG)	32	R/W	0000_0000h	14.8.13/ 606
400D_8170	Miscellaneous Register 2 (CCM_ANALOG_MISC2)	32	R/W	0027_2727h	14.8.14/ 608
400D_8174	Miscellaneous Register 2 (CCM_ANALOG_MISC2_SET)	32	R/W	0027_2727h	14.8.14/ 608
400D_8178	Miscellaneous Register 2 (CCM_ANALOG_MISC2_CLR)	32	R/W	0027_2727h	14.8.14/ 608
400D_817C	Miscellaneous Register 2 (CCM_ANALOG_MISC2_TOG)	32	R/W	0027_2727h	14.8.14/ 608

14.8.1 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



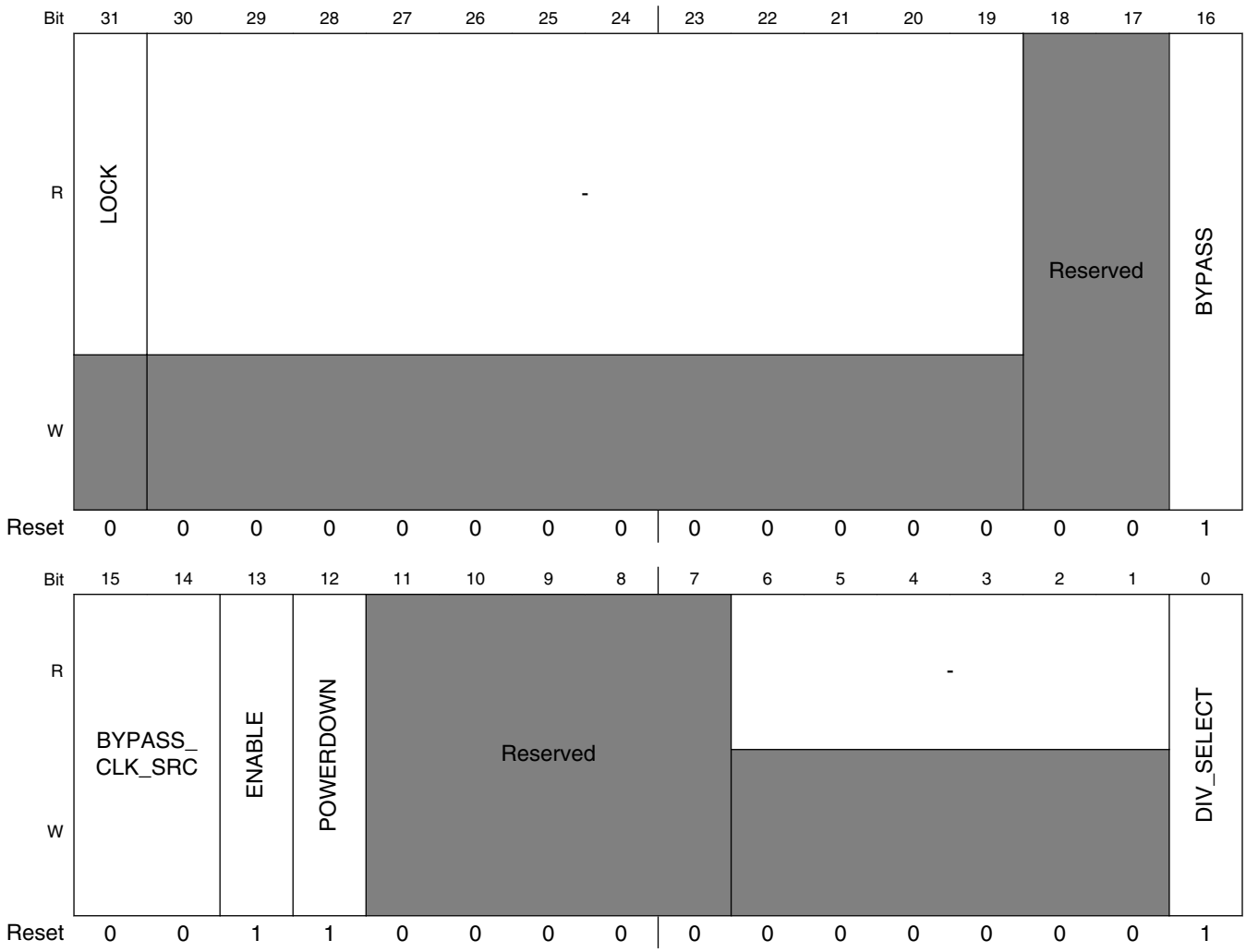
CCM_ANALOG_PLL_USB1*n* field descriptions

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.
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.
0 -	This field is reserved. Reserved.

14.8.2 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

Table continues on the next page...

CCM_ANALOG_PLL_SYS*n* field descriptions (continued)

Field	Description
16 BYPASS	Bypass the PLL.
15–14 BYPASS_CLK_SRC	Determines the bypass source. 0x0 **REF_CLK_24M** — Select the 24MHz oscillator 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.

14.8.3 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.

Table continues on the next page...

CCM_ANALOG_PLL_SYS_SS field descriptions (continued)

Field	Description
15 ENABLE	Enable bit 0 — Spread spectrum modulation disabled 1 — Soread spectrum modulation enabled
STEP	Frequency change step = step/CCM_ANALOG_PLL_SYS_DENOM[B]*24MHz.

14.8.4 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).

14.8.5 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

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	0	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	0	1	0

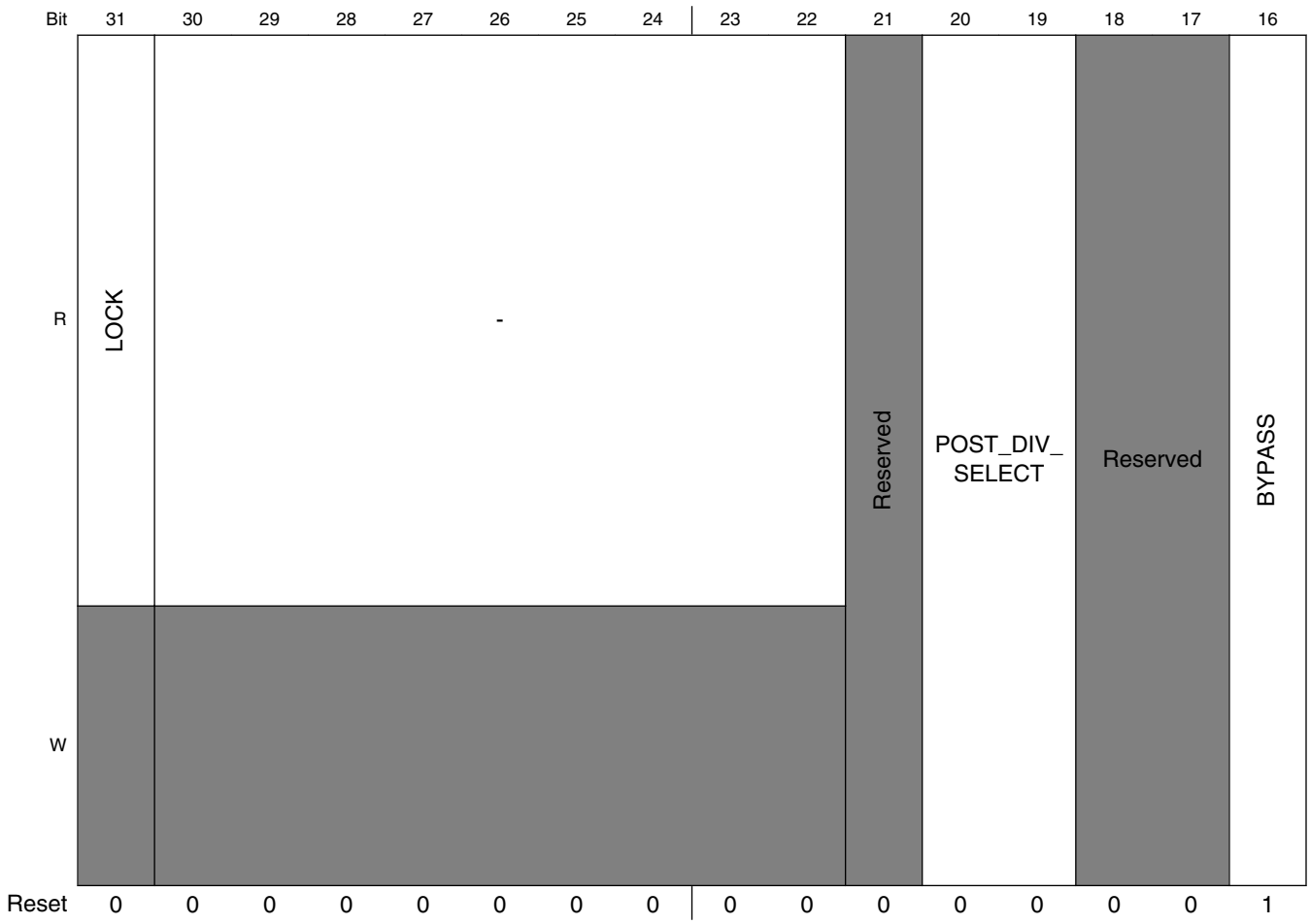
CCM_ANALOG_PLL_SYS_DENOM field descriptions

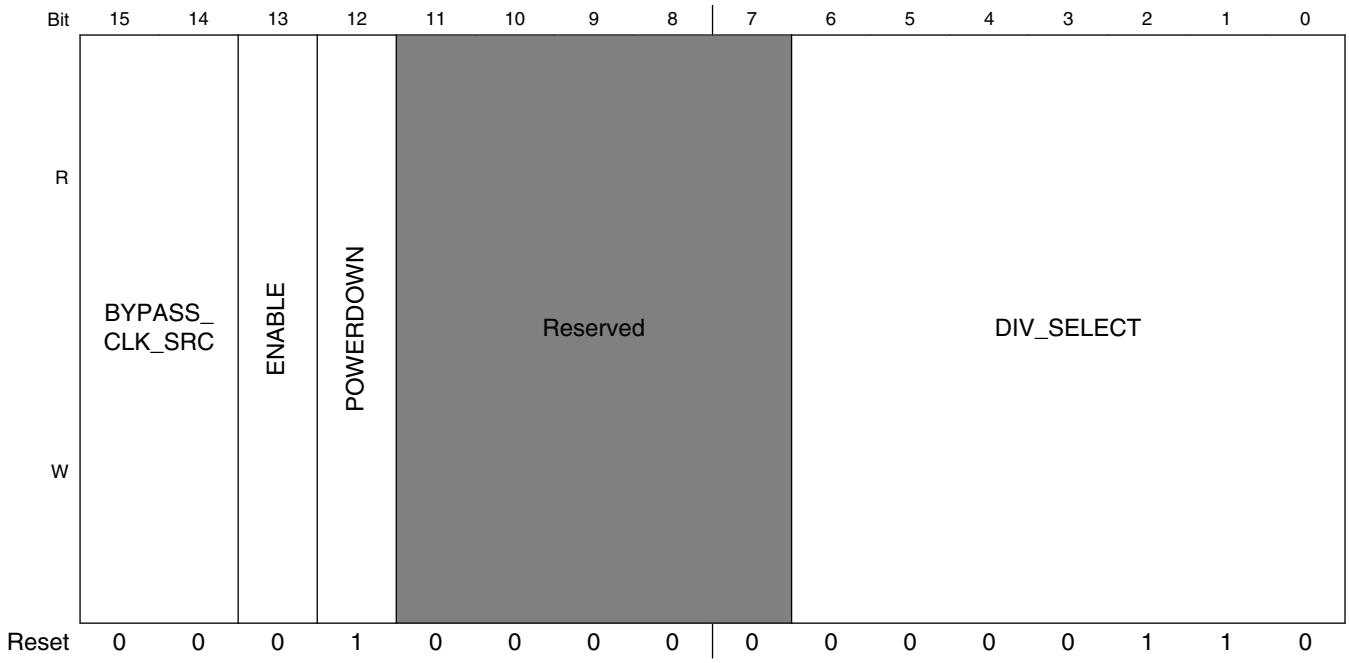
Field	Description
31–30 -	Always set to zero (0).
B	30 bit denominator (B) of fractional loop divider (unsigned integer).

14.8.6 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. 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)

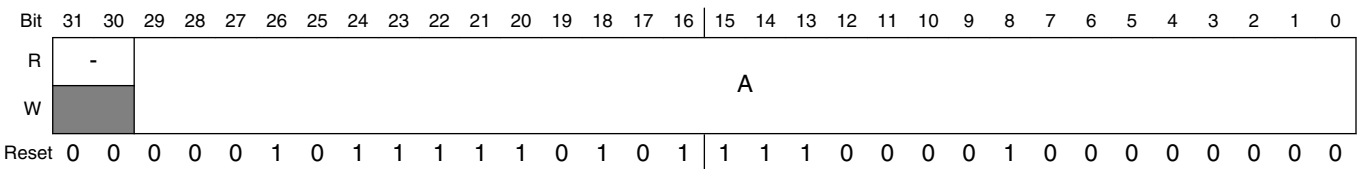
Field	Description
11–7 -	This field is reserved. Reserved.
DIV_SELECT	This field controls the PLL loop divider. Valid range for DIV_SELECT divider value: 27~54.

14.8.7 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



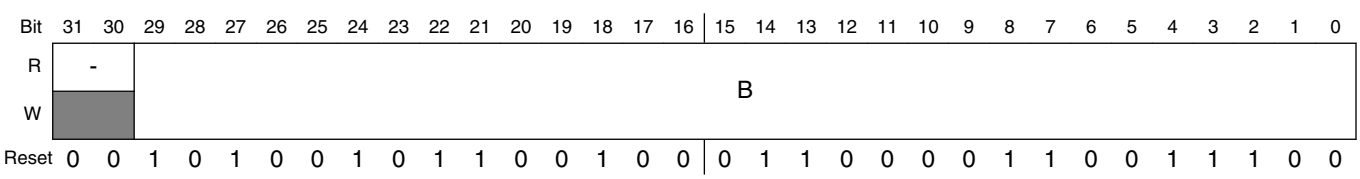
CCM_ANALOG_PLL_AUDIO_NUM field descriptions

Field	Description
31–30 -	Always set to zero (0).
A	30 bit numerator of fractional loop divider.

14.8.8 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



CCM_ANALOG_PLL_AUDIO_DENOM field descriptions

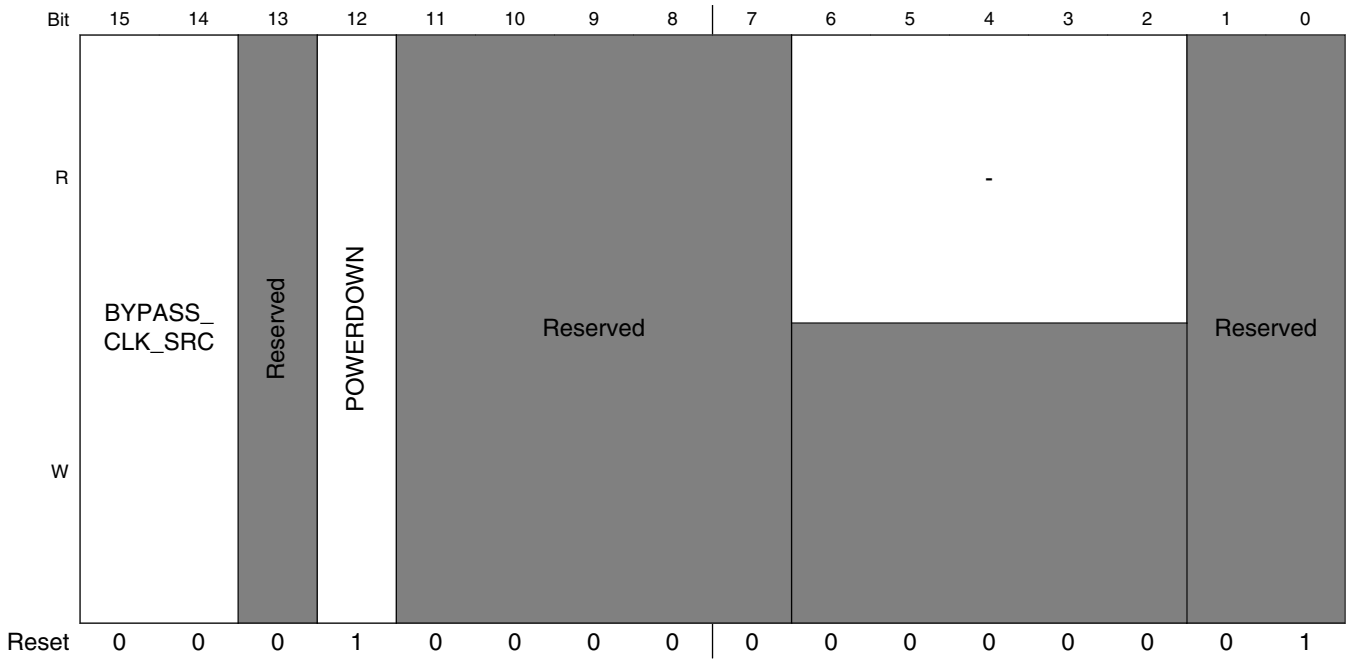
Field	Description
31–30 -	Always set to zero (0).
B	30 bit denominator of fractional loop divider.

14.8.9 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

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	LOCK	-								ENET_500M_REF_EN	Reserved	Reserved		Reserved		BYPASS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1



CCM_ANALOG_PLL_ENET*n* field descriptions

Field	Description
31 LOCK	1 - PLL is currently locked; 0 - PLL is not currently locked.
30–23 -	Always set to zero (0).
22 ENET_500M_ REF_EN	Enable the PLL providing ENET 500 MHz reference clock
21 -	This field is reserved. Reserved
20–19 -	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. 0x2 **Reserved1** — 0x3 **Reserved2** —
13 -	This field is reserved. Reserved.
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
6–2 -	Always set to zero (0).
-	This field is reserved. Reserved.

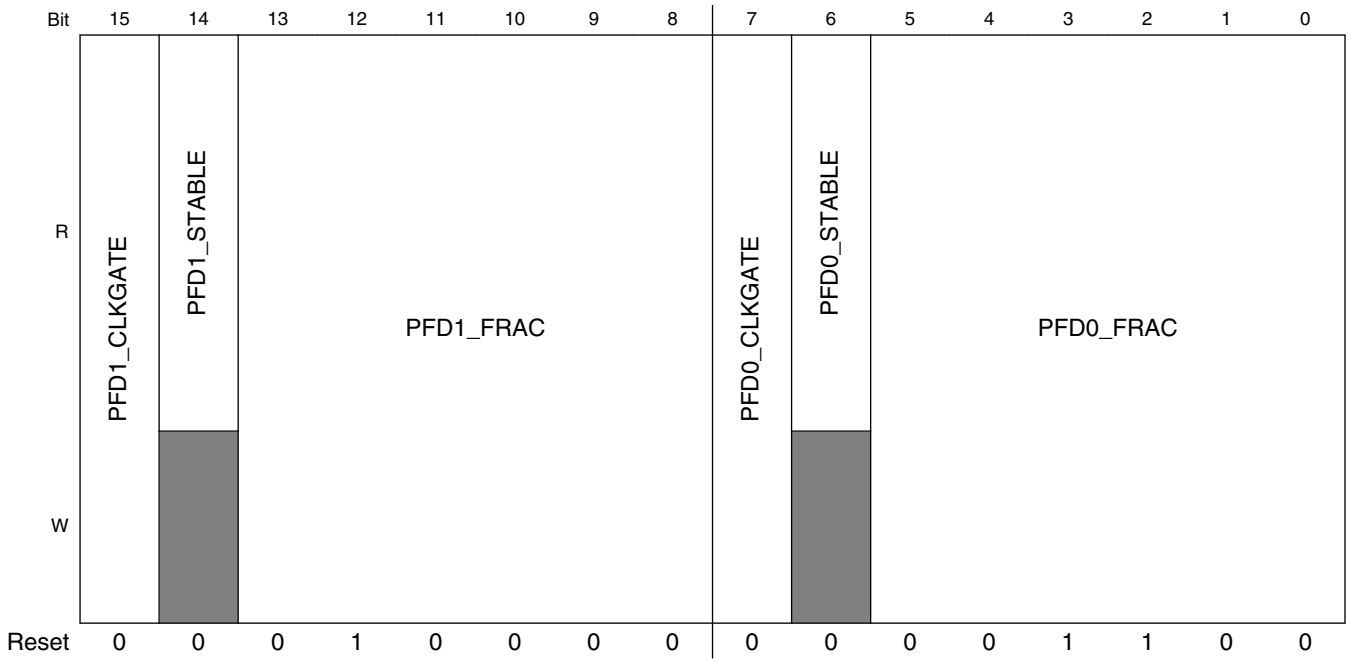
14.8.10 480MHz Clock (PLL3) Phase Fractional Divider Control Register (CCM_ANALOG_PFD_480*n*)

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

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	PFD3_CLKGATE	PFD3_STABLE	PFD3_FRAC						PFD2_CLKGATE	PFD2_STABLE	PFD2_FRAC					
W																
Reset	0	0	0	1	0	0	1	1	0	0	0	1	0	0	0	1



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)

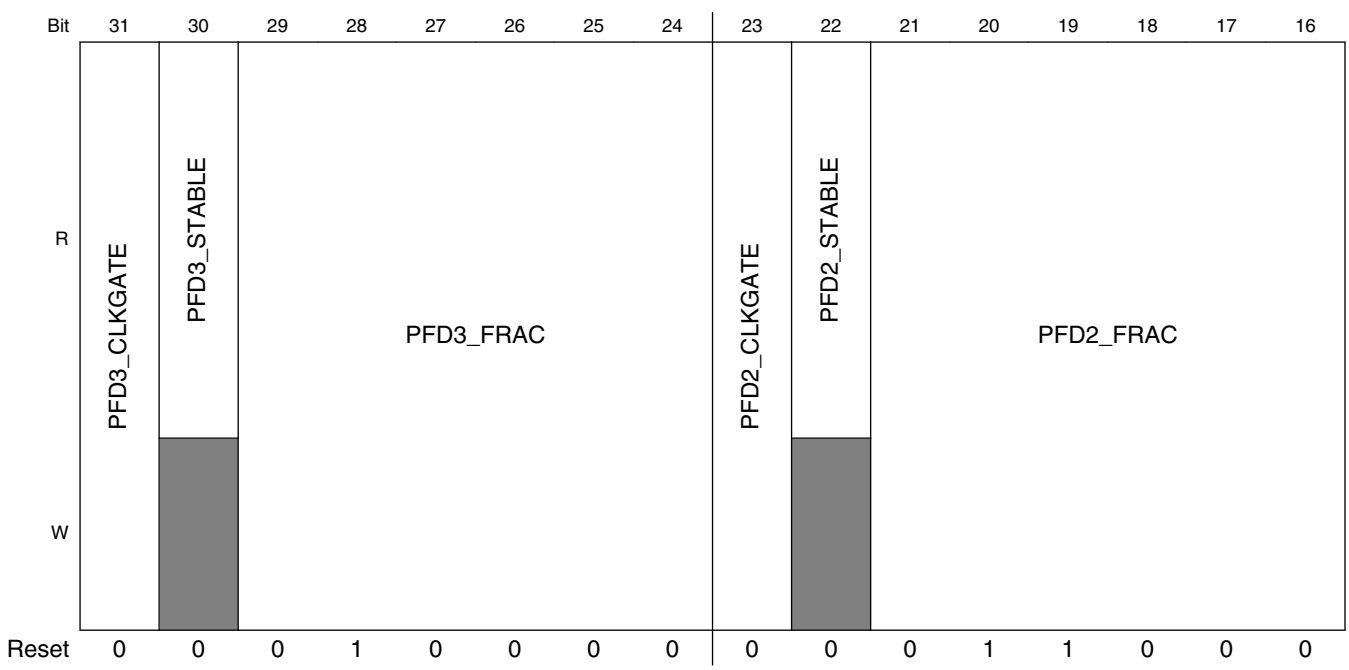
Field	Description
	program the new value, and when this bit inverts, the phase divider clock output is stable. Note that the value will not invert when the fractional divider is taken out of or placed into clock-gated state.
13–8 PFD1_FRAC	This field controls the fractional divide value. The resulting 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. 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.

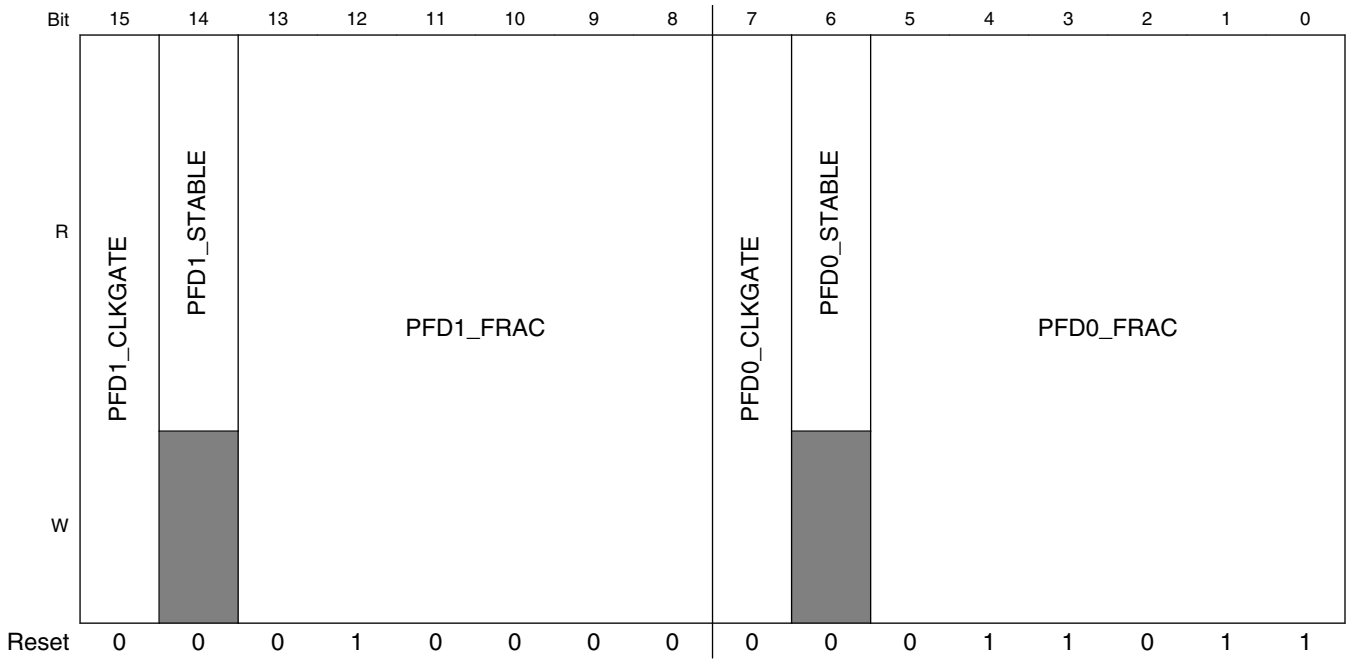
14.8.11 528MHz Clock (PLL2) Phase Fractional Divider Control Register (CCM_ANALOG_PFD_528*n*)

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)

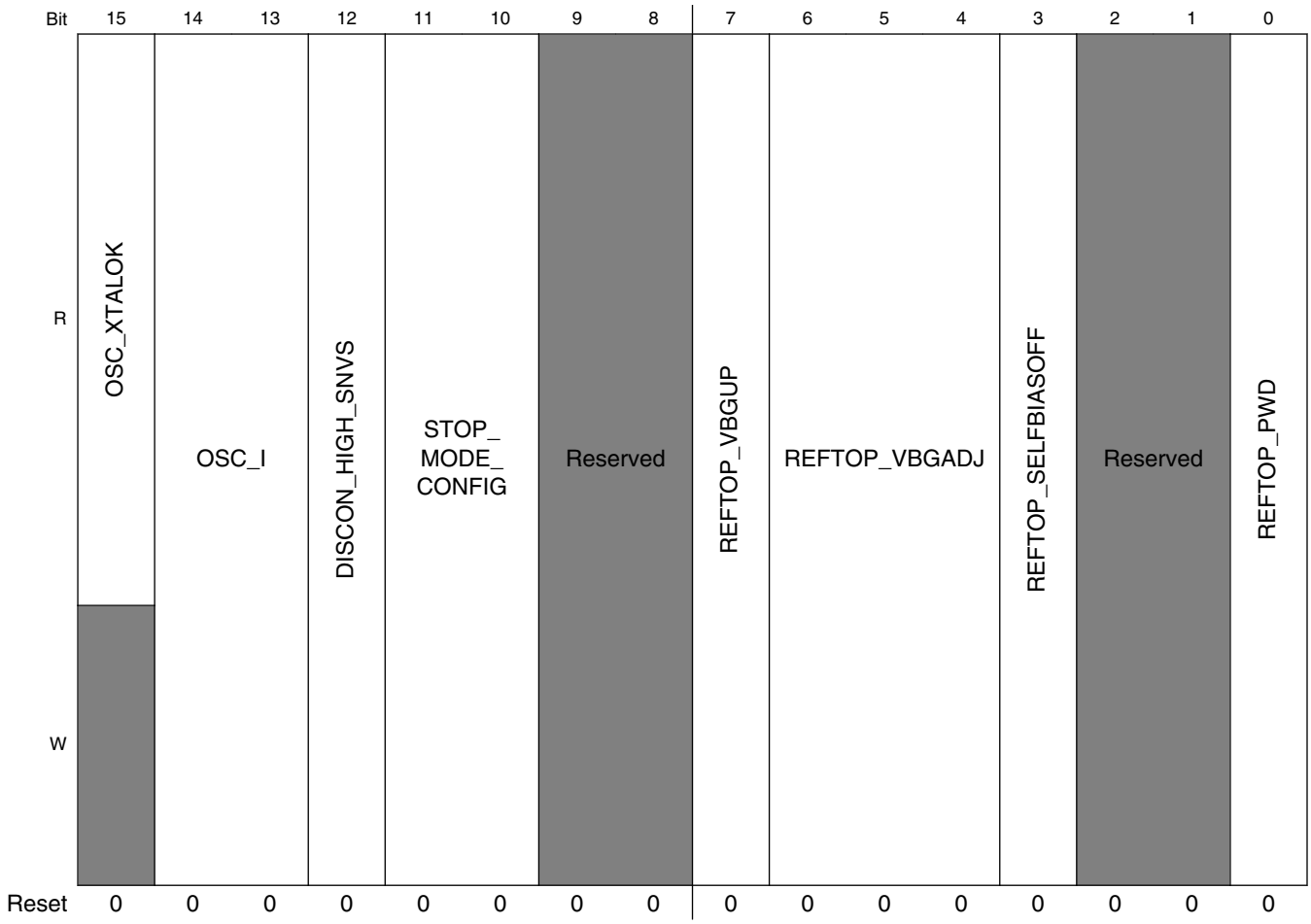
Field	Description
	program the new value, and when this bit inverts, the phase divider clock output is stable. Note that the value will not invert when the fractional divider is taken out of or placed into clock-gated state.
13–8 PFD1_FRAC	This field controls the fractional divide value. The resulting 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.

14.8.12 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

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved	XTAL_24M_PWD	RTC_XTAL_SOURCE	CLKGATE_DELAY			CLKGATE_CTRL	Reserved								OSC_XTALOK_EN
Reset	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0



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 -	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 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 -	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)

14.8.13 Miscellaneous Register 1 (CCM_ANALOG_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

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	Reserved	Reserved	Reserved	Reserved									
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 -	This field is reserved. Reserved
11 -	This field is reserved. Reserved
10 -	This field is reserved. Reserved
9–5 -	This field is reserved. Reserved
-	This field is reserved. Reserved

14.8.14 Miscellaneous Register 2 (CCM_ANALOG_MISC2*n*)

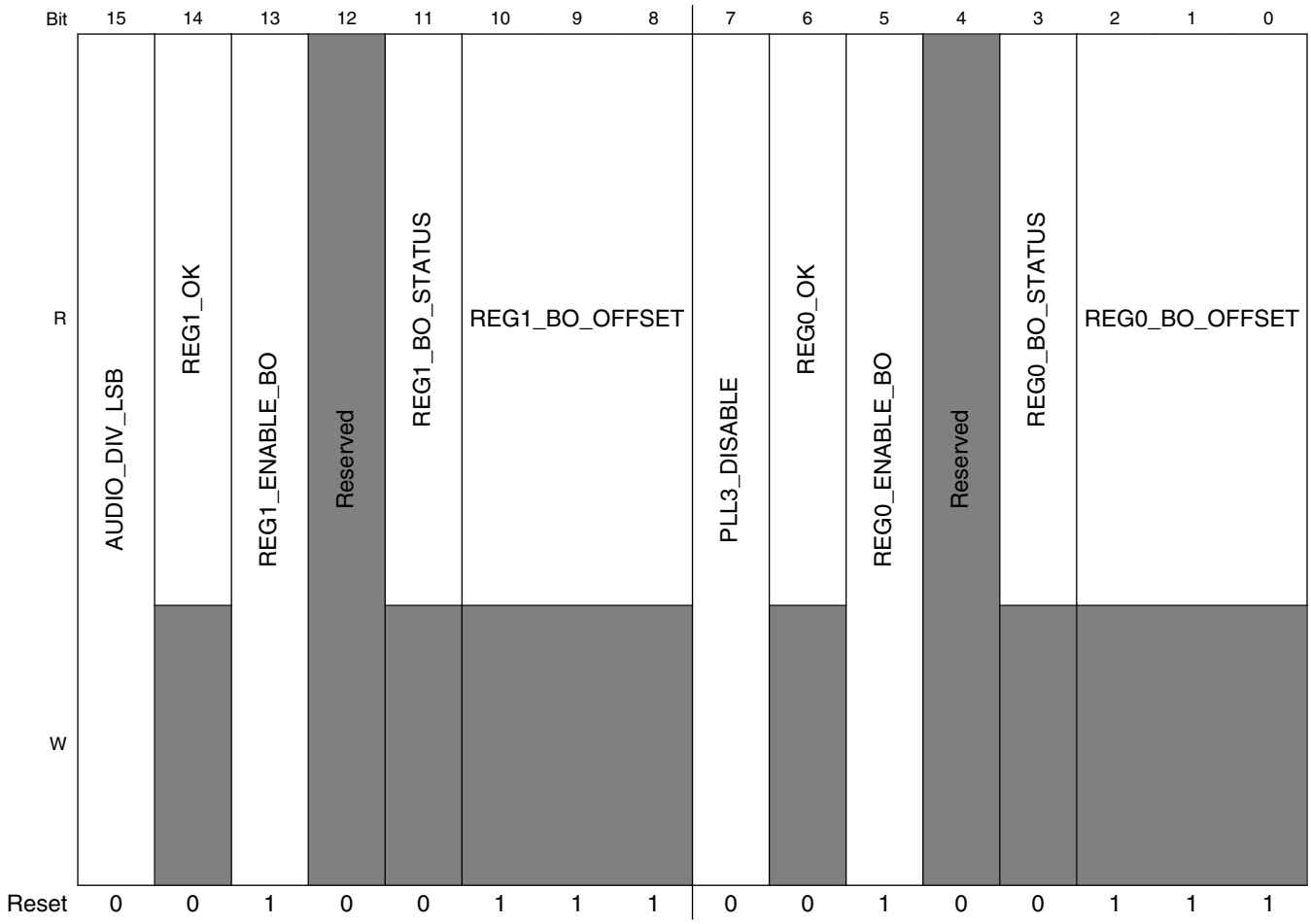
This register defines the control for miscellaneous analog blocks.

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	Reserved		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 -	This field is reserved.
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). **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

Table continues on the next page...

CCM_ANALOG_MISC2*n* field descriptions (continued)

Field	Description
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
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)

Table continues on the next page...

CCM_ANALOG_MISC2*n* field descriptions (continued)

Field	Description
	01 divide by 2 10 divide by 1 11 divide by 4
14 REG1_OK	GPU 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)
5 REG0_ENABLE_BO	Enables the brownout detection. **NOTE:** Not related to CCM. See Power Management Unit (PMU)
4 -	This field is reserved.

Table continues on the next page...

CCM_ANALOG_MISC2*n* field descriptions (continued)

Field	Description
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_07	ALT6	O
REF_CLK_24M	24 MHz reference clock	GPIO_AD_14	ALT6	O
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.

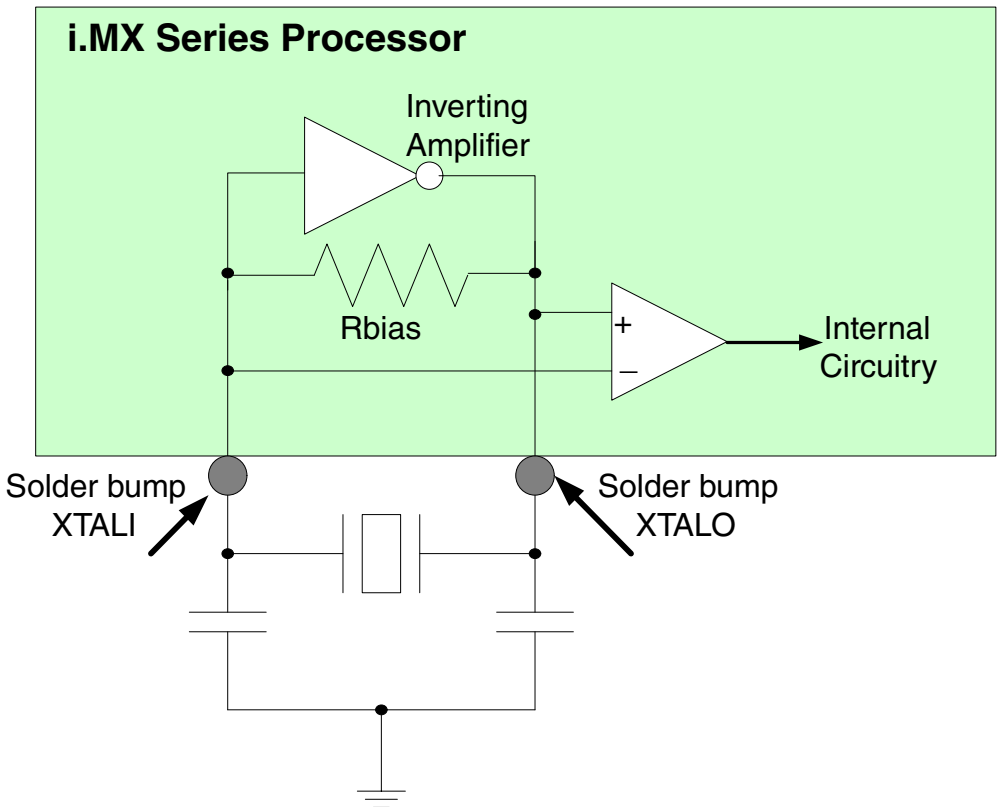


Figure 15-1. Oscillator Configuration (24 MHz)

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.

For more details on the possible usage of this module please contact a NXP FAE for pertinent application-notes.

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.

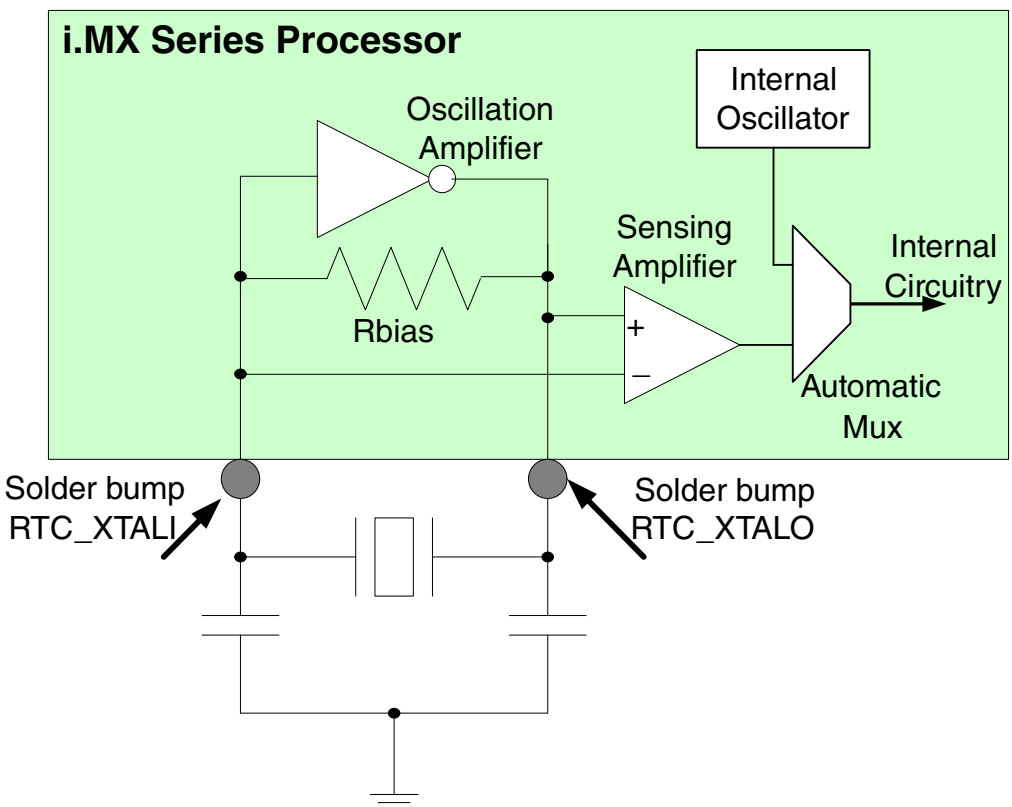


Figure 15-2. Oscillator Configuration (32 kHz)

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. An external 32 kHz clock source must be used for production systems.

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	15.6.1/620
400D_8154	Miscellaneous Register 0 (XTALOSC24M_MISC0_SET)	32	R/W	0400_0000h	15.6.1/620
400D_8158	Miscellaneous Register 0 (XTALOSC24M_MISC0_CLR)	32	R/W	0400_0000h	15.6.1/620

Table continues on the next page...

XTALOSC24M memory map (continued)

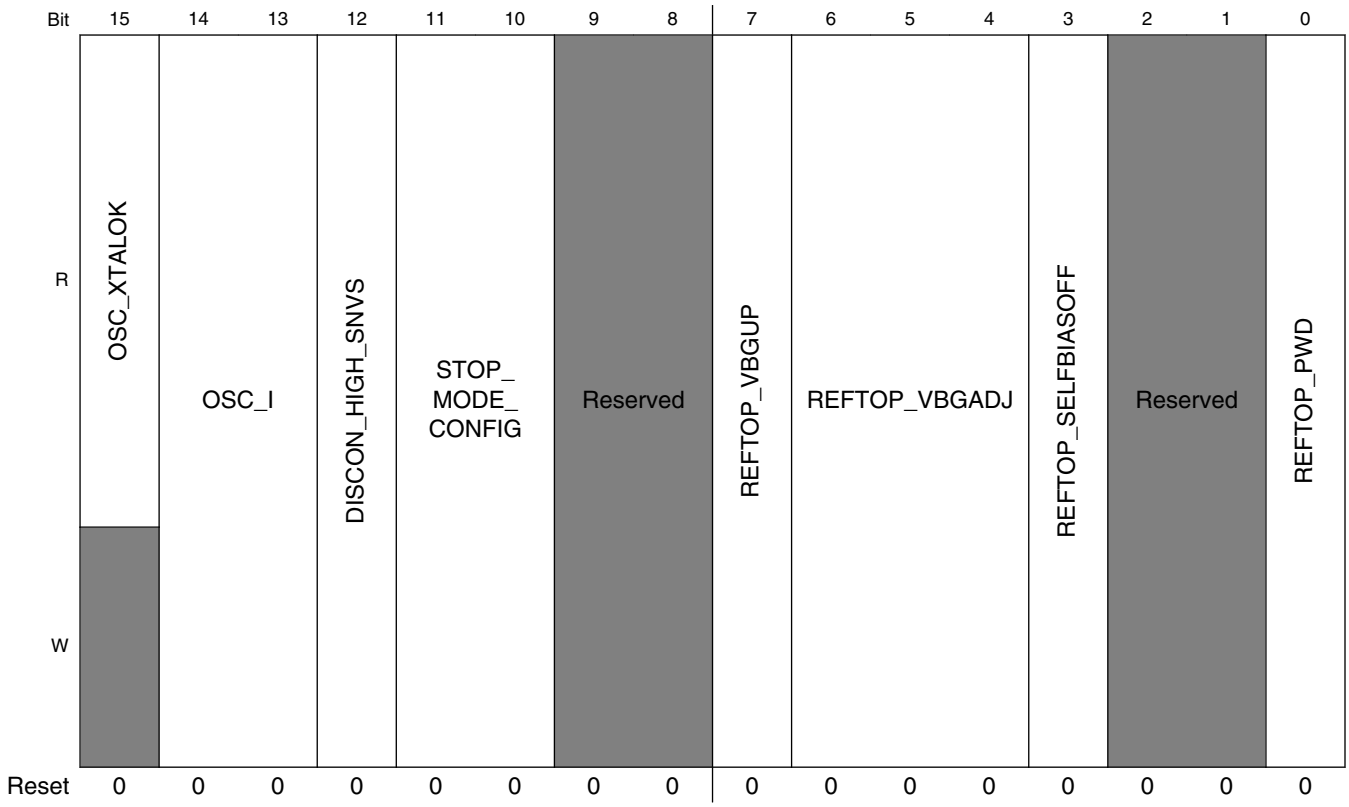
Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400D_815C	Miscellaneous Register 0 (XTALOSC24M_MISC0_TOG)	32	R/W	0400_0000h	15.6.1/620
400D_8270	XTAL OSC (LP) Control Register (XTALOSC24M_LOWPWR_CTRL)	32	R/W	See section	15.6.2/624
400D_8274	XTAL OSC (LP) Control Register (XTALOSC24M_LOWPWR_CTRL_SET)	32	R/W	See section	15.6.2/624
400D_8278	XTAL OSC (LP) Control Register (XTALOSC24M_LOWPWR_CTRL_CLR)	32	R/W	See section	15.6.2/624
400D_827C	XTAL OSC (LP) Control Register (XTALOSC24M_LOWPWR_CTRL_TOG)	32	R/W	See section	15.6.2/624
400D_82A0	XTAL OSC Configuration 0 Register (XTALOSC24M_OSC_CONFIG0)	32	R/W	0000_1020h	15.6.3/627
400D_82A4	XTAL OSC Configuration 0 Register (XTALOSC24M_OSC_CONFIG0_SET)	32	R/W	0000_1020h	15.6.3/627
400D_82A8	XTAL OSC Configuration 0 Register (XTALOSC24M_OSC_CONFIG0_CLR)	32	R/W	0000_1020h	15.6.3/627
400D_82AC	XTAL OSC Configuration 0 Register (XTALOSC24M_OSC_CONFIG0_TOG)	32	R/W	0000_1020h	15.6.3/627
400D_82B0	XTAL OSC Configuration 1 Register (XTALOSC24M_OSC_CONFIG1)	32	R/W	0000_02EEh	15.6.4/628
400D_82B4	XTAL OSC Configuration 1 Register (XTALOSC24M_OSC_CONFIG1_SET)	32	R/W	0000_02EEh	15.6.4/628
400D_82B8	XTAL OSC Configuration 1 Register (XTALOSC24M_OSC_CONFIG1_CLR)	32	R/W	0000_02EEh	15.6.4/628
400D_82BC	XTAL OSC Configuration 1 Register (XTALOSC24M_OSC_CONFIG1_TOG)	32	R/W	0000_02EEh	15.6.4/628
400D_82C0	XTAL OSC Configuration 2 Register (XTALOSC24M_OSC_CONFIG2)	32	R/W	0001_02E2h	15.6.5/629
400D_82C4	XTAL OSC Configuration 2 Register (XTALOSC24M_OSC_CONFIG2_SET)	32	R/W	0001_02E2h	15.6.5/629
400D_82C8	XTAL OSC Configuration 2 Register (XTALOSC24M_OSC_CONFIG2_CLR)	32	R/W	0001_02E2h	15.6.5/629
400D_82CC	XTAL OSC Configuration 2 Register (XTALOSC24M_OSC_CONFIG2_TOG)	32	R/W	0001_02E2h	15.6.5/629

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...

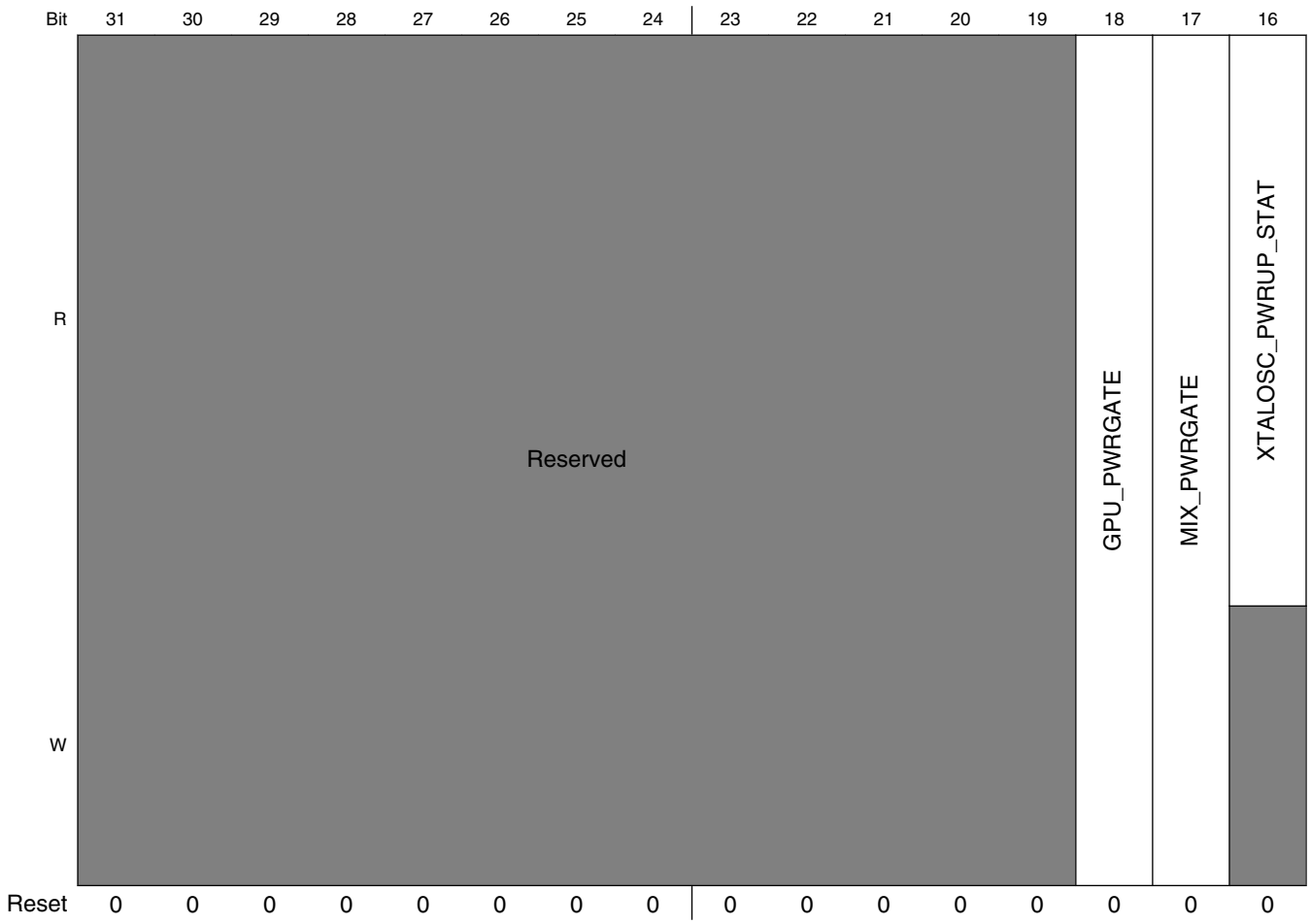
XTALOSC24M_MISC0*n* field descriptions (continued)

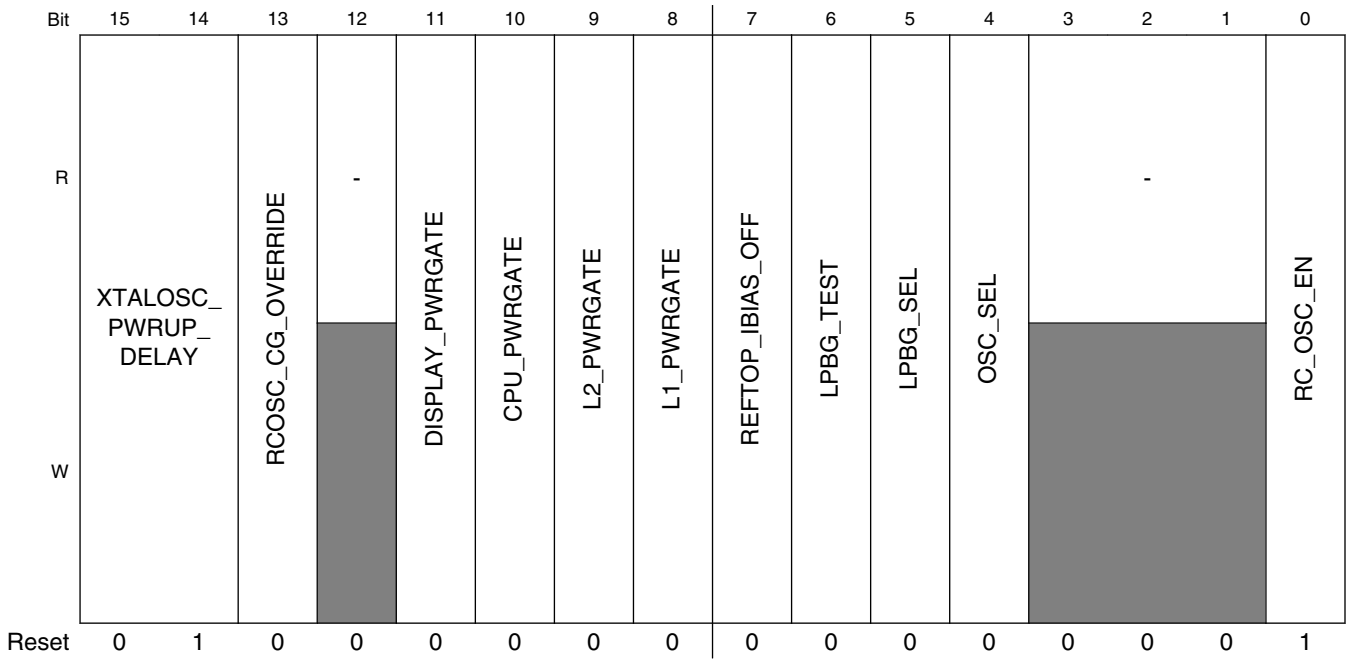
Field	Description
	001 VBG+0.78% 010 VBG+1.56% 011 VBG+2.34% 100 VBG-0.78% 101 VBG-1.56% 110 VBG-2.34% 111 VBG-3.12%
3 REFTOP_ SELFBIASOFF	Control bit to disable the self-bias circuit in the analog bandgap. The self-bias circuit is used by the bandgap during startup. This bit should be set after the 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...

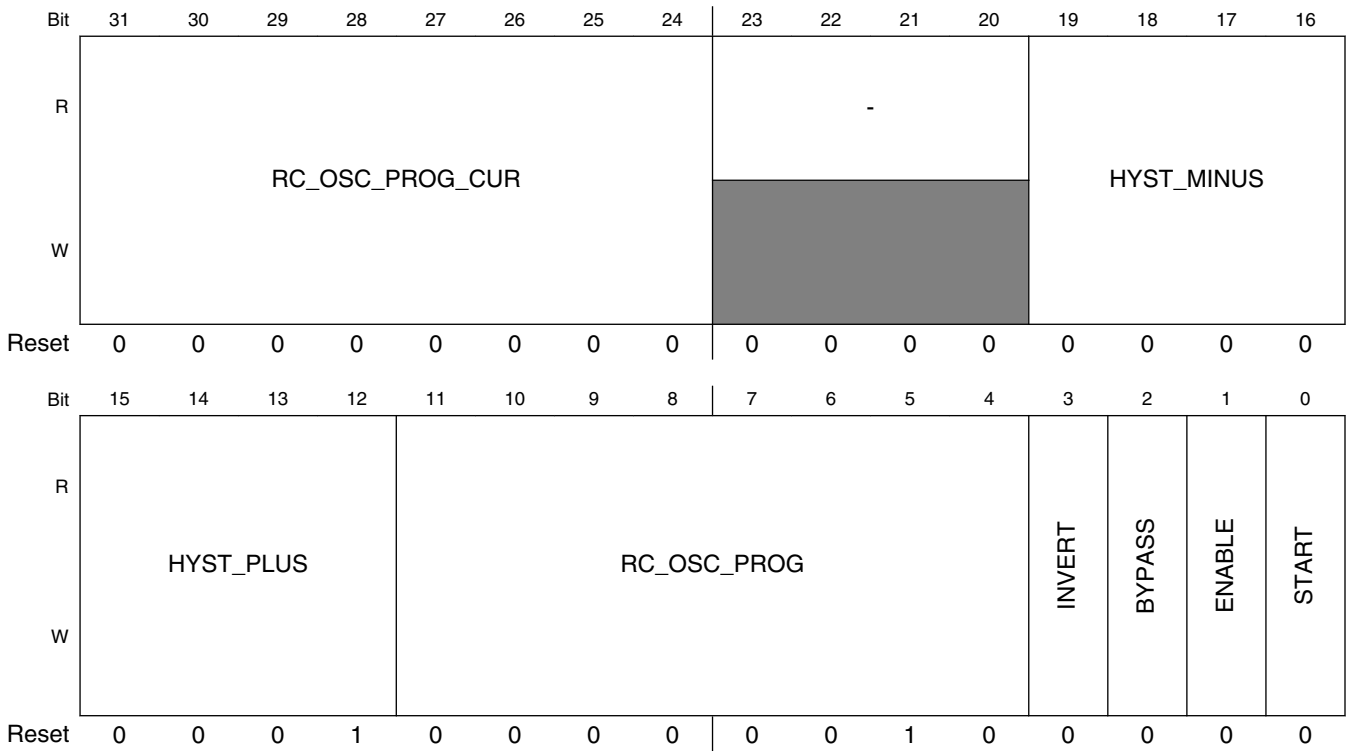
XTALOSC24M_LOWPWR_CTRL*n* field descriptions (continued)

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

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	COUNT_RC_CUR												-				COUNT_RC_TRG															
W																																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	1	0	1	1	1	0

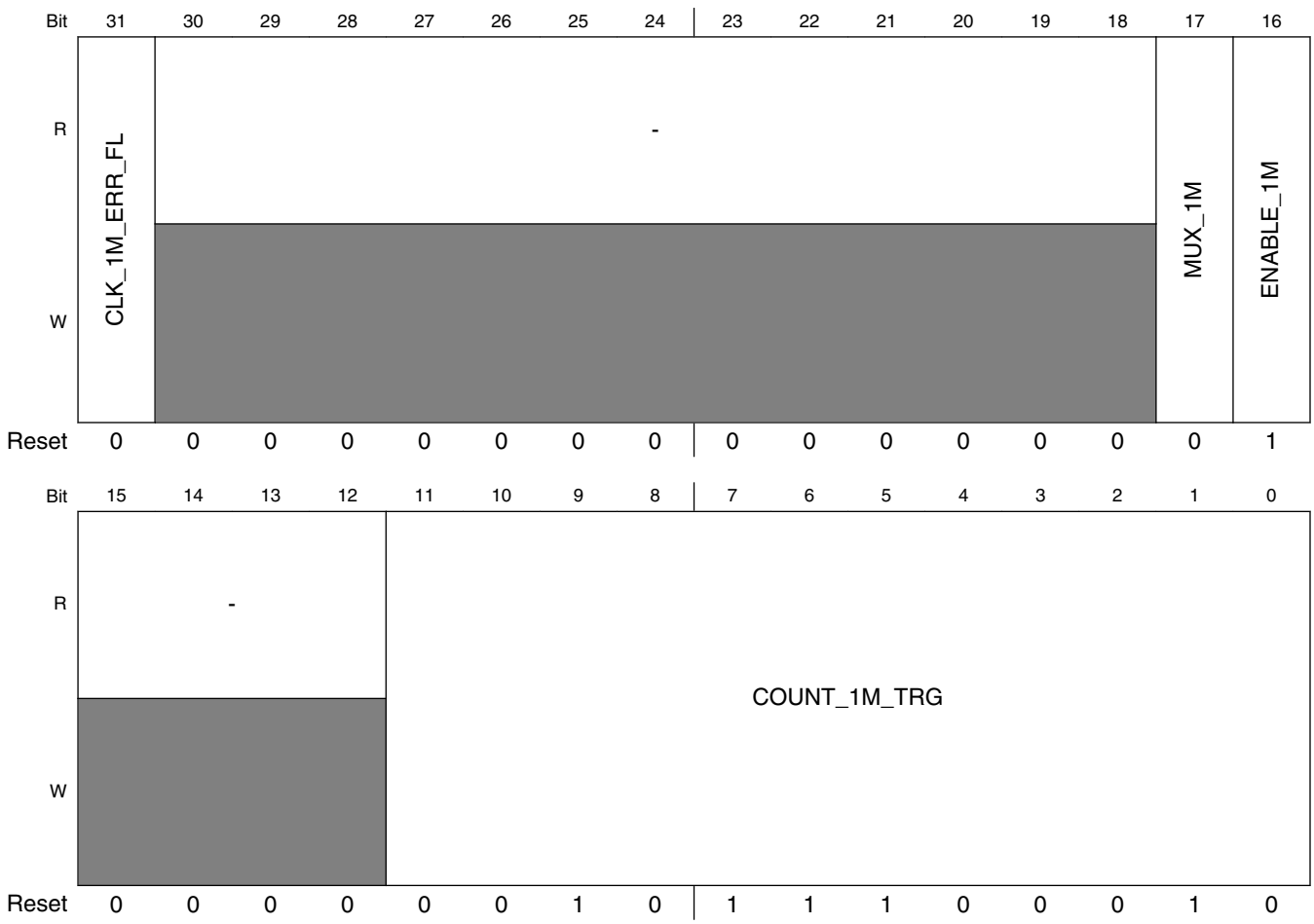
XTALOSC24M_OSC_CONFIG1*n* field descriptions

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.

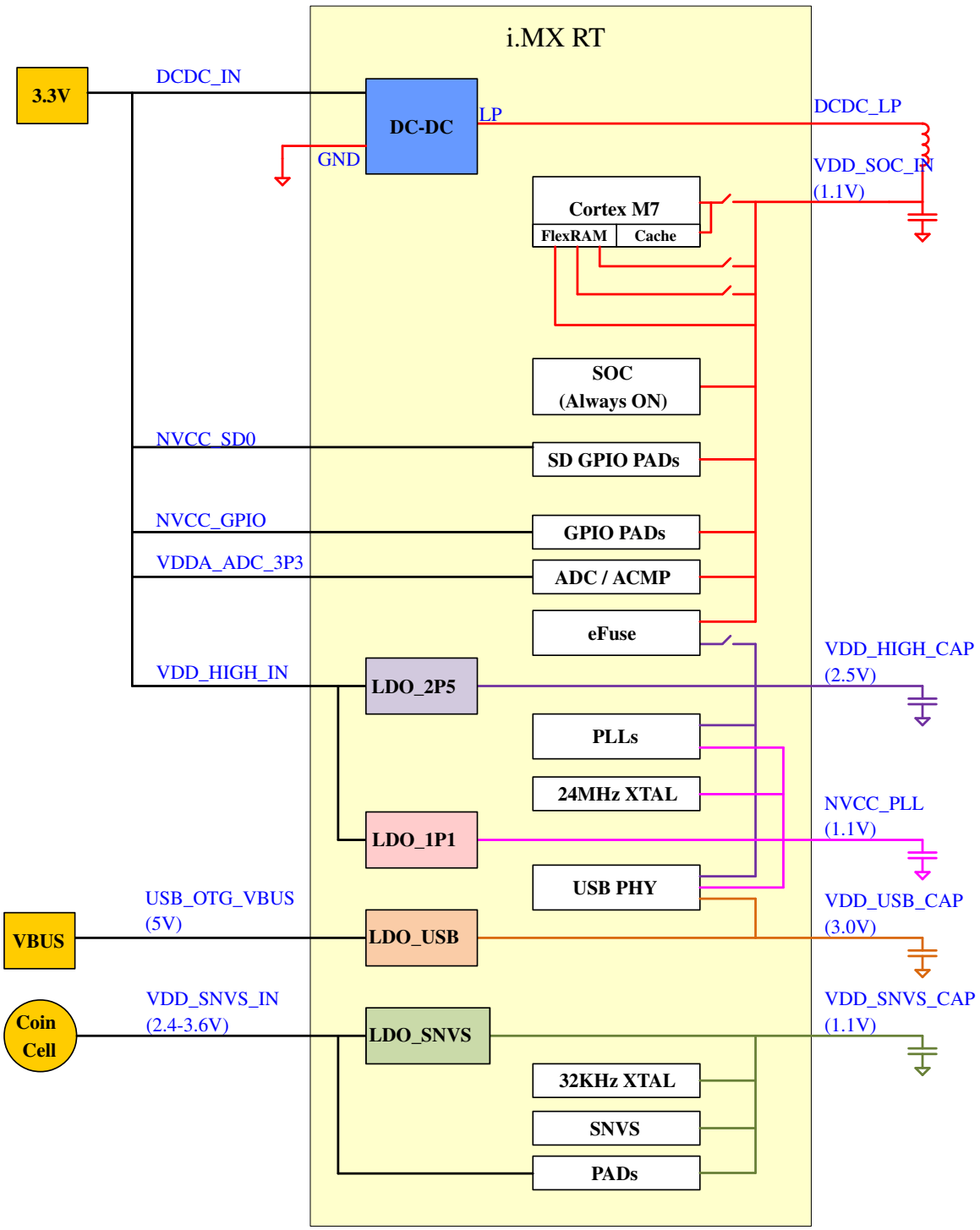


Figure 16-1. Power system overview

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	16.6.1/638

Table continues on the next page...

PMU memory map (continued)

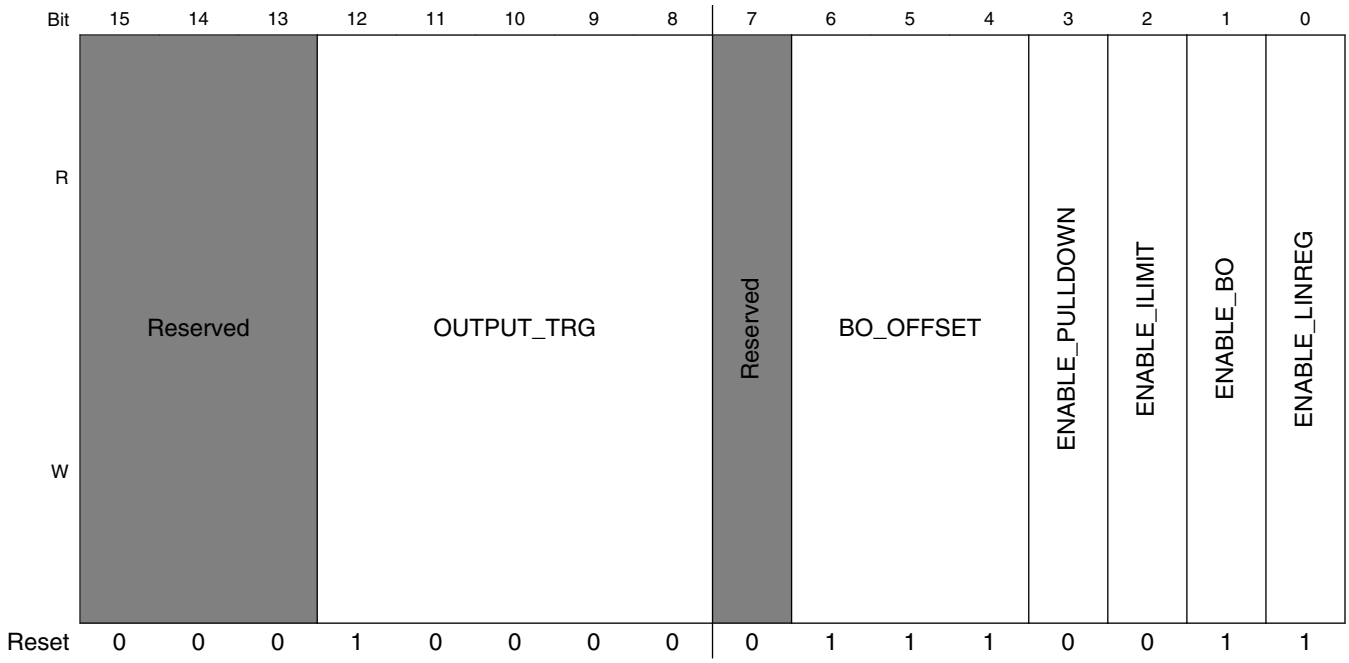
Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400D_8114	Regulator 1P1 Register (PMU_REG_1P1_SET)	32	R/W	0000_1073h	16.6.1/638
400D_8118	Regulator 1P1 Register (PMU_REG_1P1_CLR)	32	R/W	0000_1073h	16.6.1/638
400D_811C	Regulator 1P1 Register (PMU_REG_1P1_TOG)	32	R/W	0000_1073h	16.6.1/638
400D_8120	Regulator 3P0 Register (PMU_REG_3P0)	32	R/W	0000_0F74h	16.6.2/641
400D_8124	Regulator 3P0 Register (PMU_REG_3P0_SET)	32	R/W	0000_0F74h	16.6.2/641
400D_8128	Regulator 3P0 Register (PMU_REG_3P0_CLR)	32	R/W	0000_0F74h	16.6.2/641
400D_812C	Regulator 3P0 Register (PMU_REG_3P0_TOG)	32	R/W	0000_0F74h	16.6.2/641
400D_8130	Regulator 2P5 Register (PMU_REG_2P5)	32	R/W	0000_1073h	16.6.3/643
400D_8134	Regulator 2P5 Register (PMU_REG_2P5_SET)	32	R/W	0000_1073h	16.6.3/643
400D_8138	Regulator 2P5 Register (PMU_REG_2P5_CLR)	32	R/W	0000_1073h	16.6.3/643
400D_813C	Regulator 2P5 Register (PMU_REG_2P5_TOG)	32	R/W	0000_1073h	16.6.3/643
400D_8140	Digital Regulator Core Register (PMU_REG_CORE)	32	R/W	0048_2012h	16.6.4/645
400D_8144	Digital Regulator Core Register (PMU_REG_CORE_SET)	32	R/W	0048_2012h	16.6.4/645
400D_8148	Digital Regulator Core Register (PMU_REG_CORE_CLR)	32	R/W	0048_2012h	16.6.4/645
400D_814C	Digital Regulator Core Register (PMU_REG_CORE_TOG)	32	R/W	0048_2012h	16.6.4/645
400D_8150	Miscellaneous Register 0 (PMU_MISC0)	32	R/W	0400_0000h	16.6.5/649
400D_8154	Miscellaneous Register 0 (PMU_MISC0_SET)	32	R/W	0400_0000h	16.6.5/649
400D_8158	Miscellaneous Register 0 (PMU_MISC0_CLR)	32	R/W	0400_0000h	16.6.5/649
400D_815C	Miscellaneous Register 0 (PMU_MISC0_TOG)	32	R/W	0400_0000h	16.6.5/649
400D_8160	Miscellaneous Register 1 (PMU_MISC1)	32	R/W	0000_0000h	16.6.6/653
400D_8164	Miscellaneous Register 1 (PMU_MISC1_SET)	32	R/W	0000_0000h	16.6.6/653
400D_8168	Miscellaneous Register 1 (PMU_MISC1_CLR)	32	R/W	0000_0000h	16.6.6/653
400D_816C	Miscellaneous Register 1 (PMU_MISC1_TOG)	32	R/W	0000_0000h	16.6.6/653
400D_8170	Miscellaneous Control Register (PMU_MISC2)	32	R/W	0027_2727h	16.6.7/654
400D_8174	Miscellaneous Control Register (PMU_MISC2_SET)	32	R/W	0027_2727h	16.6.7/654
400D_8178	Miscellaneous Control Register (PMU_MISC2_CLR)	32	R/W	0027_2727h	16.6.7/654
400D_817C	Miscellaneous Control Register (PMU_MISC2_TOG)	32	R/W	0027_2727h	16.6.7/654

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...

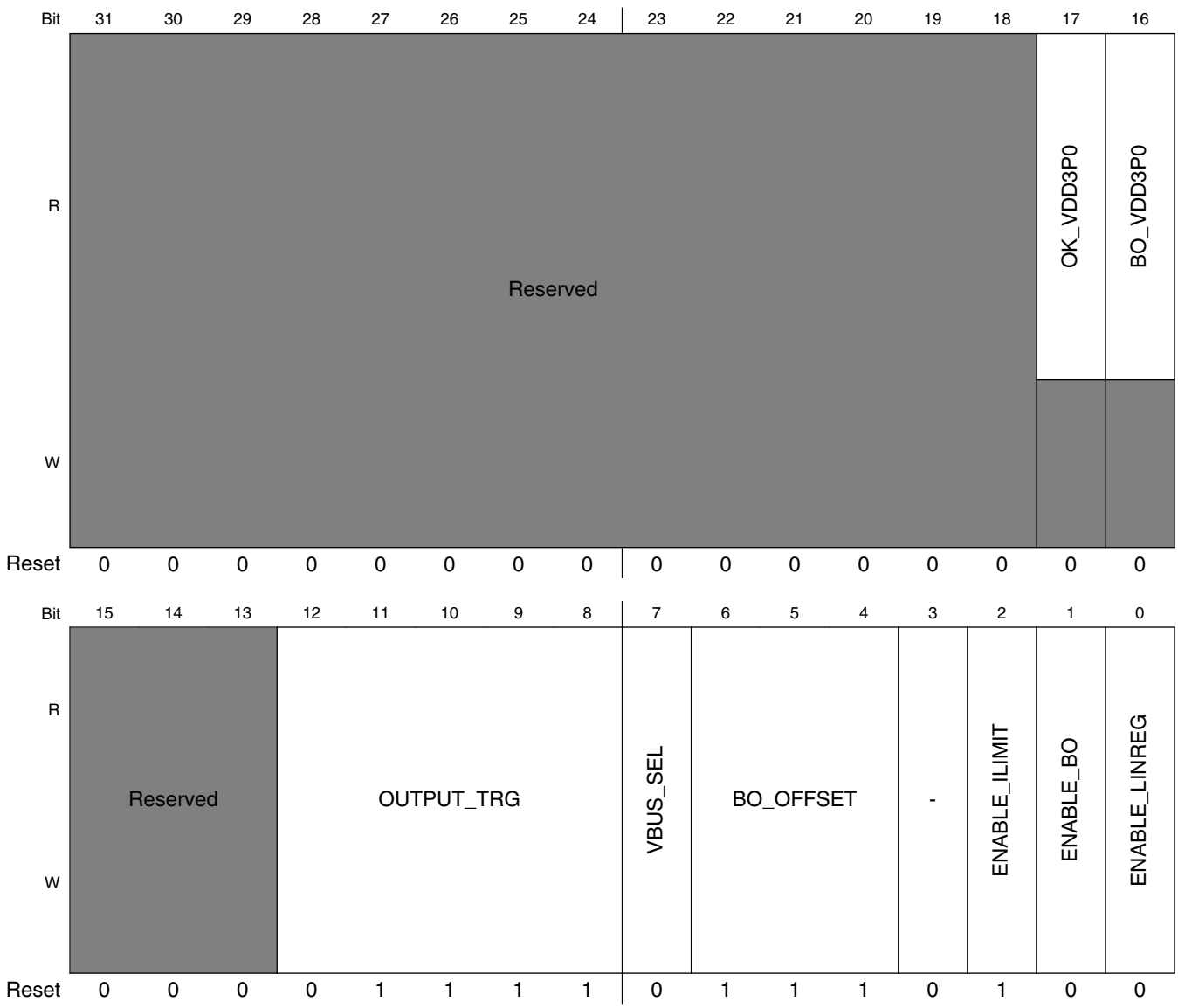
PMU_REG_1P1*n* field descriptions (continued)

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...

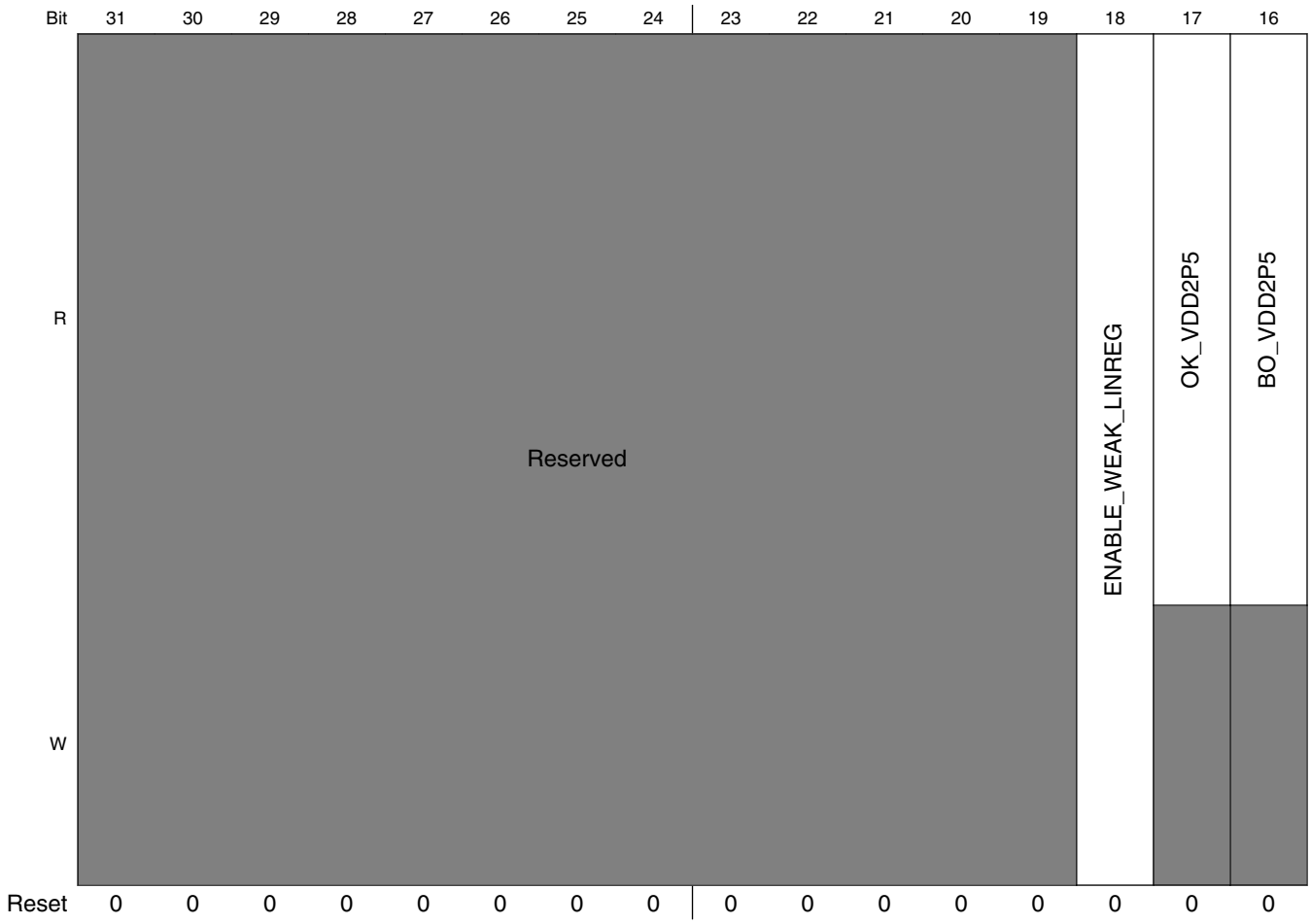
PMU_REG_3P0*n* field descriptions (continued)

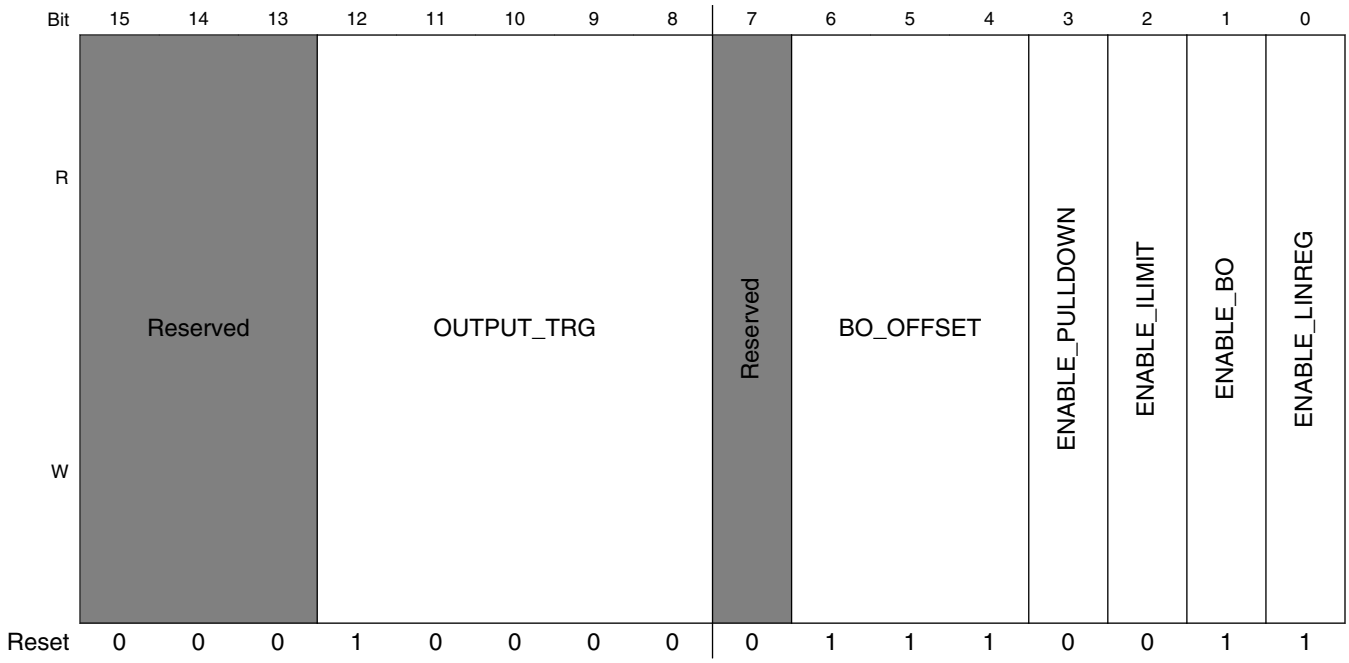
Field	Description
17 OK_VDD3P0	Status bit that signals when the regulator output is ok. 1 = regulator output > brownout target
16 BO_VDD3P0	Status bit that signals when a brownout is detected on the regulator output.
15–13 -	This field is reserved.
12–8 OUTPUT_TRG	Control bits to adjust the regulator 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...

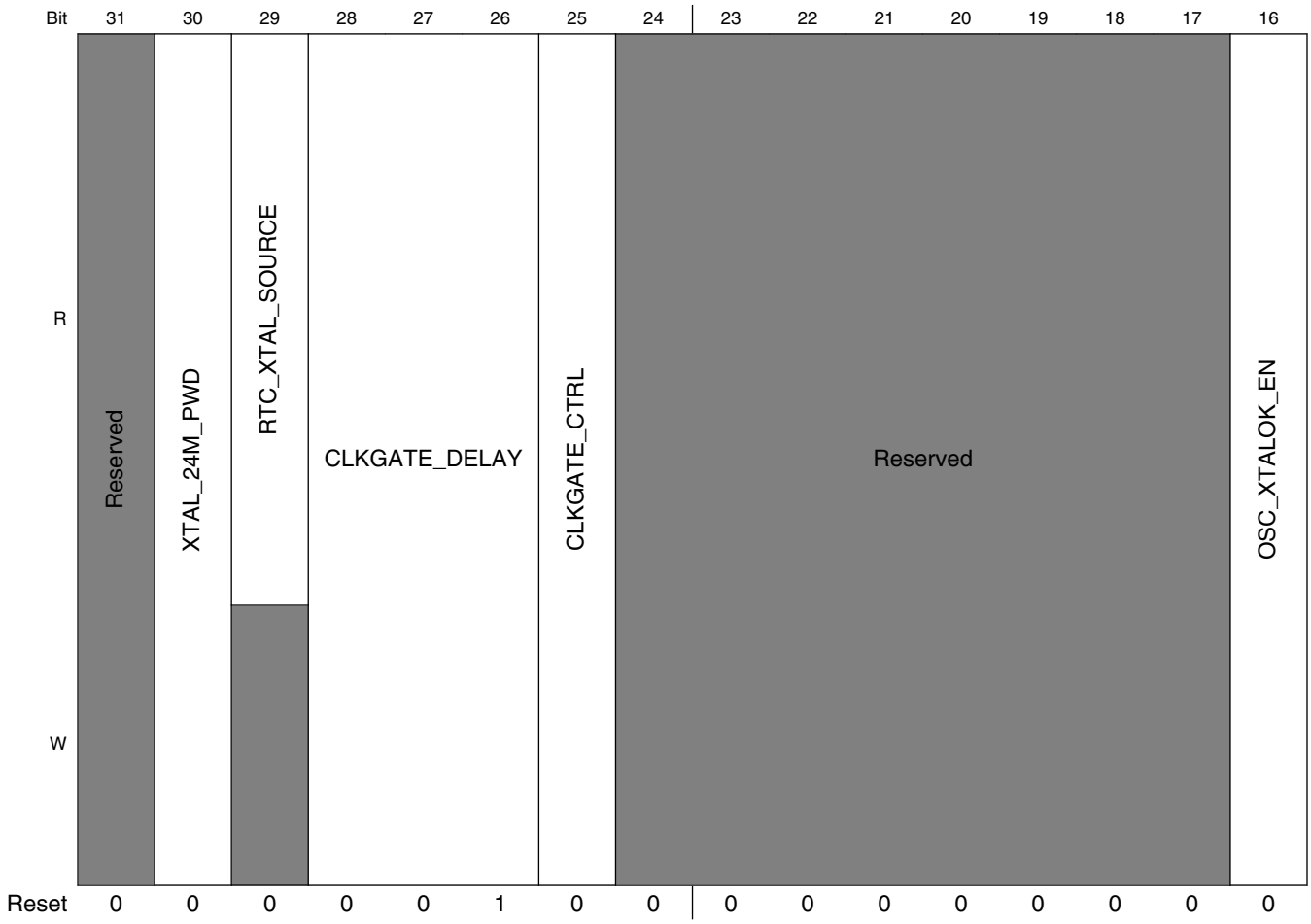
PMU_REG_CORE*n* field descriptions (continued)

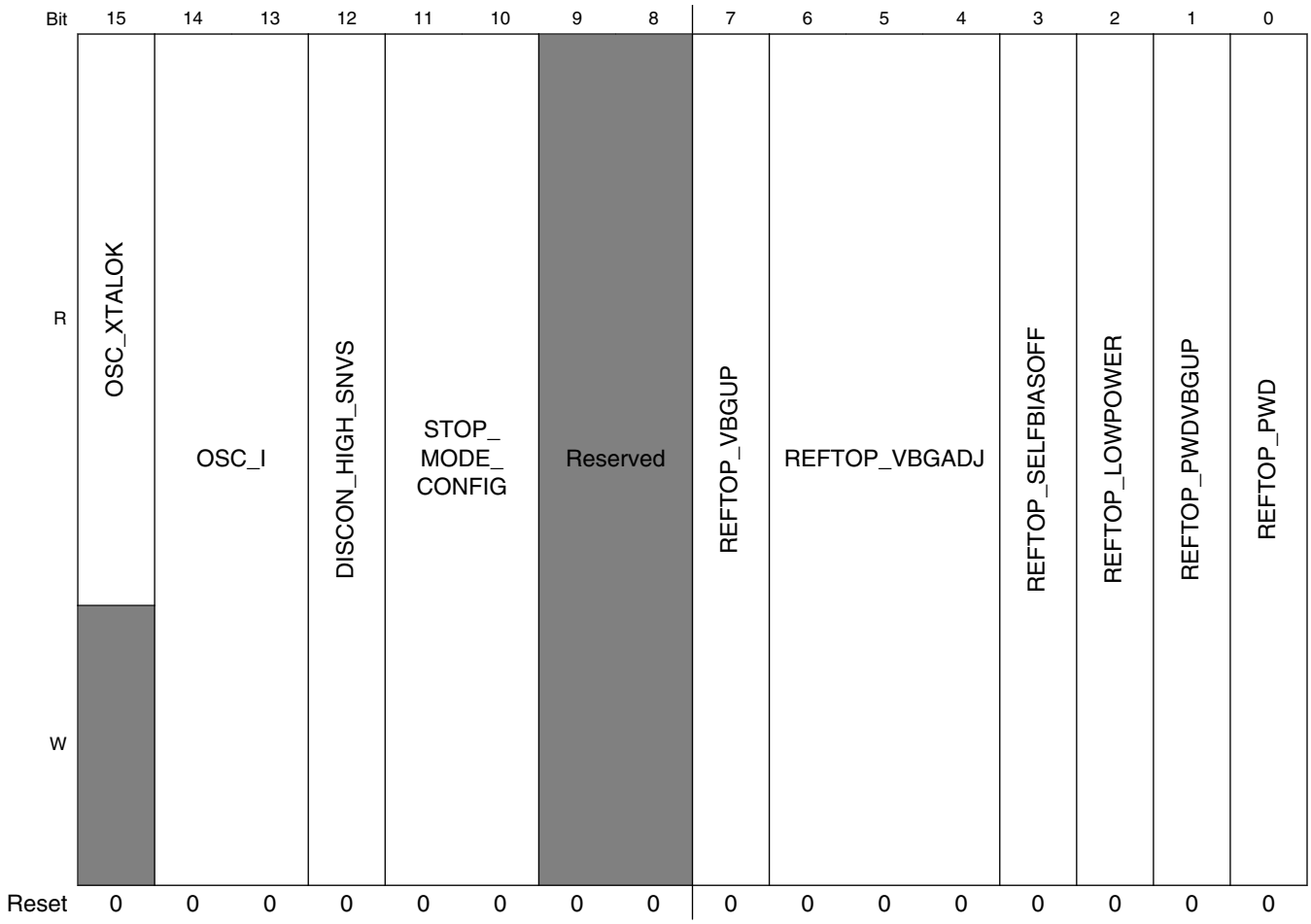
Field	Description
	00011 Target core voltage = 0.775V ... 10000 Target core voltage = 1.100V ... 11110 Target core voltage = 1.450V 11111 Power FET switched full on. No regulation.

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 -	This field is reserved.
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

Table continues on the next page...

PMU_MISC0*n* field descriptions (continued)

Field	Description
	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 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

Table continues on the next page...

PMU_MISC0*n* field descriptions (continued)

Field	Description
7 REFTOP_ VBGUP	Status bit that signals the analog bandgap voltage is up and stable. 1 - Stable.
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

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													
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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
-	This field is reserved. Reserved

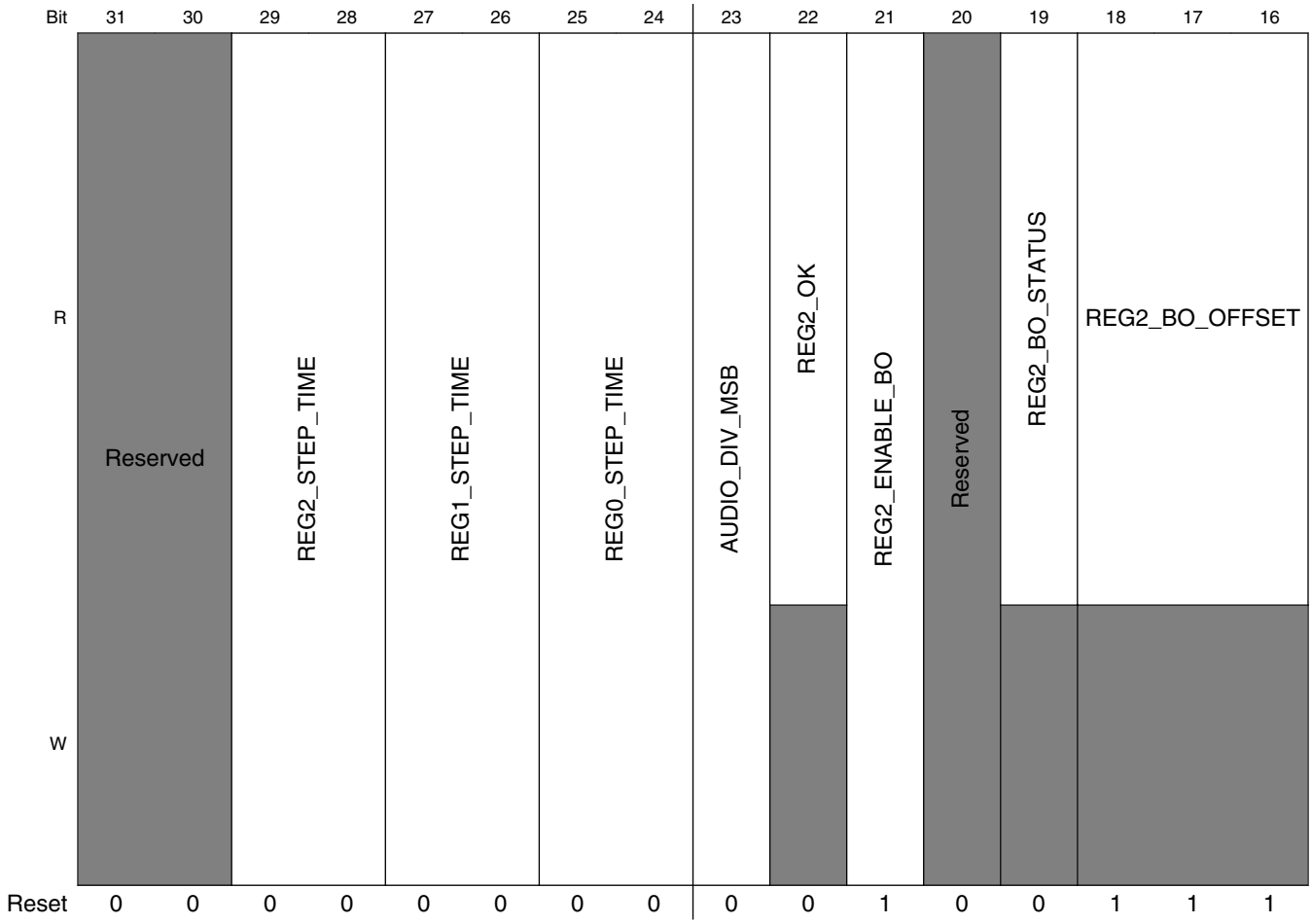
16.6.7 Miscellaneous Control Register (PMU_MISC2*n*)

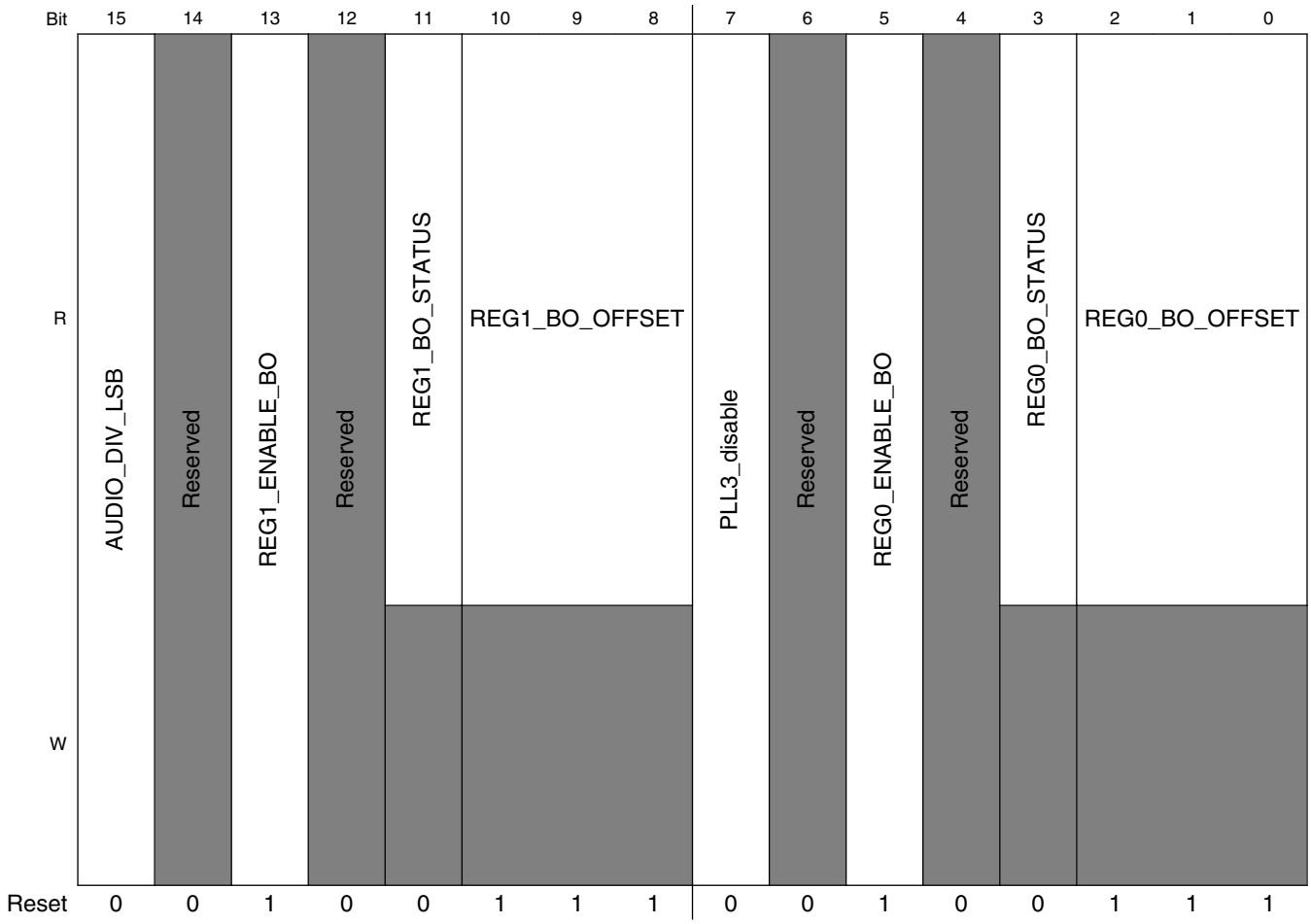
This register defines the control for miscellaneous PMU Analog blocks.

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 -	This field is reserved.
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 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

Table continues on the next page...

PMU_MISC2*n* field descriptions (continued)

Field	Description
	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
14 -	This field is reserved. Reserved
13 REG1_ENABLE_BO	Enables the brownout detection.

Table continues on the next page...

PMU_MISC2*n* field descriptions (continued)

Field	Description
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).

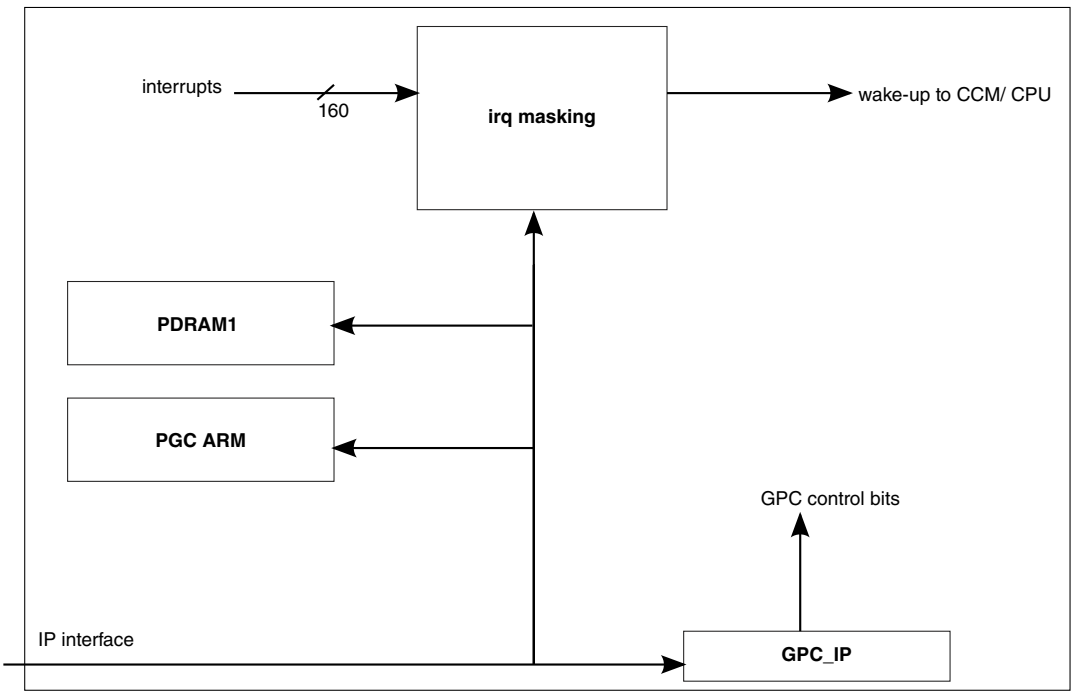


Figure 17-1. GPC Block Diagram

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 only 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 (PGCR 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.

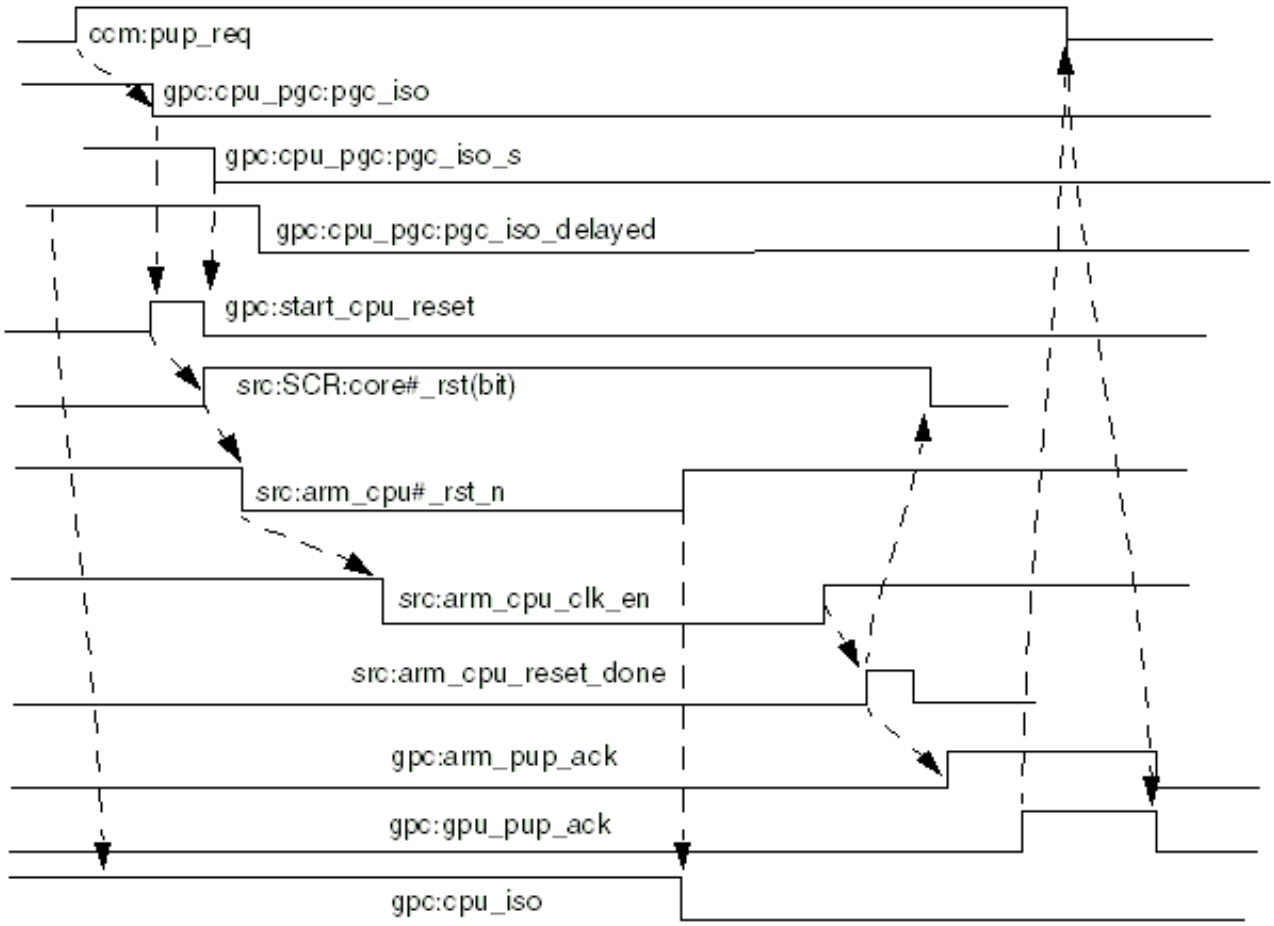


Figure 17-2. GPC-SRC handshake for reset after power gating

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.

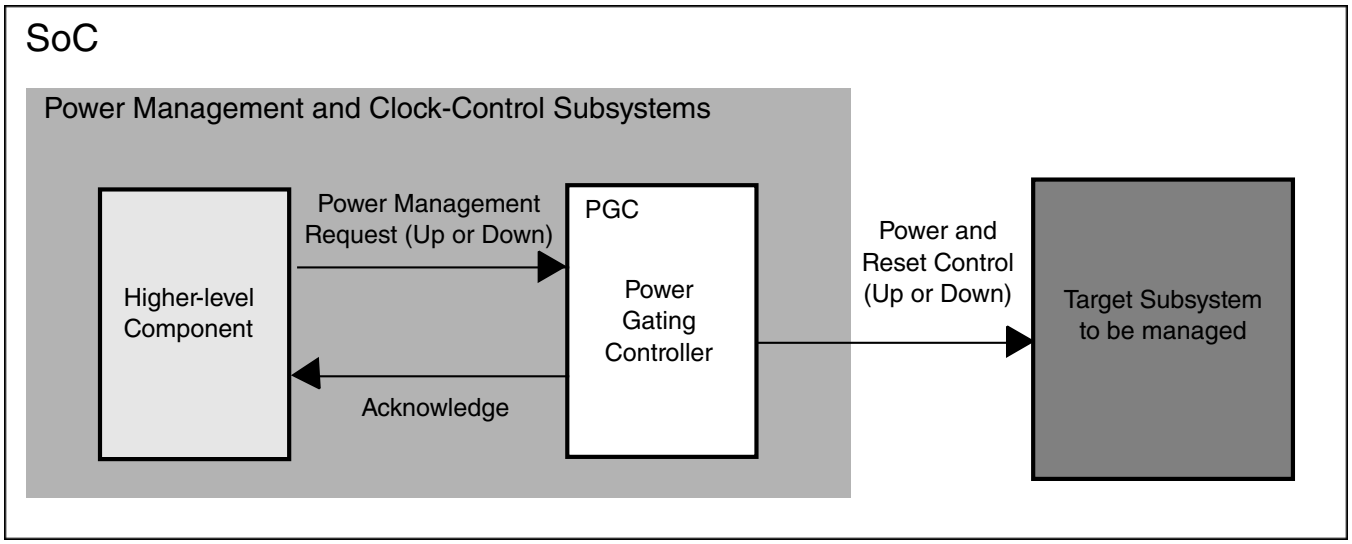


Figure 17-3. PGC Block Diagram

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

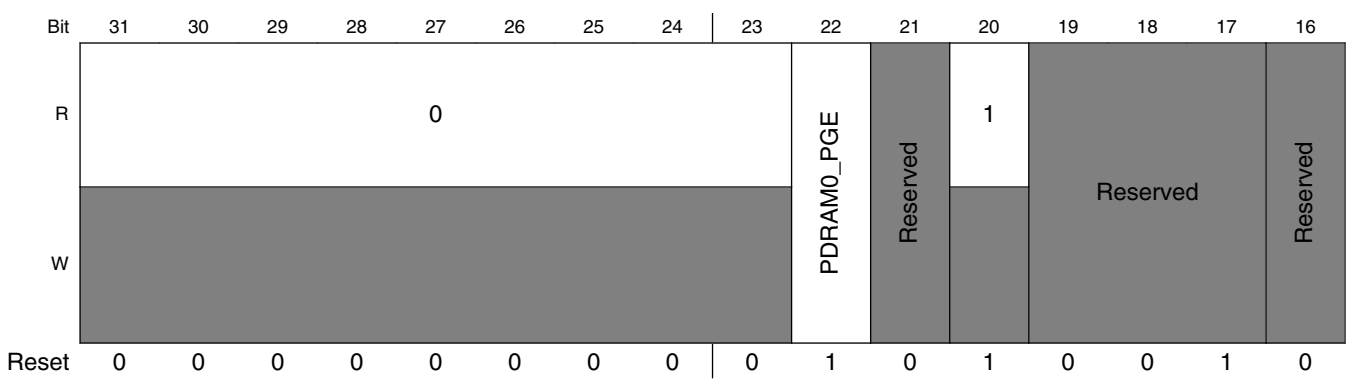
Detailed descriptions of each register can be found below.

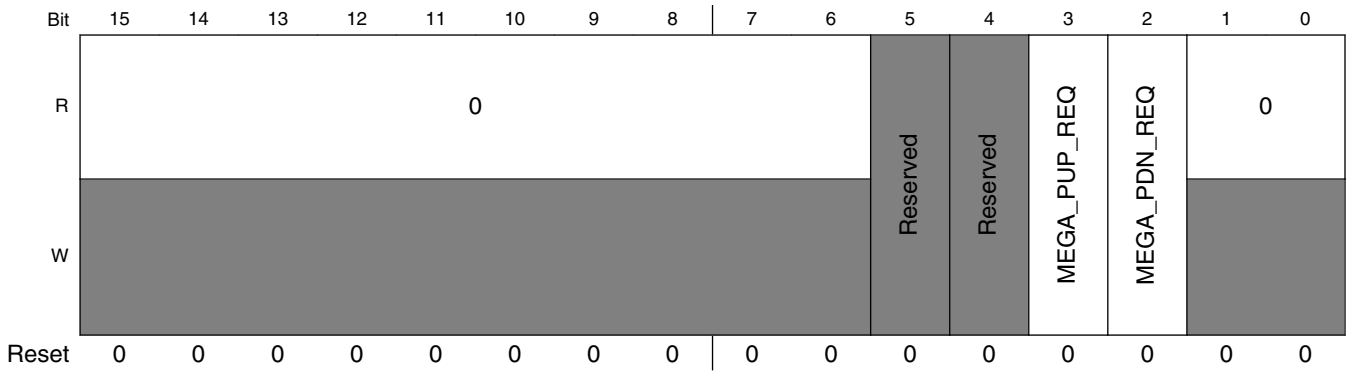
GPC memory map

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	17.6.1/664
400F_4008	IRQ masking register 1 (GPC_IMR1)	32	R/W	0000_0000h	17.6.2/666
400F_400C	IRQ masking register 2 (GPC_IMR2)	32	R/W	0000_0000h	17.6.3/666
400F_4010	IRQ masking register 3 (GPC_IMR3)	32	R/W	0000_0000h	17.6.4/666
400F_4014	IRQ masking register 4 (GPC_IMR4)	32	R/W	0000_0000h	17.6.5/667
400F_4018	IRQ status resister 1 (GPC_ISR1)	32	R	0000_0000h	17.6.6/667
400F_401C	IRQ status resister 2 (GPC_ISR2)	32	R	0000_0000h	17.6.7/668
400F_4020	IRQ status resister 3 (GPC_ISR3)	32	R	0000_0000h	17.6.8/668
400F_4024	IRQ status resister 4 (GPC_ISR4)	32	R	0000_0000h	17.6.9/669
400F_4034	IRQ masking register 5 (GPC_IMR5)	32	R/W	0000_0000h	17.6.10/ 669
400F_4038	IRQ status resister 5 (GPC_ISR5)	32	R	0000_0000h	17.6.11/ 670

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 -	This field is reserved. Reserved
16 -	This field is reserved. Reserved
15–6 Reserved	This read-only field is reserved and always has the value 0.
5 -	This field is reserved. Reserved
4 -	This field is reserved. Reserved
3 MEGA_PUP_ REQ	MEGA domain power up request. Self-clear bit. Writable but read will always return 0. **NOTE:** Software may directly control PDRAM1 power gate and utilize hardware control for reset sequence. 0 — No Request 1 — Request power up sequence
2 MEGA_PDN_ REQ	MEGA domain power down request. Self-clear bit. Writable but read will always return 0. **NOTE:** Software may directly control PDRAM1 power gate and utilize hardware control for reset sequence. 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

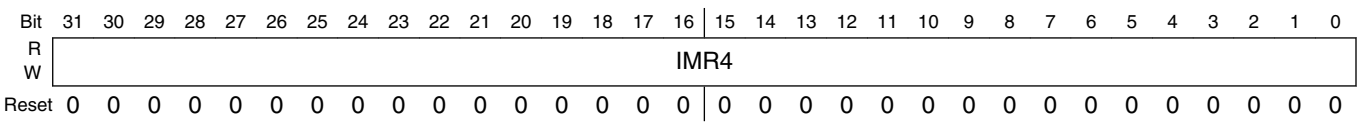
GPC_IMR3 field descriptions

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



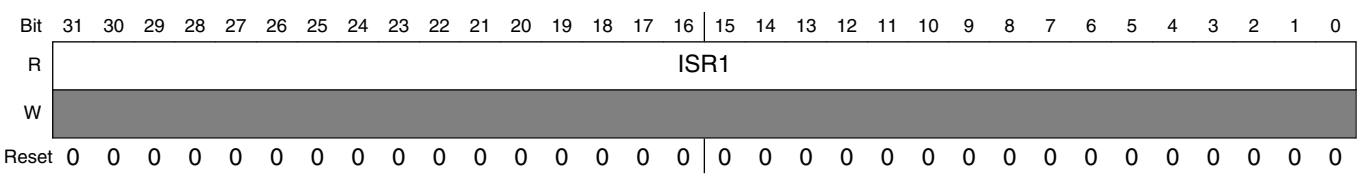
GPC_IMR4 field descriptions

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



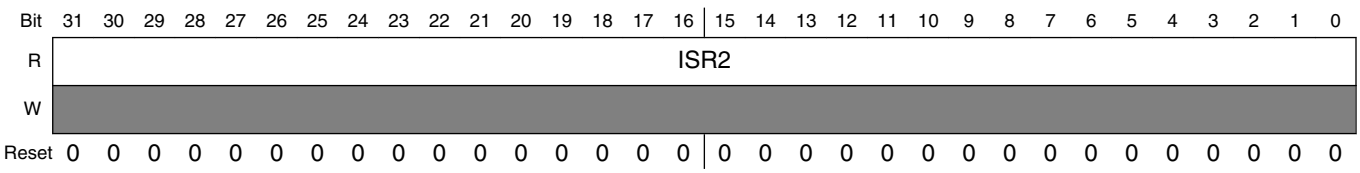
GPC_ISR1 field descriptions

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



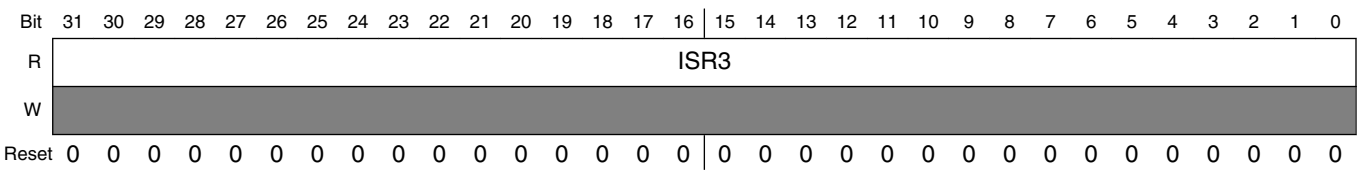
GPC_ISR2 field descriptions

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



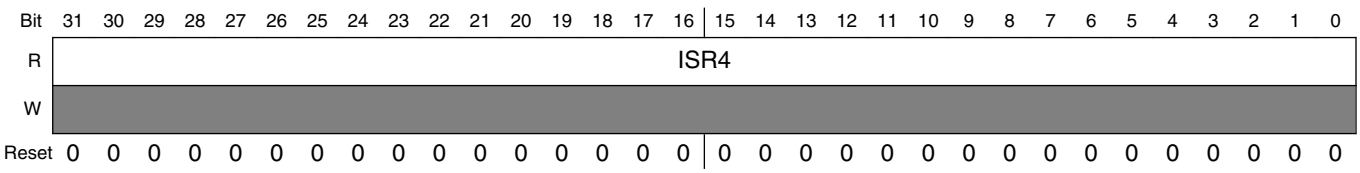
GPC_ISR3 field descriptions

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



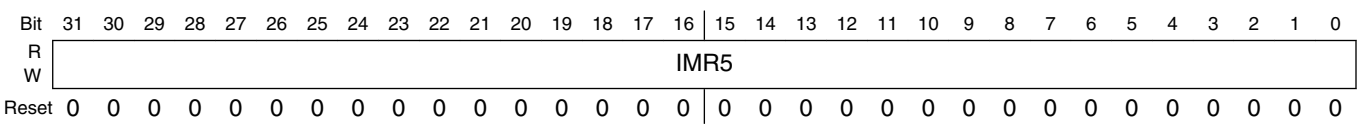
GPC_ISR4 field descriptions

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



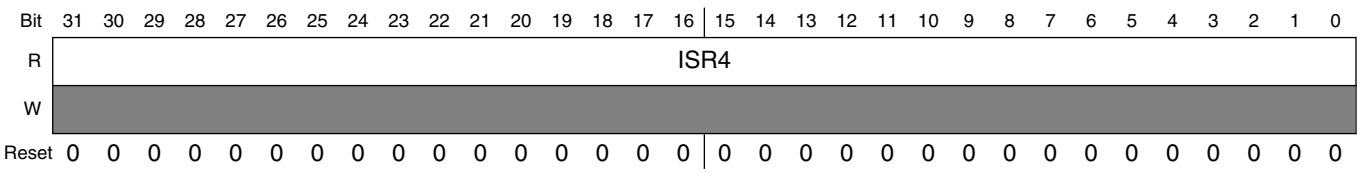
GPC_IMR5 field descriptions

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
ISR4	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. (The hardware asserts the signal ips_xfr_err if the PGC has been integrated with resp_sel tied low.) 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

The base address of each PGC module instantiation is specified in the GPC module. Absolute address values will be calculated by [GPC base address] + [PGC CPU/MEGA Offset].

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	17.7.1/671
400F_4224	PGC Mega Power Up Sequence Control Register (PGC_MEGA_PUPSCR)	32	R/W	0000_0F01h	17.7.2/672
400F_4228	PGC Mega Pull Down Sequence Control Register (PGC_MEGA_PDNSCR)	32	R/W	0000_0101h	17.7.3/672
400F_422C	PGC Mega Power Gating Controller Status Register (PGC_MEGA_SR)	32	R/W	0000_0000h	17.7.4/673
400F_42A0	PGC CPU Control Register (PGC_CPU_CTRL)	32	R/W	0000_0000h	17.7.5/674
400F_42A4	PGC CPU Power Up Sequence Control Register (PGC_CPU_PUPSCR)	32	R/W	0000_0F01h	17.7.6/674
400F_42A8	PGC CPU Pull Down Sequence Control Register (PGC_CPU_PDNSCR)	32	R/W	0000_0101h	17.7.7/675
400F_42AC	PGC CPU Power Gating Controller Status Register (PGC_CPU_SR)	32	R/W	0000_0000h	17.7.8/676

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), 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). **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). **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. 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

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_CPU_PUPSCR field descriptions

Field	Description
31–14 Reserved	This read-only field is reserved and always has the value 0.

13–8
SW2ISO
	There are two different silicon revisions: 1.0 and 1.1 (defined by USB_ANALOG_DIGPROG).
	• For silicon revision 1.0: PGC waits for 2048*SW2ISO cycles of IPG_CLK before negating isolation.
	• For silicon revision 1.1:
	  • SW[5] = 1: PGC waits for 64*SW2ISO cycles of IPG_CLK before negating isolation.
	  • SW[5] = 0: PGC waits for 2048*SW2ISO cycles of IPG_CLK before negating isolation.
	
	NOTE: SW2ISO must not be programmed to zero. The SW2ISO value should be chosen such that the delay before negating isolation is greater than the LDO ramp-up time.

7–6 Reserved	This read-only field is reserved and always has the value 0.

SW	There are two different silicon revisions: 1.0 and 1.1 (defined by USB_ANALOG_DIGPROG). When IPG_CLK frequency is slow down in low power mode, it can set SW[5] to speed up Arm wakeup time.
	• For silicon revision 1.0: PGC waits for 2048*SW[5:0] cycles of IPG_CLK to switch on Arm power after pup_reg is asserted.
	• For silicon revision 1.1:
	  • SW[5] = 1: PGC waits for 64*SW[4:0] cycles of IPG_CLK to switch on Arm power after pup_req is asserted.
	  • SW[5] = 0: PGC waits for 2048*SW[4:0] cycles of IPG_CLK to switch on Arm power after pup_req is asserted.
	
	NOTE: SW[4:0] can be programmed to zero.

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

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_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 32k clocks equal to the value of ISO2SW before negating .

Table continues on the next page...

PGC_CPU_PDNSCR field descriptions (continued)

Field	Description
	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 32k 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

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_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. 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.

18.2 Introduction

This module is a synchronous buck mode DCDC converter with a single output.

18.3 Features

The DCDC module includes the following features:

- Buck mode, a single output
- PSWITCH pin to control DCDC to work 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

18.4 Block diagram

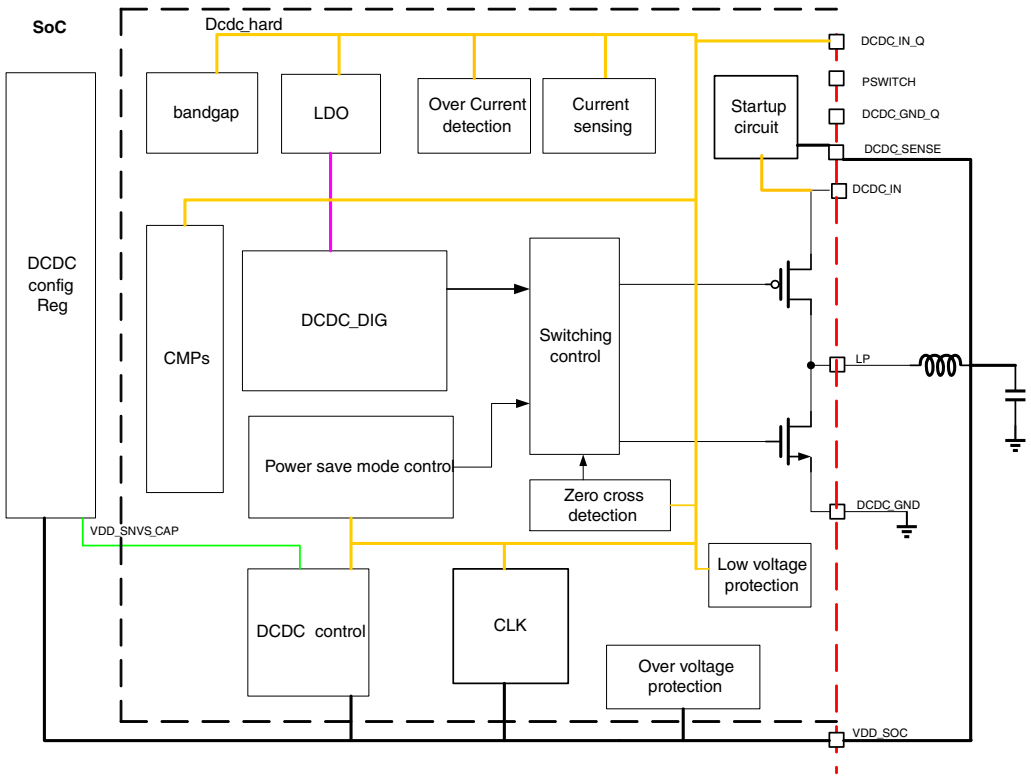


Figure 18-1. DCDC block diagram

18.5 Functional description

The DCDC converter is a synchronous step down converters with a single output. In run mode, it can operate on the continuous conduction mode or discontinuous conduction mode, depending on the level of the load current. And the discontinuous conduction mode need be enabled by the register configuration.

DCDC can also be configured to operate on 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.

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.

18.6 Application information

18.6.1 Turn on DCDC

To turn on the DCDC, PSWITCH and PMIC_ON_REQ must be both high in the meantime.

NOTE

For safety purpose, over-voltage detector needs to be enabled, i.e. PWD_HIGH_VOLT_DET=1'b0.

18.6.2 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.

18.6.3 Continuous conduction mode

pwd_zcd=0x1

pwd_cmp_offset=0x0

loopctrl_en_rcscale=0x3

18.6.4 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.

18.6.5 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

18.7 Memory Map and register definition

This section includes the DCDC module memory map and detailed descriptions of all registers.

18.7.1 DCDC register descriptions

18.7.1.1 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_0009h
Ch	DCDC Register 3 (REG3)	32	RW	0000_010Eh

18.7.1.2 DCDC Register 0 (REG0)

18.7.1.2.1 Offset

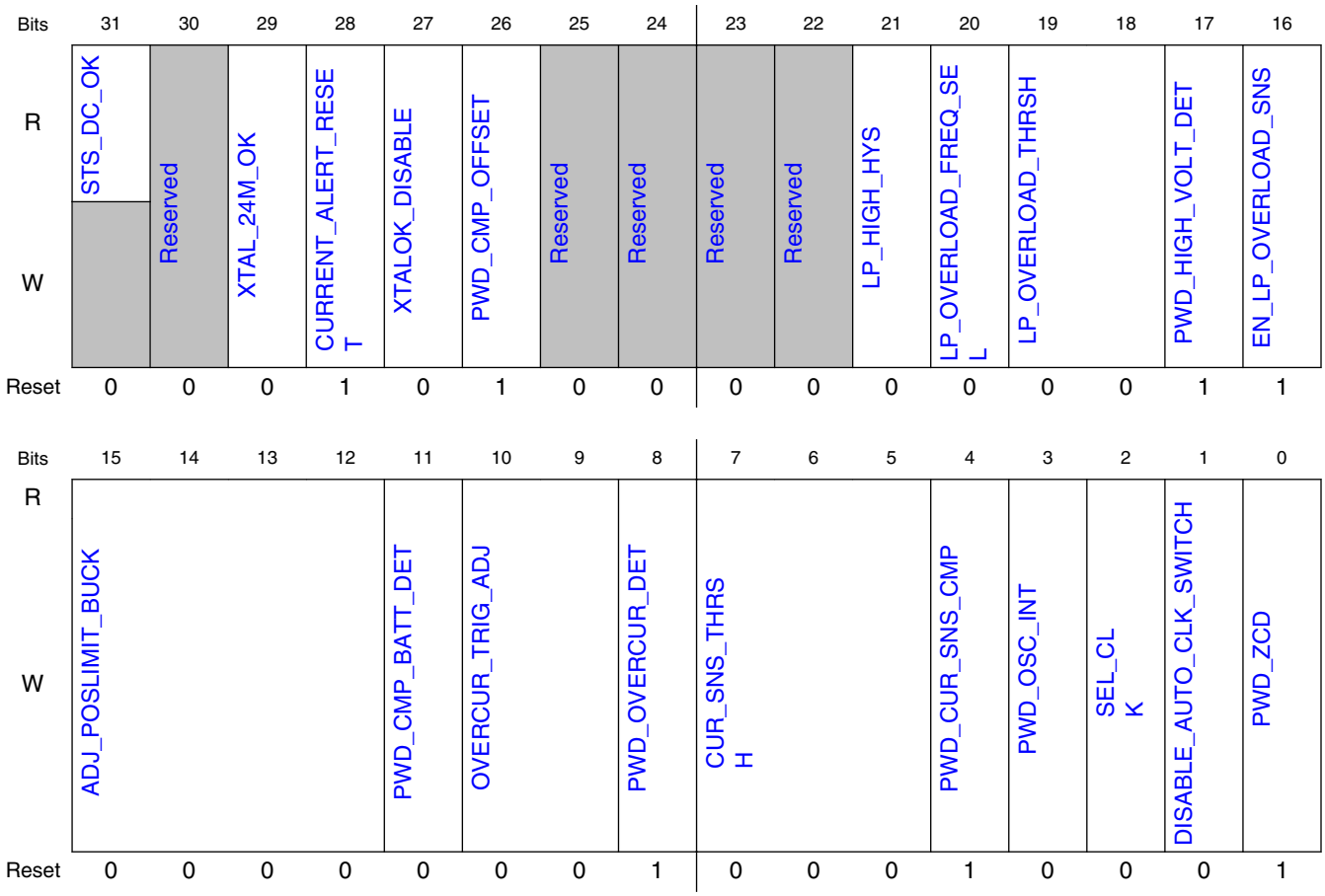
Register	Offset
REG0	0h

18.7.1.2.2 Function

DCDC register 0

This register controls various high-level functions of the DCDC

18.7.1.2.3 Diagram



18.7.1.2.4 Fields

Field	Function
31 STS_DC_OK	Status register to indicate DCDC status. 1'b1: DCDC already settled 1'b0: DCDC is settling
30 —	Reserved.
29 XTAL_24M_OK	set to 1 to switch internal ring osc to xtal 24M
28 CURRENT_ALERT_RESET	reset current alert signal
27 XTALOK_DISABLE	1'b1: Disable xtalok detection circuit 1'b0: Enable xtalok detection circuit
26	power down output range comparator

Table continues on the next page...

Field	Function
PWD_CMP_OFFSET	
25 — 24 — 23 — 22 —	Reserved.
21 LP_HIGH_HYS	Adjust hysteretic value in low power from 12.5mV to 25mV
20 LP_OVERLOAD_FREQ_SEL	the period of counting the charging times in power save mode • 0: eight 32k cycle • 1: sixteen 32k cycle
19-18 LP_OVERLOAD_THRSH	the threshold of the counting number of charging times during the period that lp_overload_freq_sel sets in power save mode. • 0x0: 32 • 0x1: 64 • 0x2: 16 • 0x3: 8
17 PWD_HIGH_VOLT_DET	power down overvoltage detection comparator
16 EN_LP_OVERLOAD_SNS	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
15-12 ADJ_POSLIMIT_BUCK	adjust value to poslimit_buck register
11 PWD_CMP_BATT_DET	set to "1" to power down the low voltage detection comparator
10-9 OVERCUR_TRIG_ADJ	The threshold of over current detection in run mode and power save mode: 0x0: run mode 1 A, power save mode 0.25 A 0x1: run mode 2 A, power save mode 0.25 A 0x2: run mode 1 A, power save mode 0.2 A 0x3: run mode 2 A, power save mode 0.2 A
8 PWD_OVERCUR_DET	power down overcurrent detection comparator

Table continues on the next page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td>7-5
CUR_SNS_THRSH</td><td>Set the threshold of current detector, if the peak current of the inductor exceeds the threshold, the current detector will assert.
<table><tr><td>Cur_sns_thrsh</td><td>threshold</td></tr><tr><td>0x0</td><td>150 mA</td></tr><tr><td>0x1</td><td>250 mA</td></tr><tr><td>0x2</td><td>350 mA</td></tr><tr><td>0x3</td><td>450 mA</td></tr><tr><td>0x4</td><td>550 mA</td></tr><tr><td>0x5</td><td>650 mA</td></tr></table></td></tr>
<tr><td>4
PWD_CUR_SNS_CMP</td><td>The power down signal of the current detector.</td></tr>
<tr><td>3
PWD_OSC_INT</td><td>Power down internal osc. Only set this bit, when 24 MHz crystal osc is available</td></tr>
<tr><td>2
SEL_CLK</td><td>select 24 MHz Crystal clock for DCDC, when dcdc_disable_auto_clk_switch is set.</td></tr>
<tr><td>1
DISABLE_AUTO_CLK_SWITCH</td><td>Disable automatic clock switch from internal osc to xtal clock.</td></tr>
<tr><td>0
PWD_ZCD</td><td>power down the zero cross detection function for discontinuous conductor mode</td></tr>
</table>

18.7.1.3 DCDC Register 1 (REG1)

18.7.1.3.1 Offset

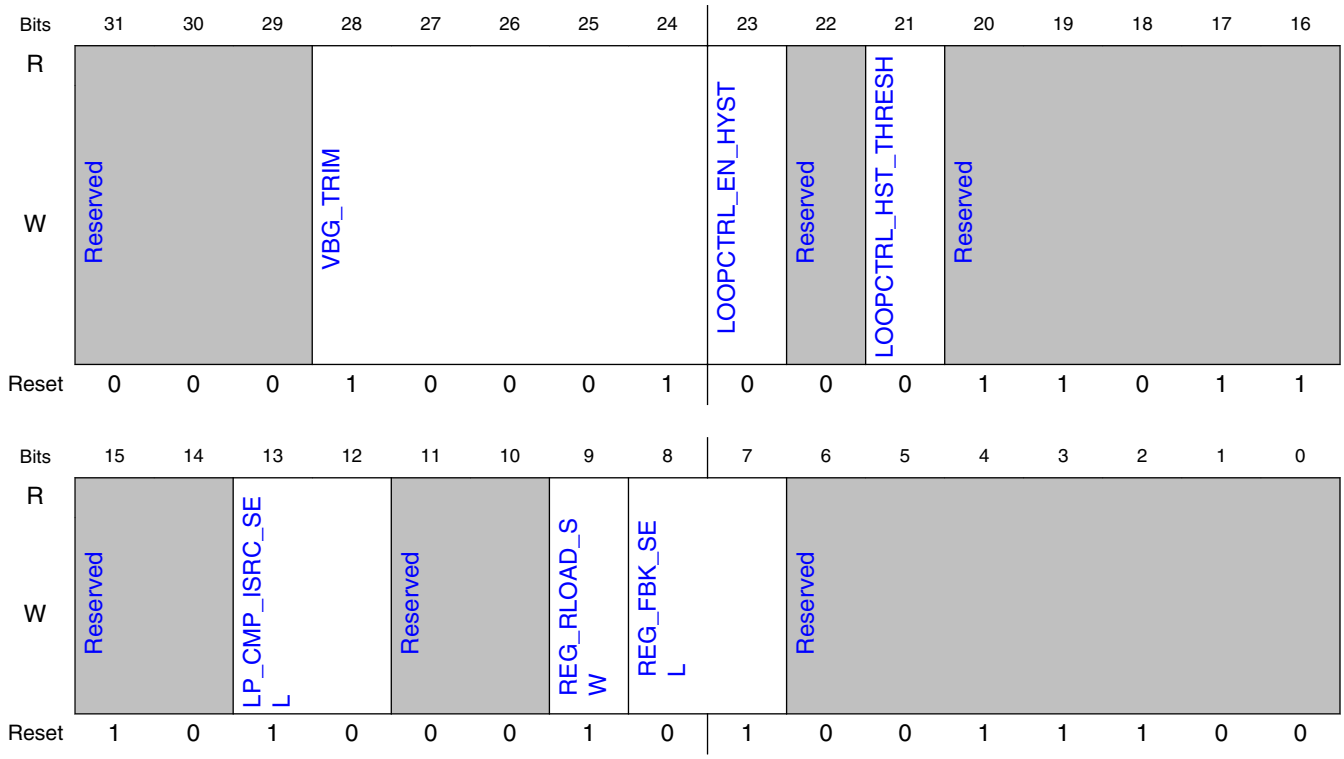
Register	Offset
REG1	4h

18.7.1.3.2 Function

DCDC register 1

This register controls various high-level functions of the DCDC

18.7.1.3.3 Diagram



18.7.1.3.4 Fields

Field	Function
31-29 —	Reserved.
28-24 VBG_TRIM	trim bandgap voltage
23 LOOPCTRL_EN_HYST	Enable hysteresis in switching converter common mode analog comparators. This feature will improve transient supply ripple and efficiency
22 —	Reserved.
21 LOOPCTRL_HST_THRESH	increase the threshold detection for common mode analog comparator
20-14 —	Reserved.
13-12 LP_CMP_ISRC_SEL	set the current bias of low power comparator • 0x0: 50 nA • 0x1: 100 nA

Table continues on the next page...

Field	Function
	• 0x2: 200 nA • 0x3: 400 nA
11-10 —	Reserved.
9 REG_RLOAD_SW	control the load resistor of the internal regulator of DCDC, the load resistor is connected as default "1", and need set to "0" to disconnect the load resistor
8-7 REG_FBK_SEL	select the feedback point of the internal regulator
6-0 —	Reserved.

18.7.1.4 DCDC Register 2 (REG2)

18.7.1.4.1 Offset

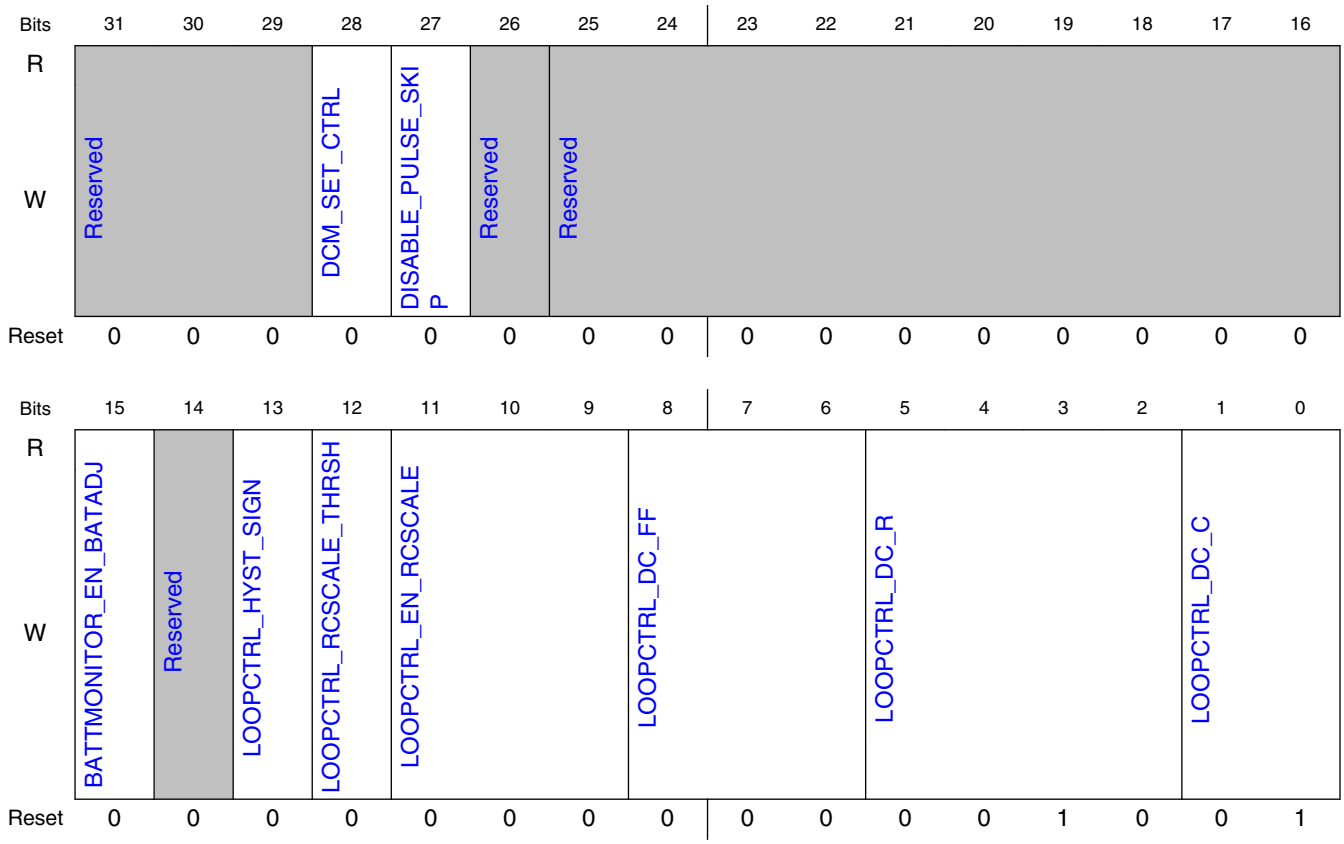
Register	Offset
REG2	8h

18.7.1.4.2 Function

DCDC register 2

This register controls various high-level functions of the DCDC

18.7.1.4.3 Diagram



18.7.1.4.4 Fields

Field	Function
31-29 —	Reserved.
28 DCM_SET_CTRL	Set high to improve the transition from heavy load to light load
27 DISABLE_PULSE_SKIP	Set to "0" : stop charging if the duty cycle is lower than what set by dcdc_neglimit_in
26 —	Reserved.
25-16 —	Reserved.
15 BATTMONITOR_EN_BATADJ	This bit enables the DC-DC to improve efficiency and minimize ripple using the information from the BATT_VAL field. It is very important that BATT_VAL contain accurate information before setting EN_BATADJ.

Table continues on the next page...

Field	Function
	NOTE: Set to logic 1 to make DCDC work in power saving mode.
14 —	Reserved.
13 LOOPCTRL_HYST_SIGN	Invert the sign of the hysteresis in DC-DC analog comparators.
12 LOOPCTRL_RCSCALE_THRSH	Increase the threshold detection for RC scale circuit.
11-9 LOOPCTRL_EN_RCSCALE	Enable analog circuit of DC-DC converter to respond faster under transient load conditions.
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.
5-2 LOOPCTRL_DC_R	Magnitude of proportional control parameter in the switching DC-DC converter control loop.
1-0 LOOPCTRL_DC_C	Ratio of integral control parameter to proportional control parameter in the switching DC-DC converter, and can be used to optimize efficiency and loop response.

18.7.1.5 DCDC Register 3 (REG3)

18.7.1.5.1 Offset

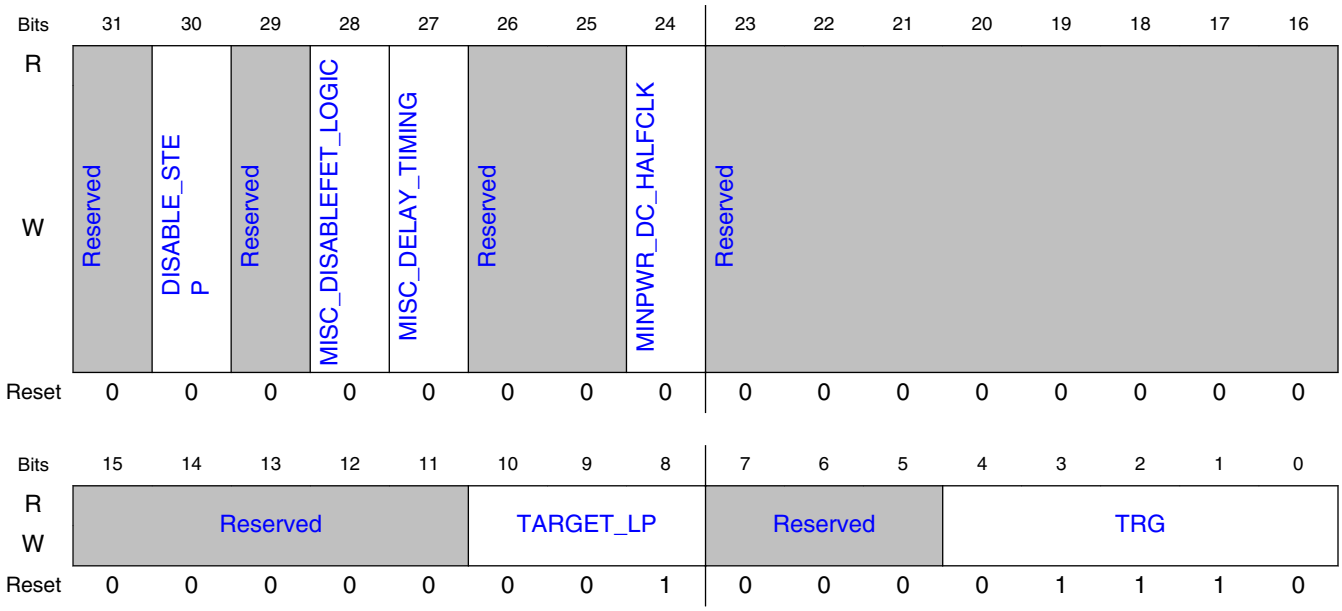
Register	Offset
REG3	Ch

18.7.1.5.2 Function

DCDC register 3

This register controls various high-level functions of the DCDC

18.7.1.5.3 Diagram



18.7.1.5.4 Fields

Field	Function
31 —	Reserved.
30 DISABLE_STEP	Disable stepping for the output VDD_SOC of DCDC
29 —	Reserved.
28 MISC_DISABLEFET_LOGIC	Reserved
27 MISC_DELAY_TIMING	Ajust delay to reduce ground noise
26-25 —	Reserved.
24 MINPWR_DC_HALFCLK	Set DCDC clock to half freqeuncy for continuous mode
23-11 —	Reserved.
10-8	Target value of standby (low power) mode • 0x0: 0.9 V

Table continues on the next page...

Field	Function
TARGET_LP	• 0x1: 0.925 V • 0x2: 0.95 V • 0x3: 0.975 V • 0x4: 1.0 V
7-5 —	Reserved.
4-0 TRG	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.

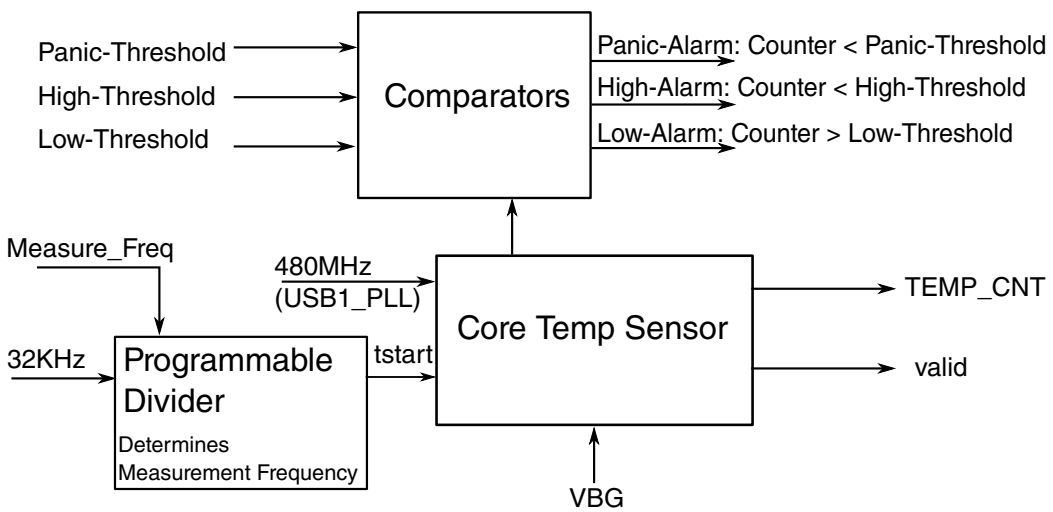


Figure 19-1. High Level Temp Sensor System Diagram

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.

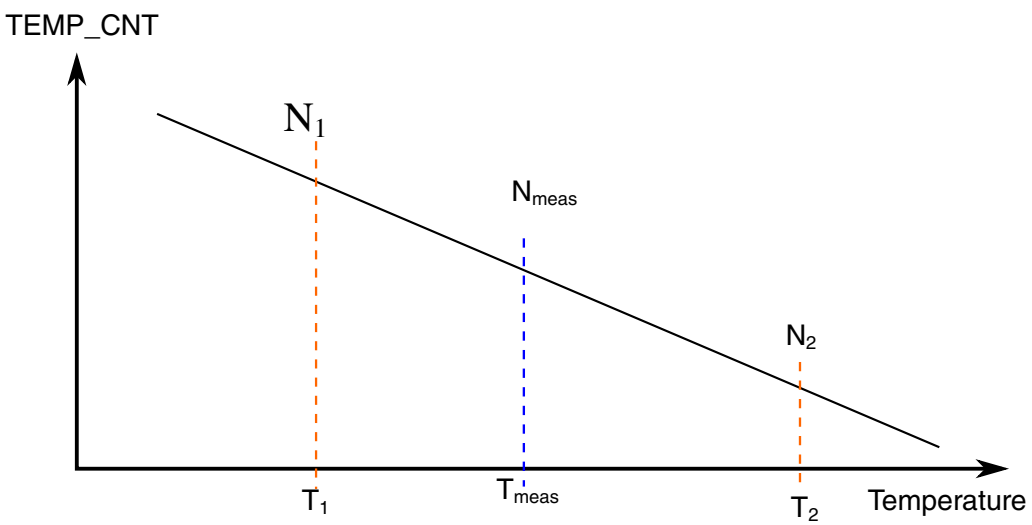


Figure 19-2. Temperature Measurement Cycle

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	19.4.1/698
400D_8184	Tempsensor Control Register 0 (TEMPMON_TEMPSENSE0_SET)	32	R/W	0000_0001h	19.4.1/698
400D_8188	Tempsensor Control Register 0 (TEMPMON_TEMPSENSE0_CLR)	32	R/W	0000_0001h	19.4.1/698
400D_818C	Tempsensor Control Register 0 (TEMPMON_TEMPSENSE0_TOG)	32	R/W	0000_0001h	19.4.1/698
400D_8190	Tempsensor Control Register 1 (TEMPMON_TEMPSENSE1)	32	R/W	0000_0001h	19.4.2/700
400D_8194	Tempsensor Control Register 1 (TEMPMON_TEMPSENSE1_SET)	32	R/W	0000_0001h	19.4.2/700
400D_8198	Tempsensor Control Register 1 (TEMPMON_TEMPSENSE1_CLR)	32	R/W	0000_0001h	19.4.2/700
400D_819C	Tempsensor Control Register 1 (TEMPMON_TEMPSENSE1_TOG)	32	R/W	0000_0001h	19.4.2/700
400D_8290	Tempsensor Control Register 2 (TEMPMON_TEMPSENSE2)	32	R/W	0000_0000h	19.4.3/700

Table continues on the next page...

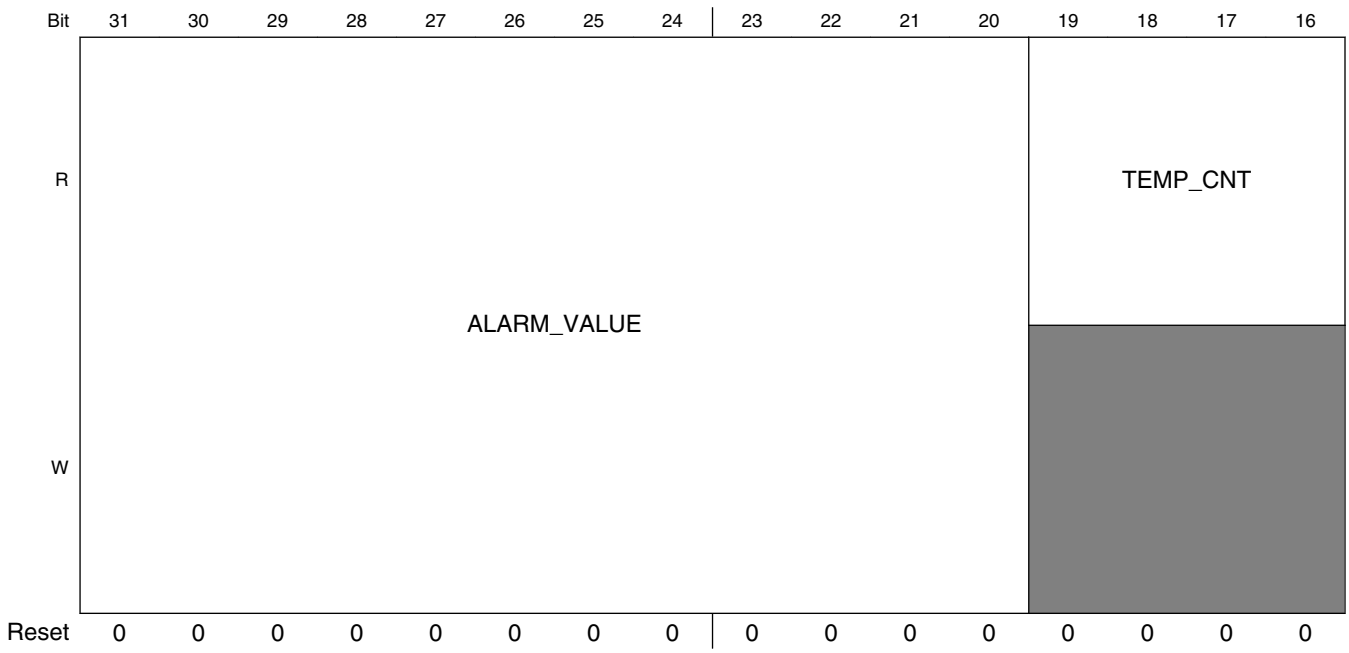
TEMPMON memory map (continued)

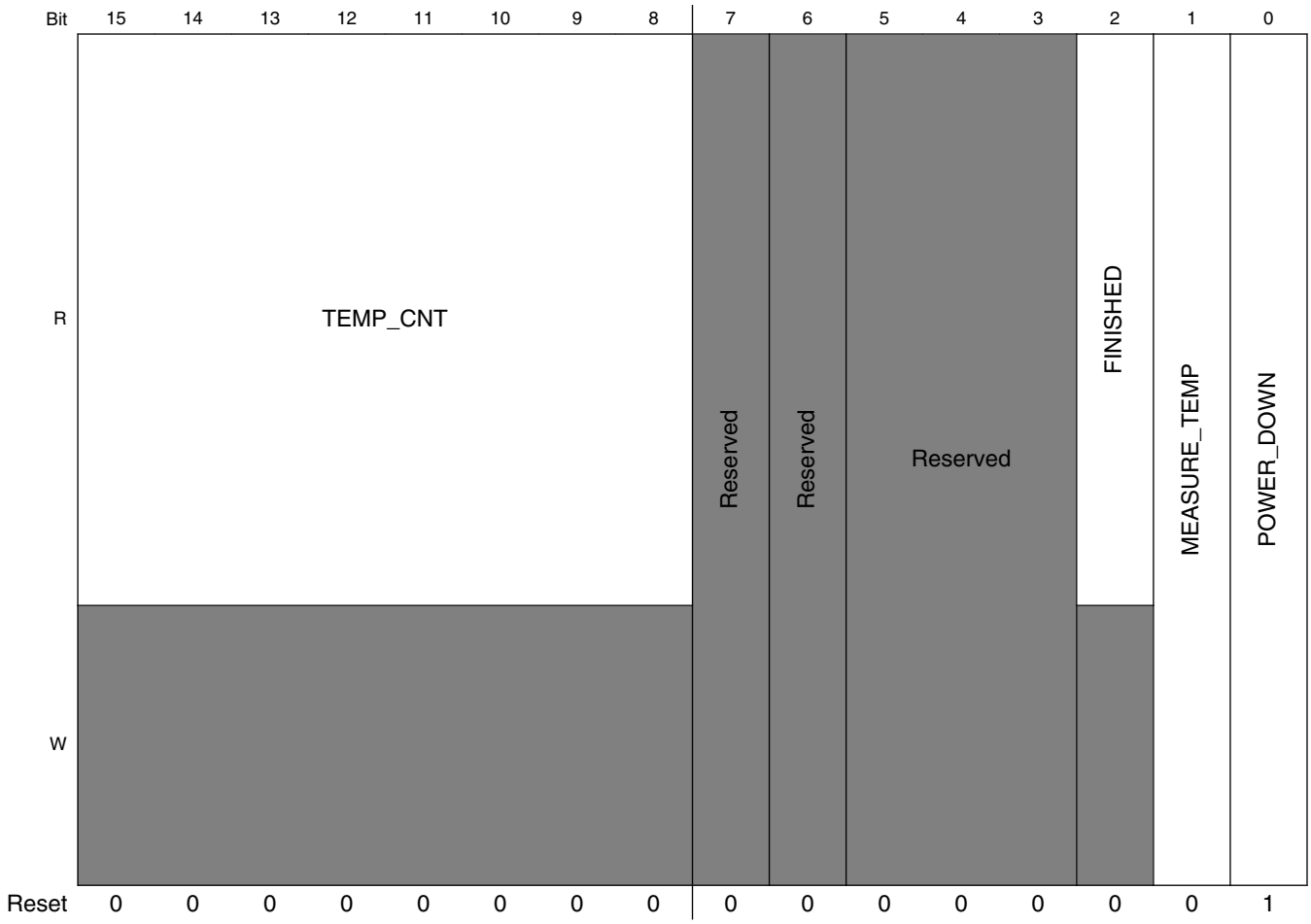
Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400D_8294	Tempsensor Control Register 2 (TEMPMON_TEMPSENSE2_SET)	32	R/W	0000_0000h	19.4.3/700
400D_8298	Tempsensor Control Register 2 (TEMPMON_TEMPSENSE2_CLR)	32	R/W	0000_0000h	19.4.3/700
400D_829C	Tempsensor Control Register 2 (TEMPMON_TEMPSENSE2_TOG)	32	R/W	0000_0000h	19.4.3/700

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

NOTE

Tamper Detection feature and ZMK hardware programming are not applicable on this device.

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

20.2 SNVS introduction

SNVS is a companion module to the DCP module.

SNVS incorporates both security and non-security functionality. The SNVS non-security functionality is described in this document, but the SNVS security functionality is described only in the Security Reference Manual.

SNVS non-security functions:

- Realtime Counter (RTC) - a software accessible realtime counter
  - RTC can be set to the value in the SRTC
- Periodic Interrupt - a hardware-generated interrupt that occurs periodically at software-specified frequency

- General Purpose Register - a set of registers used to hold 128 bits of data specified by software
  - If the SNVS_LP power input is connected to an uninterrupted power supply, e.g. a coin cell battery, the GPR value is maintained when main SoC is powered off
- Chip power-on/power-off - If the SNVS_LP power input is connected to an uninterrupted power supply and the Power On button input signal is connected to a power button external to the chip, logic within SNVS_LP can be used to wake the chip from a power down.

20.2.1 SNVS feature list

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
Monotonic counter	• The monotonic counter can only increment. • The monotonic counter does not rollover. Instead the monotonic counter logic issues an alarm if the monotonic counter reaches its maximum value. • The monotonic counter value is marked as invalid if an SNVS tamper event is detected. • If the SNVS_LP power input is connected to an uninterrupted power supply (see xref href="snvs_power_domains.dita#power_domains"/>), the monotonic counter value is retained even if the main chip is powered down.	Using the Monotonic Counter (MC)
General-purpose register	• The general-purpose register is available to software to store 128 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 xref href="snvs_power_domains.dita#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	• Registers can be programmed only when the SNVS is in functional state, i.e. not in scan mode. • 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

20.2.2 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) (Npte 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 SNVS Structure

SNVS is organized as two major sub-modules:

- Low-Power Section of SNVS (SNVS_LP)

  The SNVS_LP section provides hardware that enables secure storage and protection of sensitive data. The SNVS module is designed to safely hold security-related data such as cryptographic key, time counter, monotonic counter, and general purpose security information.

  The SNVS_LP block implements the following functional units:
  - Control and Status Registers
  - General Purpose Registers

  When the LP section is powered by a backup battery the state of these registers is maintained even when the main chip power is off. (see SNVS power domains)

- High-Power Section of SNVS (SNVS_HP)

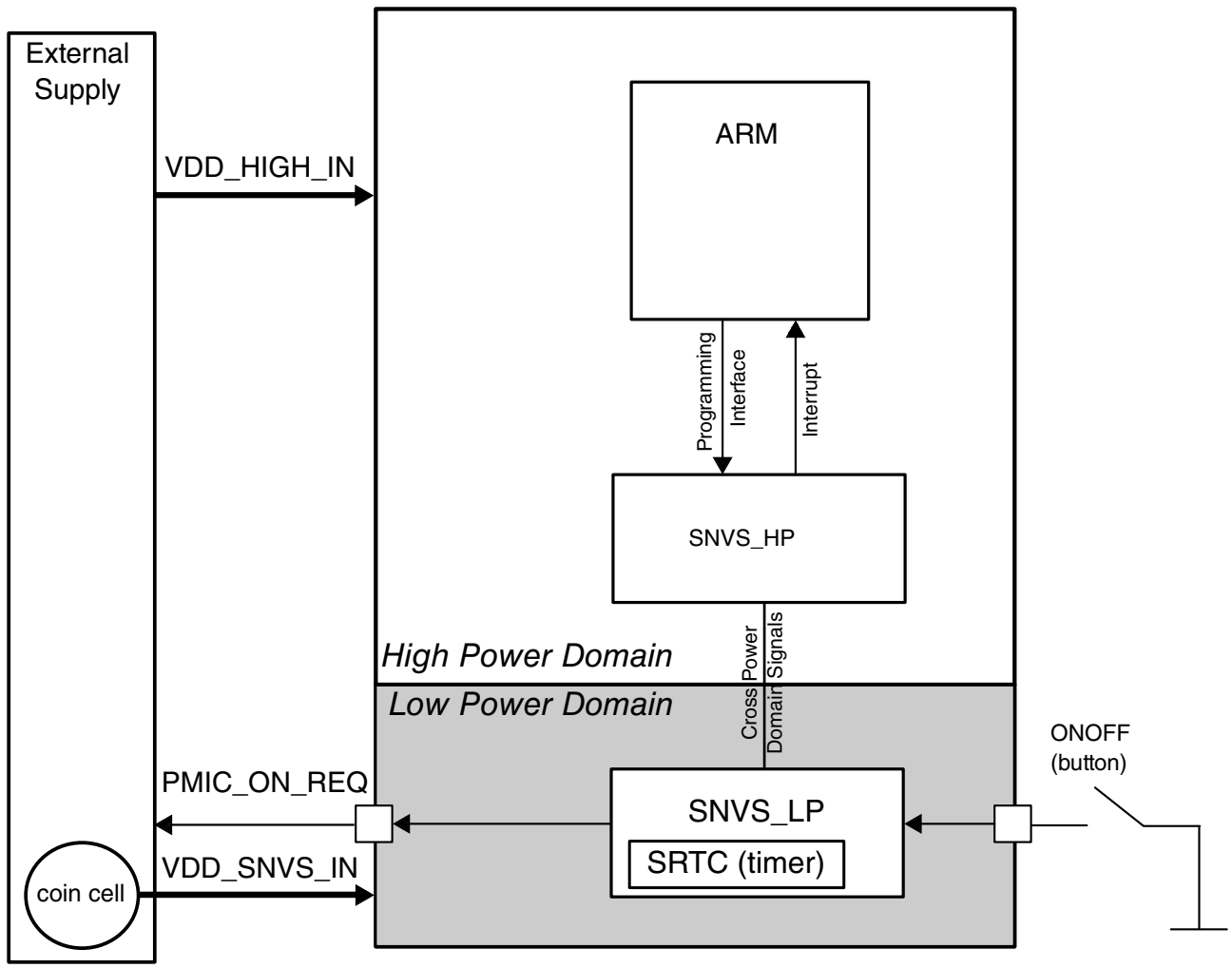
  The SNVS_HP section contains all SNVS status and configuration registers. It implements all features that enable system communication and provisioning of the SNVS_LP section.

  The SNVS_HP provides an interface between SNVS_LP and the rest of the system.

  The SNVS_HP block implements the following functional units:
  - IP Bus Interface
  - SNVS_LP Interface
  - Zeroizable Master Key Programming Mechanism
  - Real Time Counter with Alarm Control and Status Registers
  - Control and status registers

SNVS_HP is in the chip's power supply domain and thus receives power along with the rest of the chip.

The following figure illustrates the structure 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 coin-cell battery, with possibly some power management logic to power the LP section from main power (and perhaps recharge the coin cell battery) when main power is on, and switch to coin-cell 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 coin cell 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 SNVS clock sources

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.

20.4 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)
- Secure Real Time Clock (see SNVS_LP Secure Real Time Counter (SRTC))
- General Purpose Register (see Using the General-Purpose Register)
- Monotonic Counter (see Using the Monotonic Counter (MC))

Procedures for using these features are described in the following sections.

20.4.1 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.

20.4.1.1 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 (e.g. a coin cell battery) because the RTC can then be set from a chip-internal time source.

20.4.1.2 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		

Table continues on the next page...

Table 20-3. RTC/SRTC synchronized values list (continued)

Function	Value/register	Control bit setting
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.

20.4.1.3 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

20.4.2 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.

20.4.2.1 Using the General-Purpose Register

SNVS implements a 128-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 legacy 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 (e.g. coin cell battery).

20.5 Reset and Initialization of SNVS

SNVS is implemented in two sections (HP and LP) that both must be initialized by software following POR. 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 and monitors the SNVS_LP tamper detection inputs, 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 whenever 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):

1. Program the Power Glitch Detector and clear the power glitch record in the LP status register
2. If the SRTC will be used, set the Secure Real Time Clock
3. If the ZMK will be used, provision the ZMK
4. If the Monotonic Counter will be used, burn an additional Monotonic Counter Era bit in the fuse bank and reset the Monotonic Counter

Initializing the HP section The following steps should be completed to properly initialize the SNVS HP section (following successful completion of secure boot):

1. Perform normal or secure boot to put the SNVS into a functional state (Non-secure, Trusted, Secure)
2. At SNVS_LP POR a power glitch violation will be asserted. Therefore at the first HP POR following an LP POR software should write the proper initialization value (41736166h) into the LPPGDR and clear the power glitch record in the LP status register. See Power glitch detector (PGD) for more details.
3. Transition SSM from trusted state to secure state (if not already done by HAB).
4. Enable security violations and interrupts in HPSVCR and HPSICR registers
5. Program SNVS general functions/configurations
6. Select Master Key (OTPMK, ZMK or XOR of the two. Default is OTPMK.)
7. Set lock bits. The ms bit of the SNVS_HP lock register should be set before starting any functional operation. Setting this bit prevents further changes to the Master Key selection.

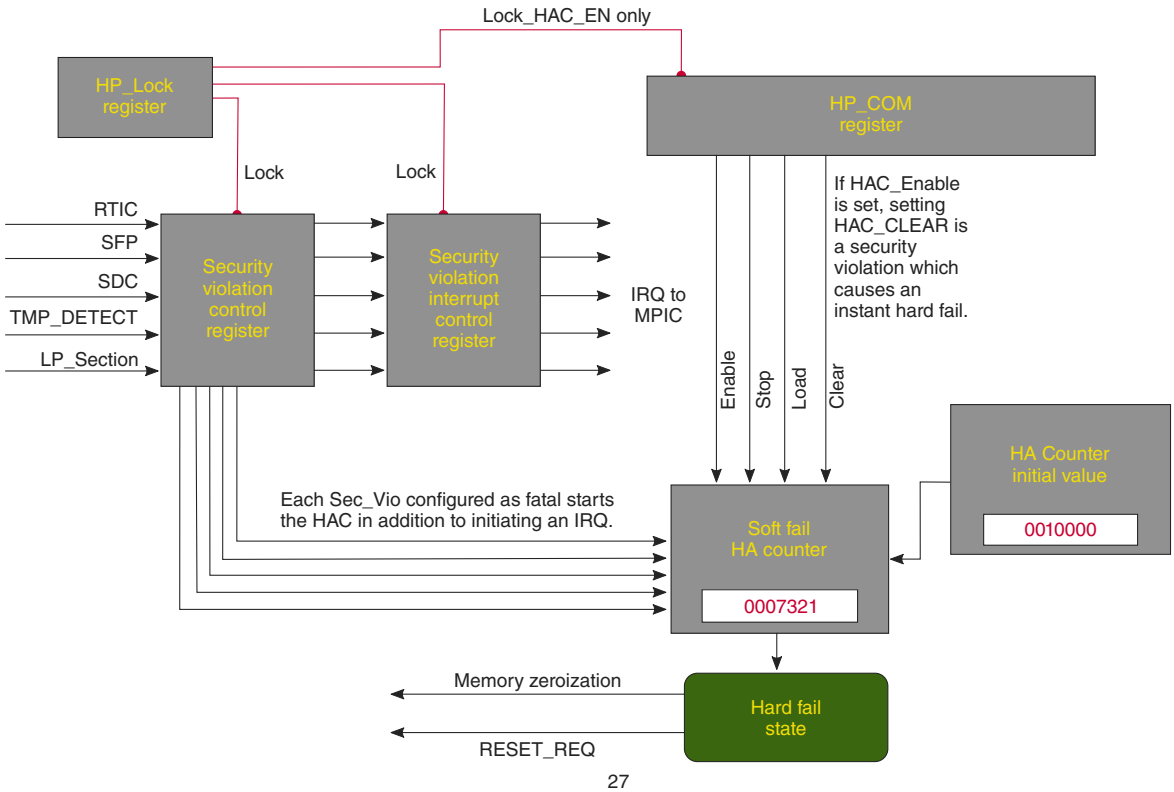


Figure 20-1. Relationship Between the Registers

20.5.1 Initialization Checklists

SNVS is implemented in two sections (HP and LP) that both must be initialized by software following POR. 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 and monitors the SNVS_LP tamper detection inputs, 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 whenever 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.

20.6 SNVS register descriptions

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 for mxrt.

20.6.1 SNVS Memory map

SNVS base address: 400D_4000h

Offset	Register	Width (In bits)	Access	Reset value
0h	SNVS_HP Lock Register (HPLR)	32	RW	0000_0000h
4h	SNVS_HP Command Register (HPCOMR)	32	RW	0000_0000h
8h	SNVS_HP Control Register (HPCR)	32	RW	0000_0000h
14h	SNVS_HP Status Register (HPSR)	32	RW	8000_0000h
24h	SNVS_HP Real Time Counter MSB Register (HPRTCMR)	32	RW	0000_0000h
28h	SNVS_HP Real Time Counter LSB Register (HPRTCLR)	32	RW	0000_0000h
2Ch	SNVS_HP Time Alarm MSB Register (HPTAMR)	32	RW	0000_0000h
30h	SNVS_HP Time Alarm LSB Register (HPTALR)	32	RW	0000_0000h
34h	SNVS_LP Lock Register (LPLR)	32	RW	0000_0000h
38h	SNVS_LP Control Register (LPCR)	32	RW	0000_0020h
4Ch	SNVS_LP Status Register (LPSR)	32	RW	0000_0008h
5Ch	SNVS_LP Secure Monotonic Counter MSB Register (LPSMCMR)	32	RO	0000_0000h
60h	SNVS_LP Secure Monotonic Counter LSB Register (LPSMCLR)	32	RO	0000_0000h
68h	SNVS_LP General Purpose Register 0 (legacy alias) (LPGPR0_legacy_alias)	32	RW	0000_0000h
90h - 9Ch	SNVS_LP General Purpose Registers 0 .. 3 (LPGPR0_alias - LPGPR3_alias)	32	RW	0000_0000h
100h - 10Ch	SNVS_LP General Purpose Registers 0 .. 3 (LPGPR0 - LPGPR3)	32	RW	0000_0000h
BF8h	SNVS_HP Version ID Register 1 (HPVIDR1)	32	RO	003E_0104h
BFCh	SNVS_HP Version ID Register 2 (HPVIDR2)	32	RO	0600_0000h

20.6.2 SNVS_HP Lock Register (HPLR)

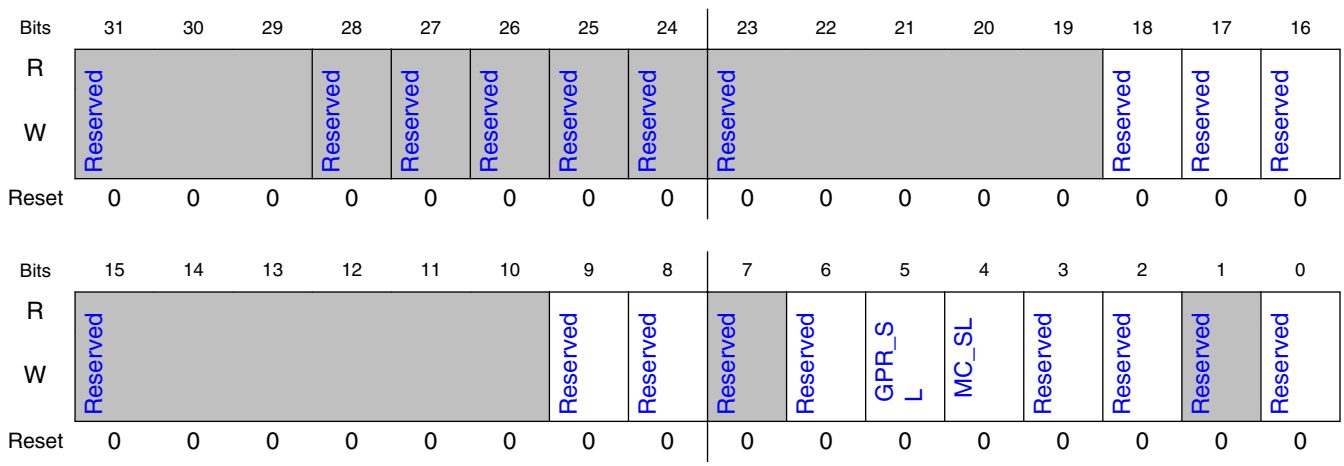
20.6.2.1 Offset

Register	Offset
HPLR	0h

20.6.2.2 Function

The SNVS_HP Lock Register contains lock bits for the SNVS registers. This is a privileged write register.

20.6.2.3 Diagram



20.6.2.4 Fields

Field	Function
31-29 —	Reserved.
28 —	Reserved.
27 —	Reserved.
26 —	Reserved.
25 —	Reserved.
24 —	Reserved.
23-19 —	Reserved.
18	Reserved

Table continues on the next page...

Field	Function
—	
17 —	Reserved
16 —	Reserved
15-10 —	Reserved.
9 —	Reserved
8 —	Reserved
7 —	Reserved
6 —	Reserved
5 GPR_SL	General Purpose Register Soft Lock When set, prevents any writes to the GPR. Once set, this bit can only be reset by the system reset. 0b - Write access is allowed 1b - Write access is not allowed
4 MC_SL	Monotonic Counter Soft 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 system reset. 0b - Write access (increment) is allowed 1b - Write access (increment) is not allowed
3 —	Reserved
2 —	Reserved
1 —	Reserved
0 —	Reserved

20.6.3 SNVS_HP Command Register (HPCOMR)

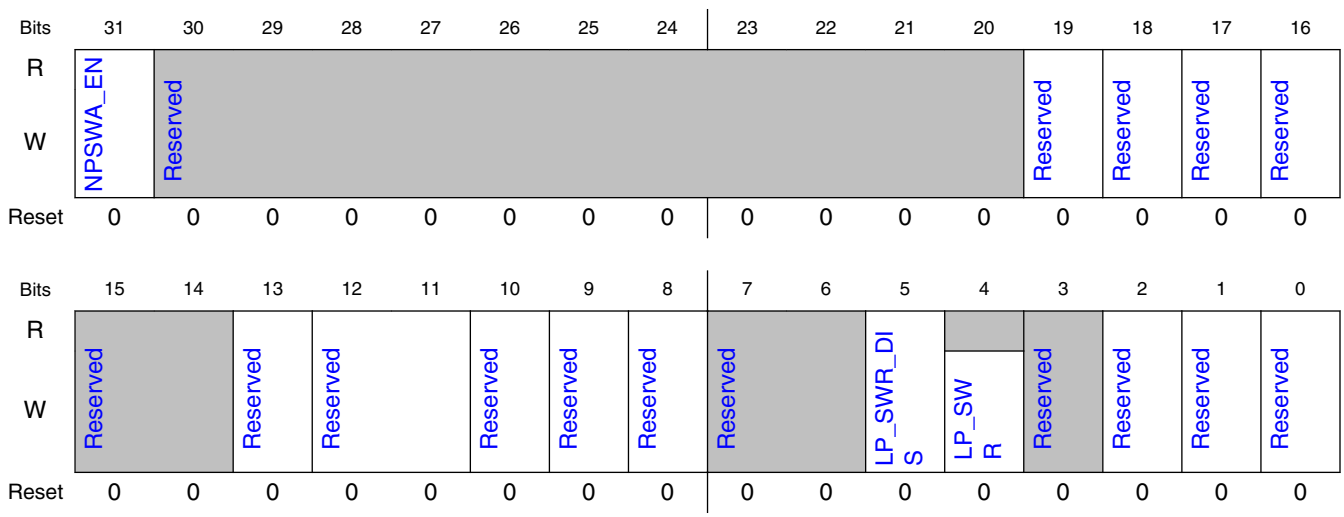
20.6.3.1 Offset

Register	Offset
HPCOMR	4h

20.6.3.2 Function

The SNVS_HP Command Register contains the command, configuration, and control bits for the SNVS block. This is a privileged write register.

20.6.3.3 Diagram



20.6.3.4 Fields

Field	Function
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.

Table continues on the next page...

Field	Function
19 —	Reserved
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. 0b - LP software reset is enabled 1b - LP software reset is disabled
4 LP_SWR	LP Software Reset When set to 1, the registers in the SNVS_LP section are reset. 0b - No Action 1b - Reset LP section
3 —	Reserved.
2 —	Reserved
1 —	Reserved
0 —	Reserved

20.6.4 SNVS_HP Control Register (HPCR)

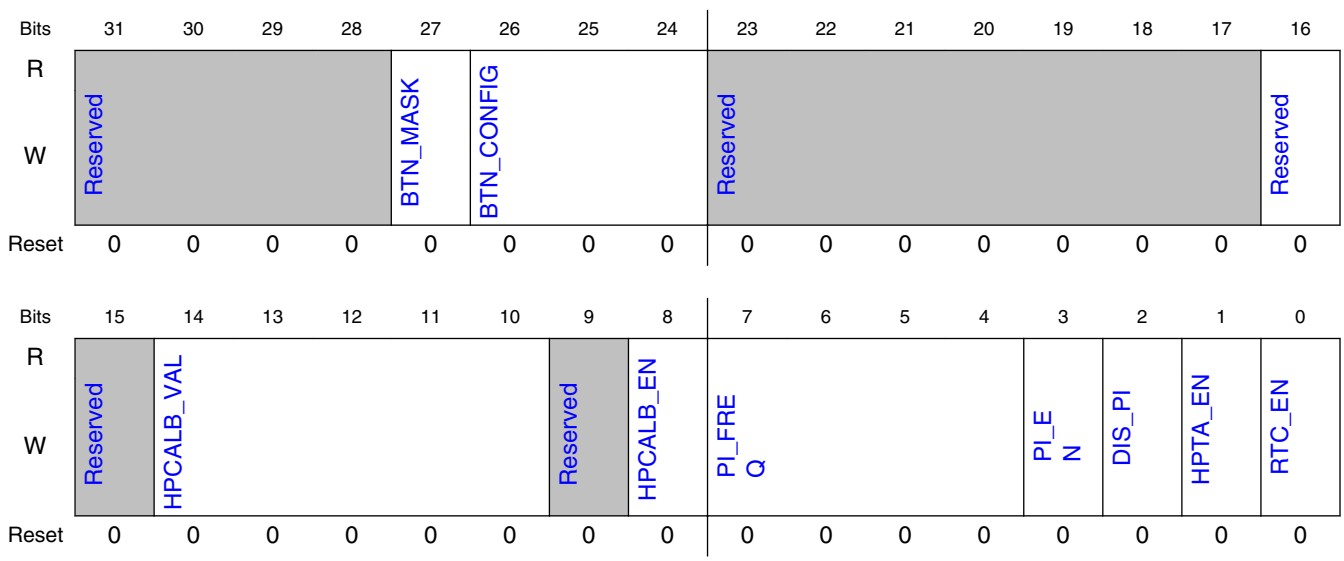
20.6.4.1 Offset

Register	Offset
HPCR	8h

20.6.4.2 Function

The SNVS_HP Control Register contains various control bits of the HP section of SNVS. This is *not* a privileged write register.

20.6.4.3 Diagram



20.6.4.4 Fields

Field	Function
31-28 —	Reserved.

Table continues on the next page...

Field	Function
27 BTN_MASK	Button interrupt mask. This bit is used to mask the ipi_snvs_btn_int_b (button) 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". 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. 00000b - +0 counts per each 32768 ticks of the counter 00001b - +1 counts per each 32768 ticks of the counter 00010b - +2 counts per each 32768 ticks of the counter 01111b - +15 counts per each 32768 ticks of the counter 10000b - -16 counts per each 32768 ticks of the counter 10001b - -15 counts per each 32768 ticks of the counter 11110b - -2 counts per each 32768 ticks of the counter 11111b - -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. 0b - HP Timer calibration disabled 1b - 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: 0000b - - bit 0 of the HPRTCLR is selected as a source of the periodic interrupt 0001b - - bit 1 of the HPRTCLR is selected as a source of the periodic interrupt 0010b - - bit 2 of the HPRTCLR is selected as a source of the periodic interrupt

Table continues on the next page...

Field	Function
	0011b - - bit 3 of the HPRTCLR is selected as a source of the periodic interrupt 0100b - - bit 4 of the HPRTCLR is selected as a source of the periodic interrupt 0101b - - bit 5 of the HPRTCLR is selected as a source of the periodic interrupt 0110b - - bit 6 of the HPRTCLR is selected as a source of the periodic interrupt 0111b - - bit 7 of the HPRTCLR is selected as a source of the periodic interrupt 1000b - - bit 8 of the HPRTCLR is selected as a source of the periodic interrupt 1001b - - bit 9 of the HPRTCLR is selected as a source of the periodic interrupt 1010b - - bit 10 of the HPRTCLR is selected as a source of the periodic interrupt 1011b - - bit 11 of the HPRTCLR is selected as a source of the periodic interrupt 1100b - - bit 12 of the HPRTCLR is selected as a source of the periodic interrupt 1101b - - bit 13 of the HPRTCLR is selected as a source of the periodic interrupt 1110b - - bit 14 of the HPRTCLR is selected as a source of the periodic interrupt 1111b - - 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. 0b - HP Periodic Interrupt is disabled 1b - HP Periodic Interrupt is enabled
2 DIS_PI	Disable periodic interrupt in the functional interrupt 0b - Periodic interrupt will trigger a functional interrupt 1b - 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. 0b - HP Time Alarm Interrupt is disabled 1b - 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. 0b - RTC is disabled 1b - RTC is enabled

20.6.5 SNVS_HP Status Register (HPSR)

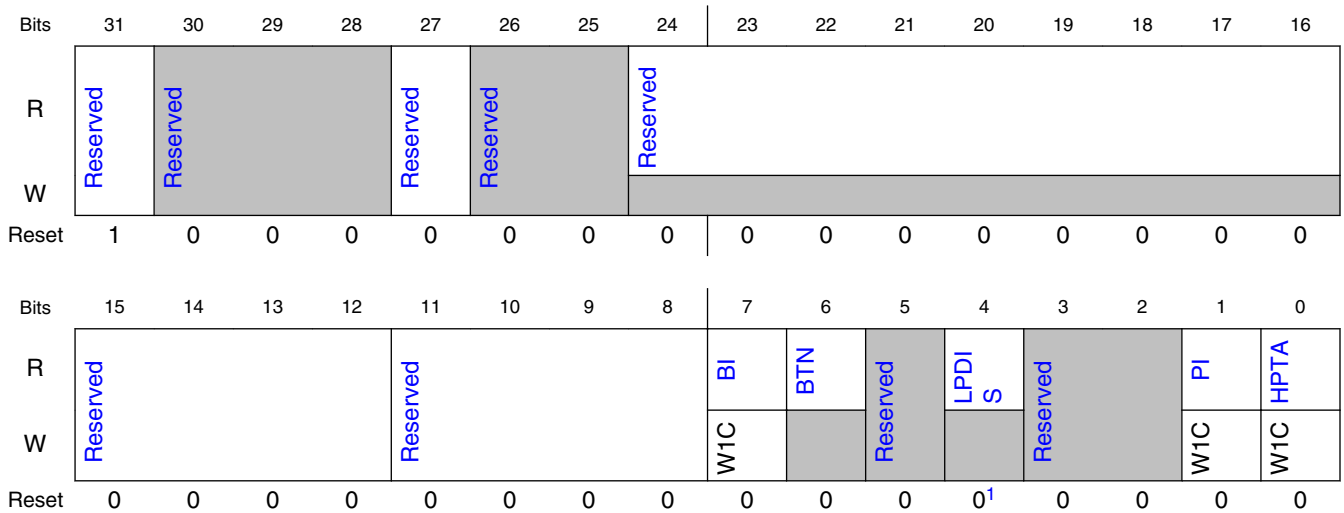
20.6.5.1 Offset

Register	Offset
HPSR	14h

20.6.5.2 Function

The HP Status Register reflects the internal state of the SNVS. This is *not* a privileged write register.

20.6.5.3 Diagram



1. The value of Low Power Disable is determined by the *no_battery* input signal to SNVS.

20.6.5.4 Fields

Field	Function
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.

Table continues on the next page...

Field	Function
	0: BTN not pressed 1: BTN pressed
5 —	Reserved.
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. 0b - No periodic interrupt occurred. 1b - A periodic interrupt occurred.
0 HPTA	HP Time Alarm Indicates that the HP Time Alarm has occurred since this bit was last cleared. 0b - No time alarm interrupt occurred. 1b - A time alarm interrupt occurred.

20.6.6 SNVS_HP Real Time Counter MSB Register (HPRTCMR)

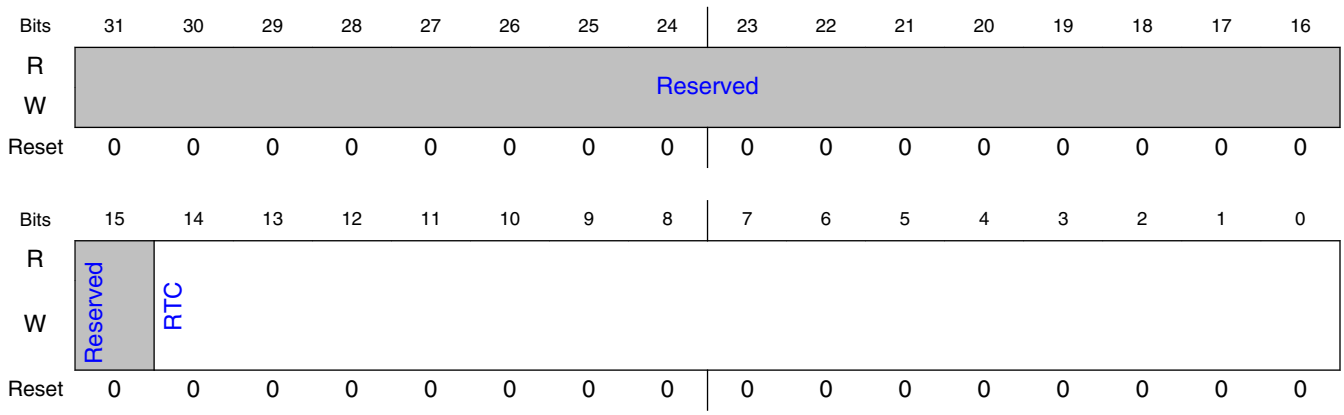
20.6.6.1 Offset

Register	Offset
HPRTCMR	24h

20.6.6.2 Function

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.

20.6.6.3 Diagram



20.6.6.4 Fields

Field	Function
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).

20.6.7 SNVS_HP Real Time Counter LSB Register (HPRTCLR)

20.6.7.1 Offset

Register	Offset
HPRTCLR	28h

20.6.7.2 Function

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.

20.6.7.3 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

20.6.7.4 Fields

Field	Function
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).

20.6.8 SNVS_HP Time Alarm MSB Register (HPTAMR)

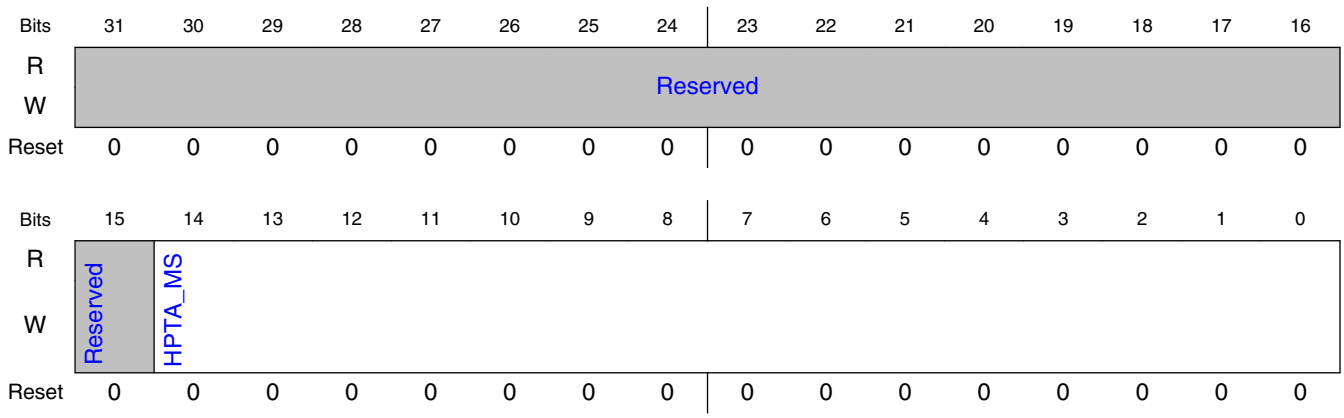
20.6.8.1 Offset

Register	Offset
HPTAMR	2Ch

20.6.8.2 Function

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.

20.6.8.3 Diagram



20.6.8.4 Fields

Field	Function
31-15 —	Reserved.
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).

20.6.9 SNVS_HP Time Alarm LSB Register (HPTALR)

20.6.9.1 Offset

Register	Offset
HPTALR	30h

20.6.9.2 Function

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.

20.6.9.3 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

20.6.9.4 Fields

Field	Function
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).

20.6.10 SNVS_LP Lock Register (LPLR)

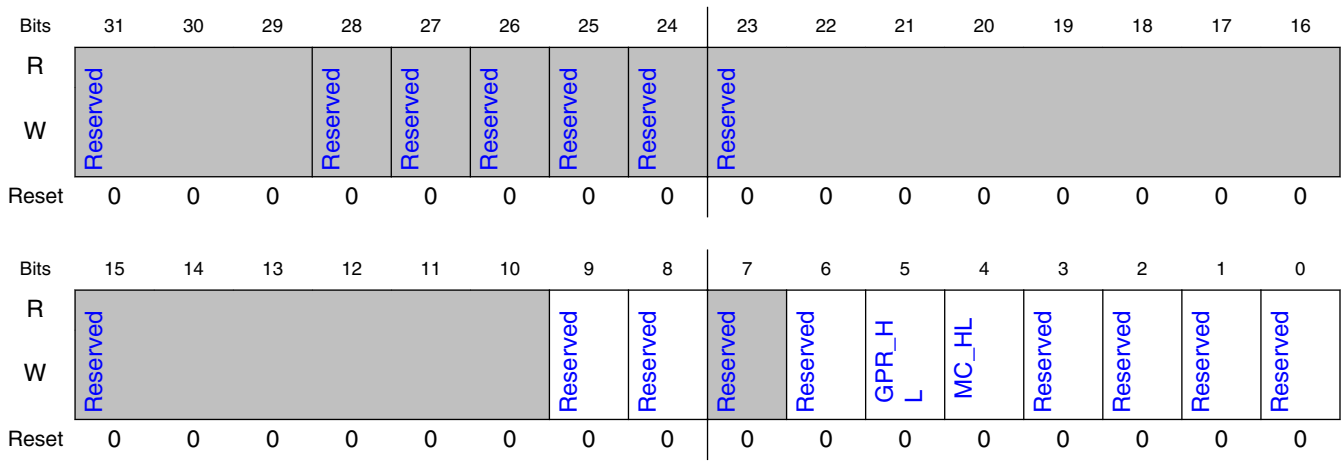
20.6.10.1 Offset

Register	Offset
LPLR	34h

20.6.10.2 Function

The SNVS_LP Lock Register contains lock bits for the SNVS_LP registers. This is a privileged write register.

20.6.10.3 Diagram



20.6.10.4 Fields

Field	Function
31-29 —	Reserved.
28 —	Reserved.
27 —	Reserved.
26 —	Reserved.
25 —	Reserved.
24 —	Reserved.
23-10 —	Reserved.
9 —	Reserved
8 —	Reserved
7 —	Reserved.
6	Reserved

Table continues on the next page...

Field	Function
—	
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. 0b - Write access is allowed. 1b - 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. 0b - Write access (increment) is allowed. 1b - Write access (increment) is not allowed.
3 —	Reserved
2 —	Reserved
1 —	Reserved
0 —	Reserved

20.6.11 SNVS_LP Control Register (LPCR)

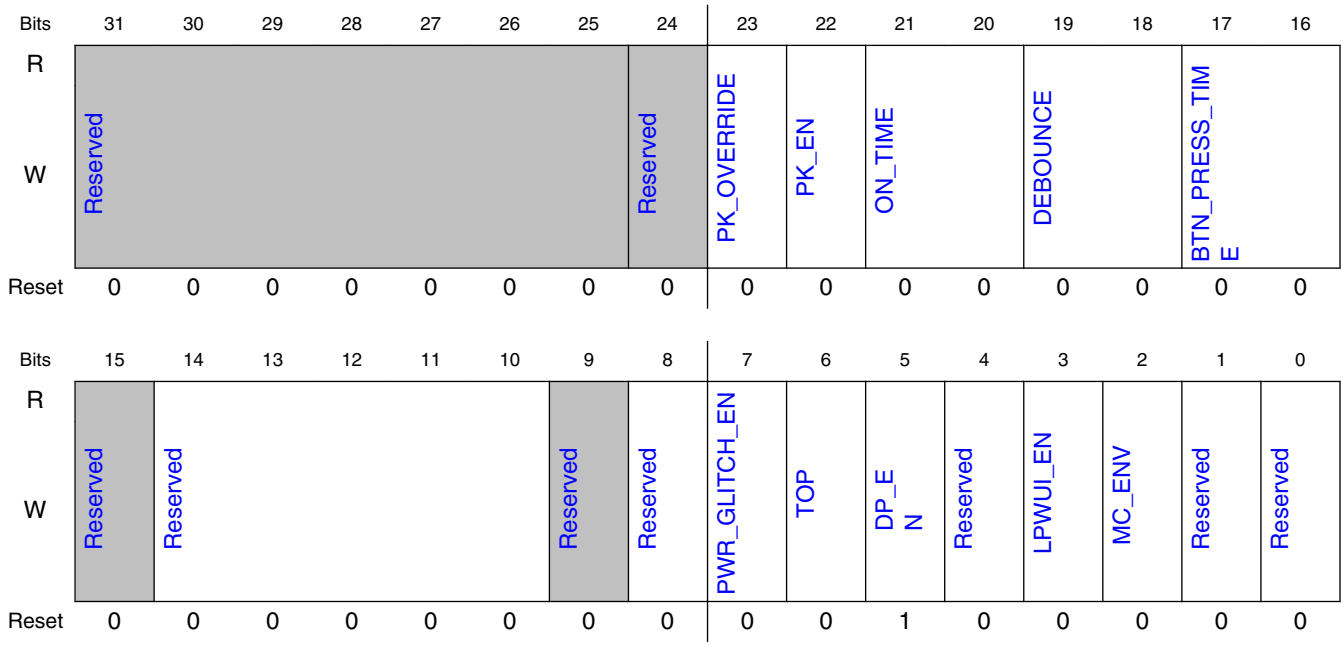
20.6.11.1 Offset

Register	Offset
LPCR	38h

20.6.11.2 Function

The SNVS_LP Control Register contains various control bits of the LP section of SNVS. This is a privileged write register.

20.6.11.3 Diagram



20.6.11.4 Fields

Field	Function
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 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

Table continues on the next page...

Field	Function
	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 PWR_GLITCH_EN	Power Glitch Enable By default the detection of a power glitch does not cause the pmic_en_b signal to be asserted. Setting the Power Glitch Enable bit to 1 enables the power glitch 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. 0b - Leave system power on. 1b - 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. 0b - Smart PMIC enabled. 1b - 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 (tamper 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. 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 software reset or LP POR.

Table continues on the next page...

Field	Function
	0b - MC is disabled or invalid. 1b - MC is enabled and valid.
1 —	Reserved
0 —	Reserved

20.6.12 SNVS_LP Status Register (LPSR)

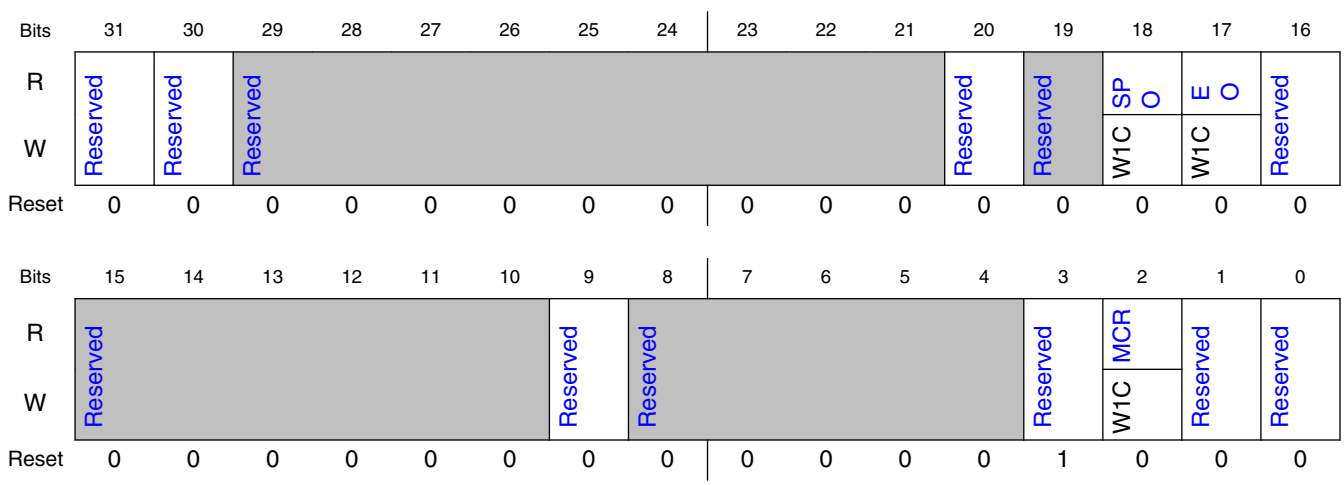
20.6.12.1 Offset

Register	Offset
LPSR	4Ch

20.6.12.2 Function

The SNVS_LP Status Register reflects the internal state and behavior of the SNVS_LP. This is a privileged write register.

20.6.12.3 Diagram



20.6.12.4 Fields

Field	Function
31 —	Reserved
30 —	Reserved
29-21 —	Reserved.
20 —	Reserved
19 —	Reserved.
18 SPO	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. 0b - Set Power Off was not detected. 1b - Set Power Off was detected.
17 EO	Emergency Off This bit is set when a power off is requested. 0b - Emergency off was not detected. 1b - Emergency off was detected.
16 —	Reserved
15-10 —	Reserved.
9 —	Reserved
8-4 —	Reserved.
3 —	Reserved
2 MCR	Monotonic Counter Rollover 0b - MC has not reached its maximum value. 1b - MC has reached its maximum value.
1 —	Reserved
0 —	Reserved

20.6.13 SNVS_LP Secure Monotonic Counter MSB Register (LPSMCMR)

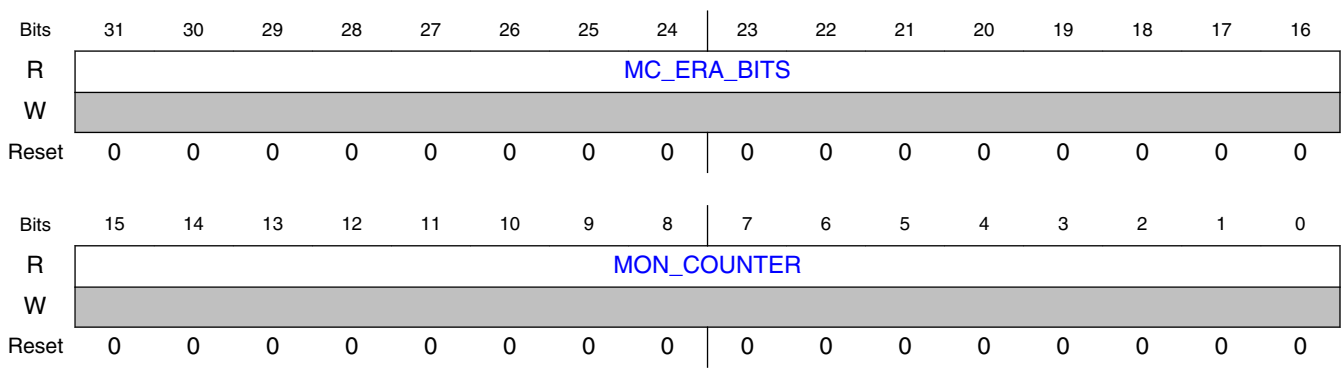
20.6.13.1 Offset

Register	Offset
LPSMCMR	5Ch

20.6.13.2 Function

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 LPSMCR or LPSMCLR register. This is a non-privileged read-only register.

20.6.13.3 Diagram



20.6.13.4 Fields

Field	Function
31-16	Monotonic Counter Era Bits
MC_ERA_BITS	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, e.g. a coin cell battery), 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

Table continues on the next page...

Field	Function
	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. Typically this would be necessary only when the coin cell battery is changed. Since the Monotonic Counter is reset on an LP Software Reset, an excessive number of MC_ERA_BITS fusez may be consumed if LP Software Reset is used repeatedly.
15-0 MON_COUNTER	Monotonic Counter most-significant 16 Bits The MC 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 software reset or LP POR.

20.6.14 SNVS_LP Secure Monotonic Counter LSB Register (LPSMCLR)

20.6.14.1 Offset

Register	Offset
LPSMCLR	60h

20.6.14.2 Function

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.

20.6.14.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	MON_COUNTER															
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

20.6.14.4 Fields

Field	Function
31-0 MON_COUNTER	Monotonic Counter bits The MC 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 software reset or LP POR.

20.6.15 SNVS_LP General Purpose Register 0 (legacy alias) (LPGPR0_legacy_alias)

20.6.15.1 Offset

Register	Offset
LPGPR0_legacy_alias	68h

20.6.15.2 Function

The SNVS_LP General Purpose Register is a 128-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 4 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 aliased at its original offset of 68h. The GPR will be automatically zeroized when a tamper event occurs, unless GPR zeroization is disabled via the GPR_Z_DIS bit in the LP Control Register.

20.6.15.3 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

20.6.15.4 Fields

Field	Function
31-0 GPR	General Purpose Register When GPR_SL or GPR_HL bit is set, the register cannot be programmed.

20.6.16 SNVS_LP General Purpose Registers 0 .. 3 (LPGPR0_alias - LPGPR3_alias)

20.6.16.1 Offset

For a = 0 to 3:

Register	Offset
LPGPRa_alias	90h + (a × 4h)

20.6.16.2 Function

The SNVS_LP General Purpose Register is a 128-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 4 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 aliased at its original offset of 68h. The GPR will be automatically zeroized when a tamper event occurs, unless GPR zeroization is disabled via the GPR_Z_DIS bit in the LP Control Register.

20.6.16.3 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

20.6.16.4 Fields

Field	Function
31-0 GPR	General Purpose Register When GPR_SL or GPR_HL bit is set, the register cannot be programmed.

20.6.17 SNVS_LP General Purpose Registers 0 .. 3 (LPGPR0 - LPGPR3)

20.6.17.1 Offset

For a = 0 to 3:

Register	Offset
LPGPRa	100h + (a × 4h)

20.6.17.2 Function

The SNVS_LP General Purpose Register is a 128-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 4 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 aliased at its original offset of 68h. The GPR will be automatically zeroized when a tamper event occurs, unless GPR zeroization is disabled via the GPR_Z_DIS bit in the LP Control Register.

20.6.17.3 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

20.6.17.4 Fields

Field	Function
31-0 GPR	General Purpose Register When GPR_SL or GPR_HL bit is set, the register cannot be programmed.

20.6.18 SNVS_HP Version ID Register 1 (HPVIDR1)

20.6.18.1 Offset

Register	Offset
HPVIDR1	BF8h

20.6.18.2 Function

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.

20.6.18.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	IP_ID															
W																
Reset	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0

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

20.6.18.4 Fields

Field	Function
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

20.6.19 SNVS_HP Version ID Register 2 (HPVIDR2)

20.6.19.1 Offset

Register	Offset
HPVIDR2	BFCh

20.6.19.2 Function

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.

20.6.19.3 Diagram

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

20.6.19.4 Fields

Field	Function
31-24 IP_ERA	IP Era 00h - Era 1 or 2 03h - Era 3 04h - Era 4 05h - Era 5
23-16 INTG_OPT	SNVS Integration Options
15-8 ECO_REV	SNVS ECO Revision
7-0 CONFIG_OPT	SNVS Configuration Options

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 SRC Overview

The System Reset Controller (SRC) controls the reset and boot operation of the SoC.

It is responsible for the generation of all reset signals and boot decoding.

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.

21.2.1 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 External Signals

The following table describes the external signals of SRC.

Table 21-2. SRC External Signals

Signal	Description	Pad	Mode	Direction
SRC_BT_CFG0	Boot configuration signals	GPIO_SD_02	ALT6	I
SRC_BT_CFG1		GPIO_SD_01	ALT6	I
SRC_BT_CFG2		GPIO_SD_00	ALT6	I
SRC_BT_CFG3		GPIO_SD_13	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_SD_04	ALT6	I
SRC_BOOT_MODE1		GPIO_SD_03	ALT6	I

21.4 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 21-3. SRC Clocks

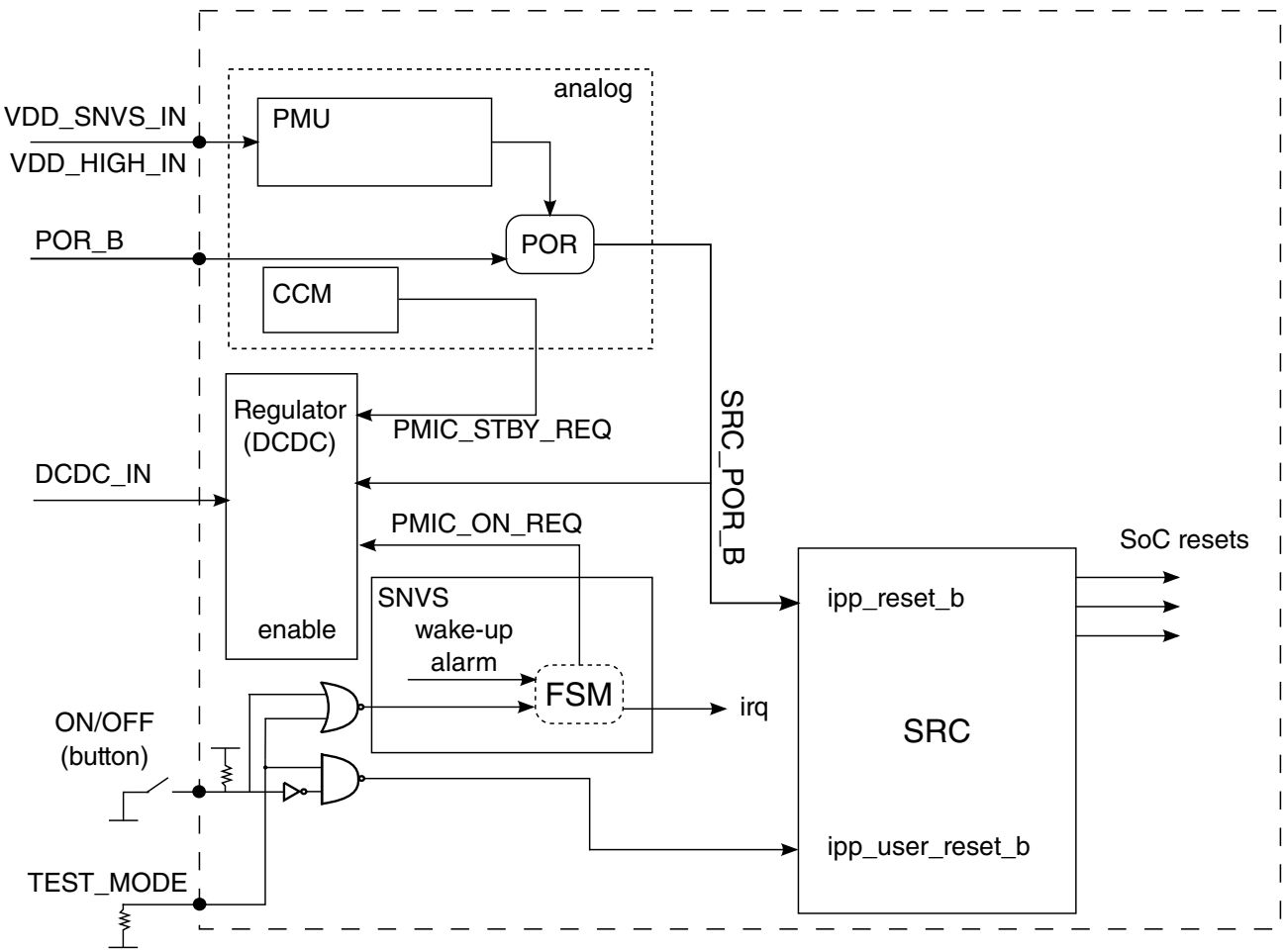
Clock name	Clock Root	Description
ipg_clk	ipg_clk_root	Peripheral clock
ipg_clk_s	ipg_clk_root	Peripheral access clock

21.5 Top-level resets, power-up sequence and external supply integration

Information found here defines chip resets, power-up sequence, and external supply integration.

21.5.1 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

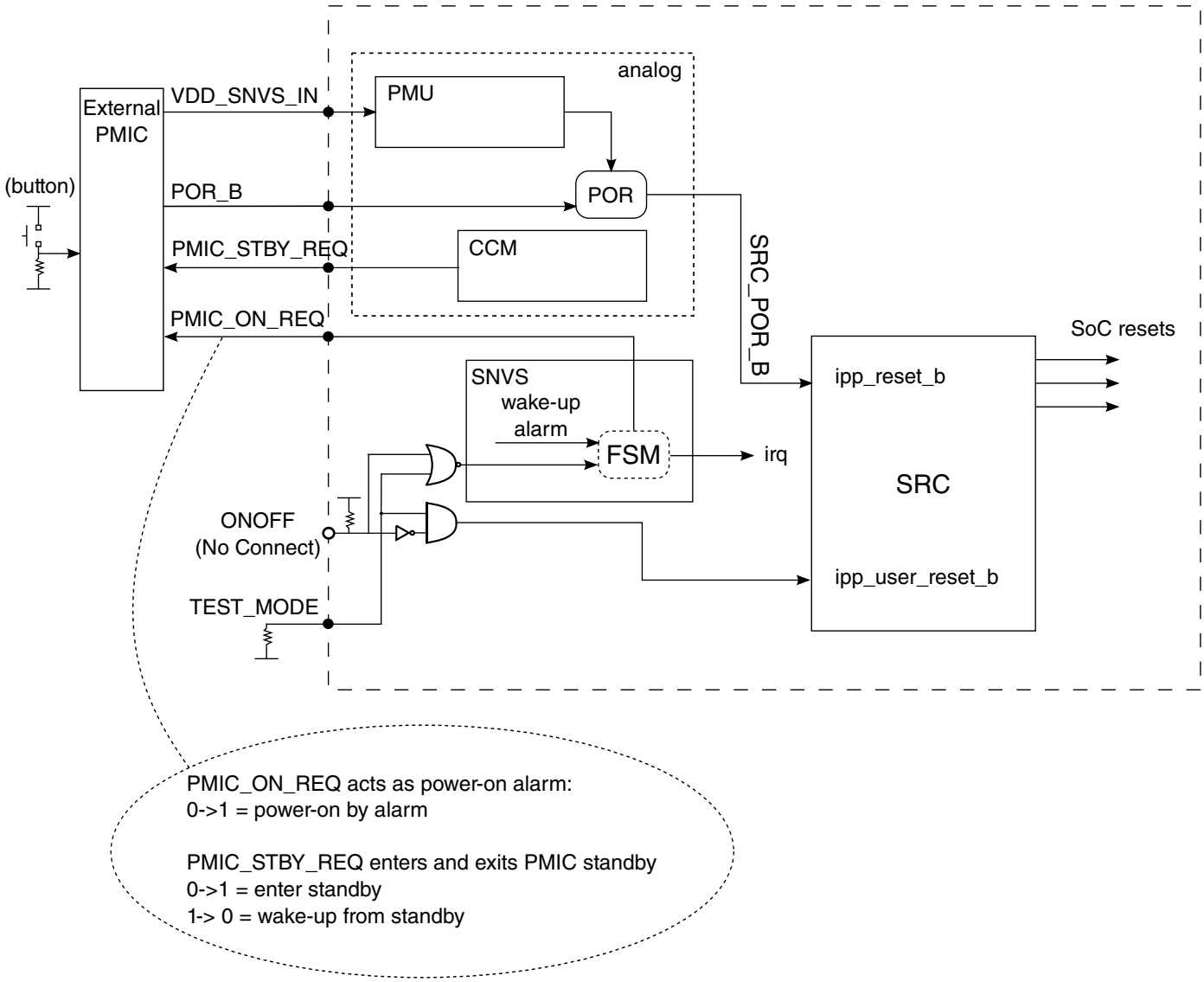


Figure 21-2. Chip reset scheme under external PMIC control

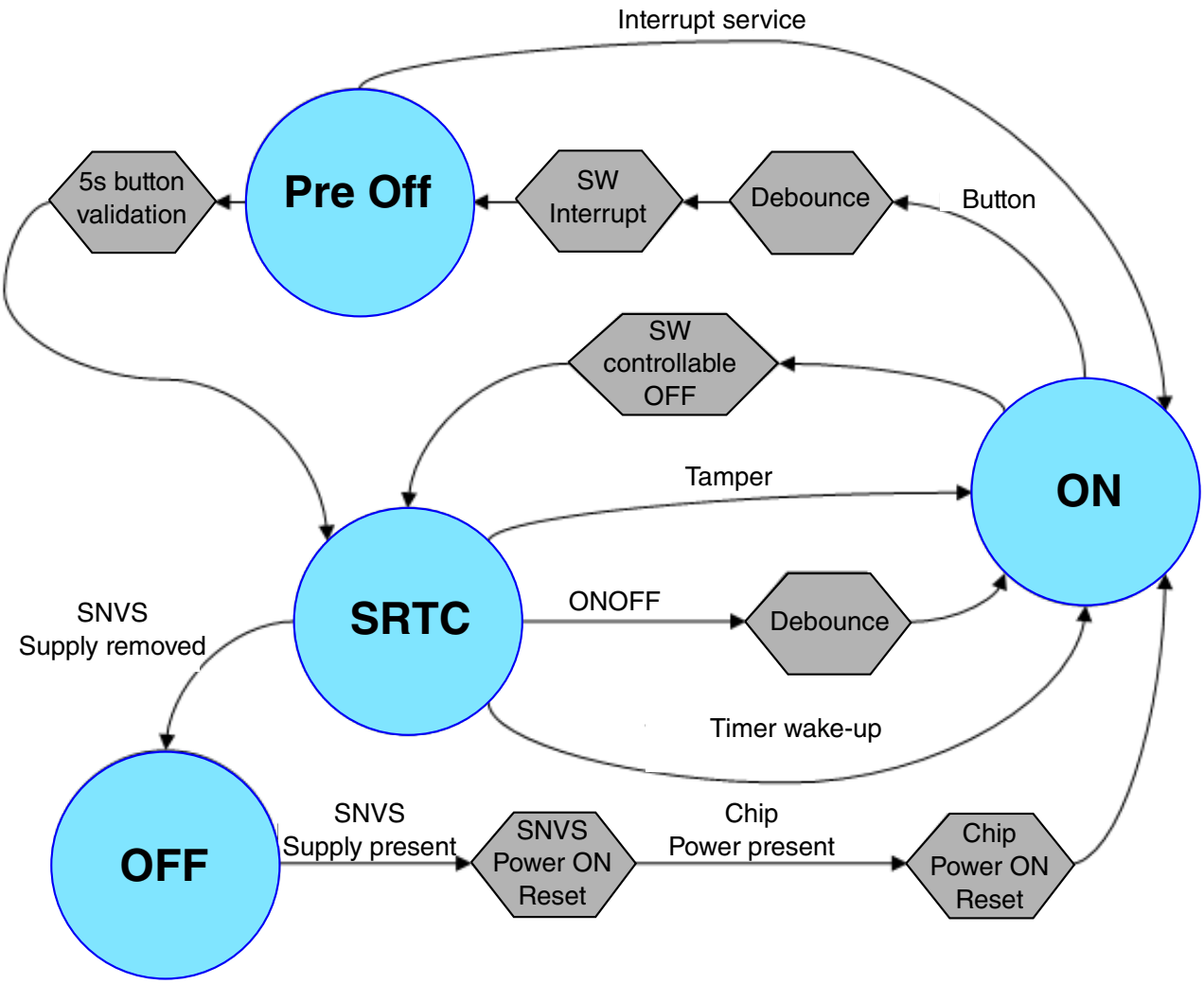


Figure 21-3. Chip on/off state flow diagram

21.5.2 Finite-State Machine (FSM)

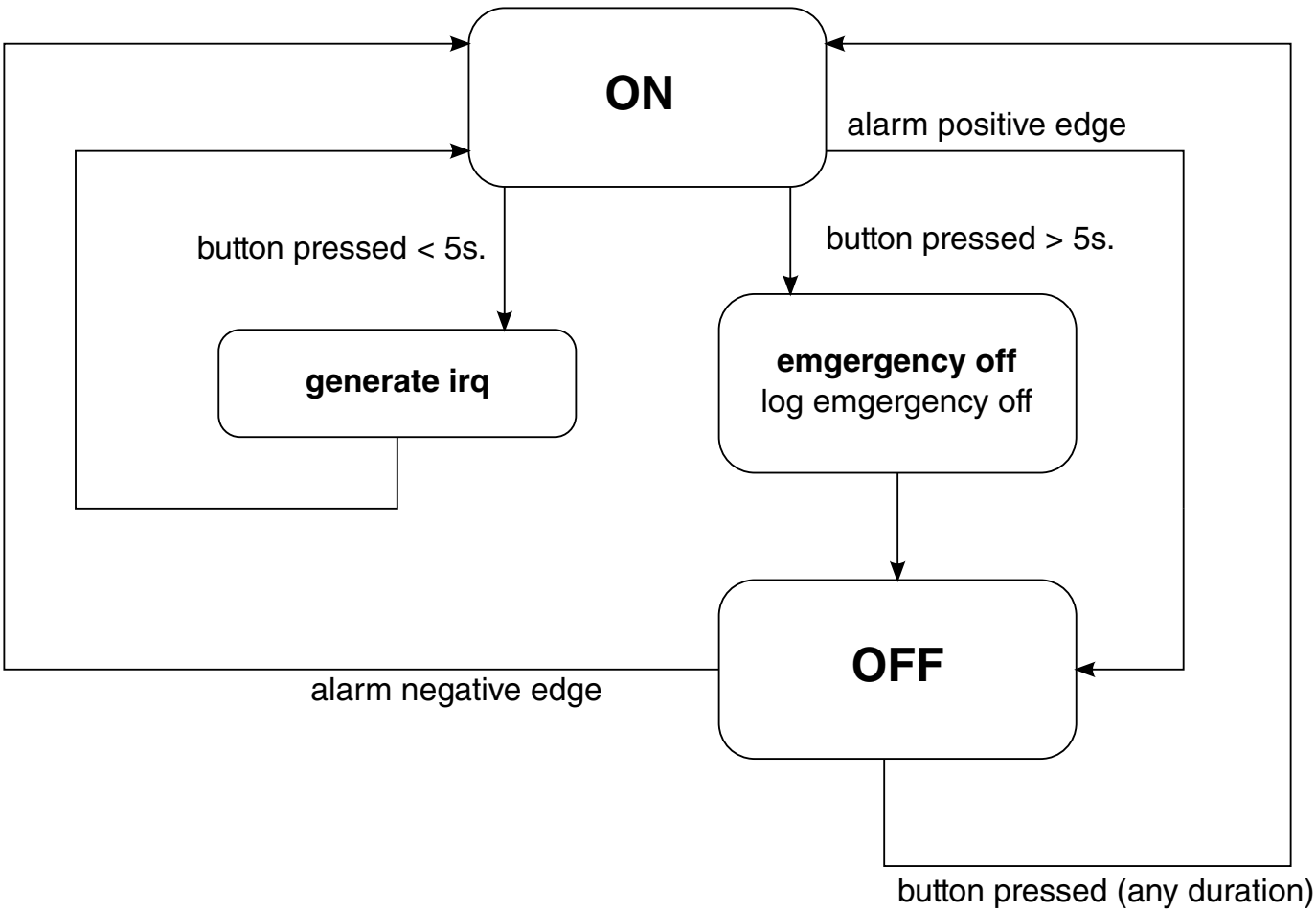


Figure 21-4. FSM

21.5.3 Power mode transitions

Table 21-4. 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 21-4. 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.

21.6 Power-On Reset and power sequencing

This 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.

21.6.1 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.

21.6.2 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).

21.7 Functional Description

21.7.1 Reset Control

This section details the reset control of this device.

21.7.1.1 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:

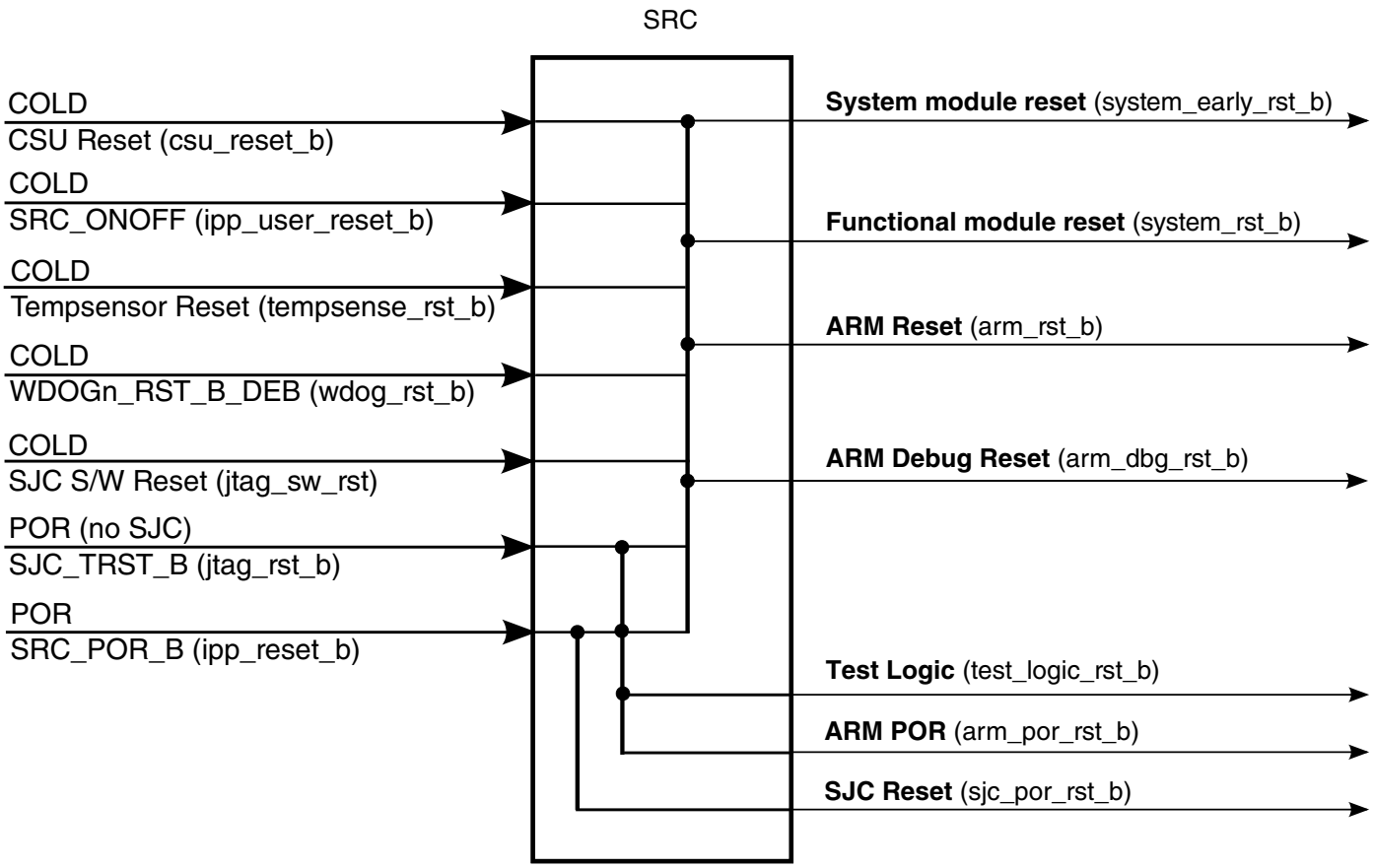


Figure 21-5. SRC inputs and outputs

The reset types and modules they affect are shown in Table 21-5. As there is no chip POR, the POR_B is used to reset the entire chip including test logic and JTAG modules.

NOTE

All resets are expected to be active low except jtag_sw_rst.

Table 21-5. 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.

21.7.1.2 Reset Handling

21.7.1.2.1 POR (SRC_POR_B)

SRC_POR_B is an external reset signal. 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).

21.7.1.2.2 COLD RESET

The sequence is similar to SRC_POR_B except the memory repair operation is not performed.

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).

21.7.2 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.

21.7.3 Boot Mode Control

21.7.3.1 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 after the OCOTP_CTRL asserts the fuse read completion flag. 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 register. The figure below shows the selection of boot mode information.

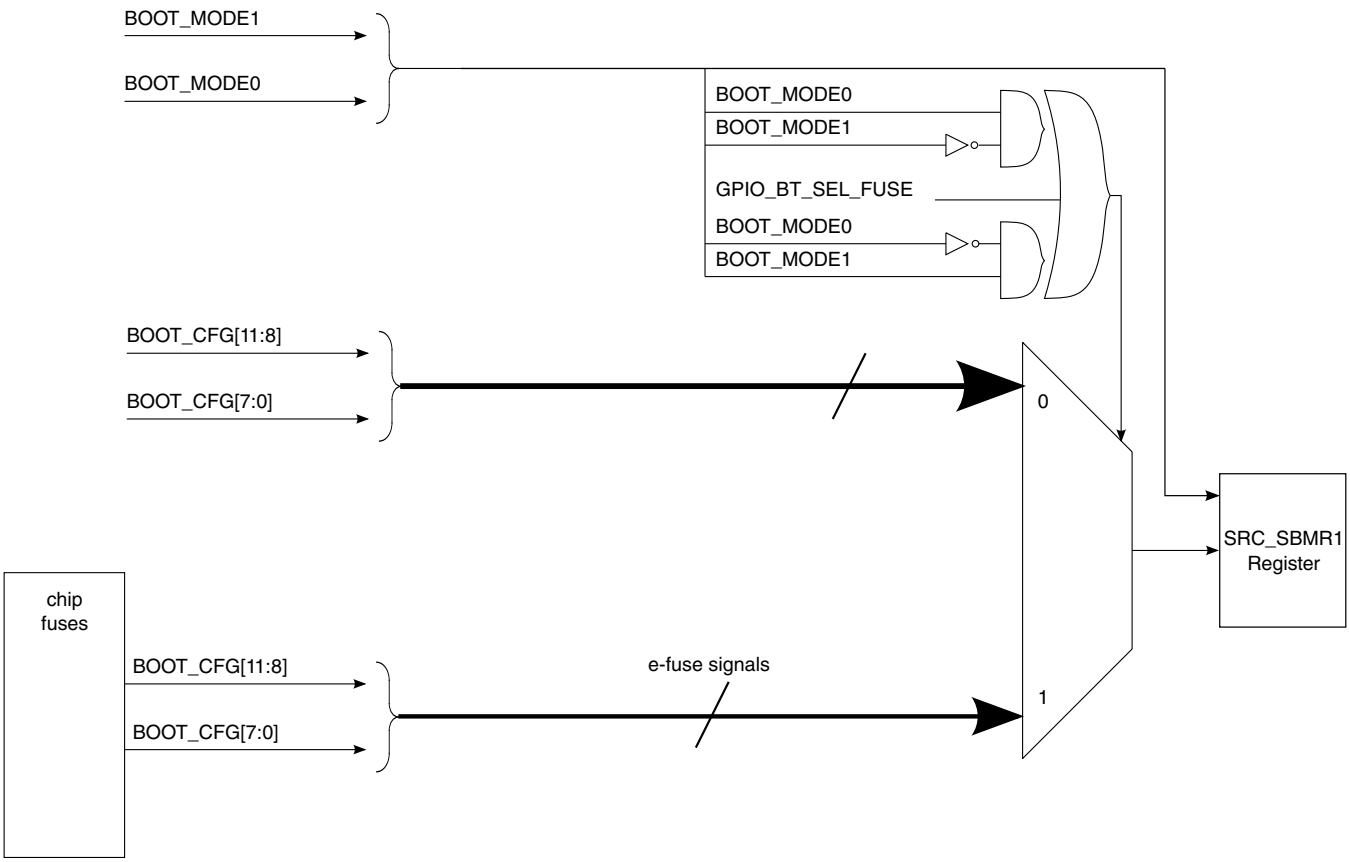


Figure 21-6. Boot mode information

21.8 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	21.8.1/756
400F_8004	SRC Boot Mode Register 1 (SRC_SBMR1)	32	R	0000_0000h	21.8.2/758
400F_8008	SRC Reset Status Register (SRC_SRSR)	32	R/W	0000_0001h	21.8.3/759
400F_801C	SRC Boot Mode Register 2 (SRC_SBMR2)	32	R	0000_0000h	21.8.4/762
400F_8020	SRC General Purpose Register 1 (SRC_GPR1)	32	R/W	0000_0000h	21.8.5/763
400F_8024	SRC General Purpose Register 2 (SRC_GPR2)	32	R/W	0000_0000h	21.8.6/764
400F_8028	SRC General Purpose Register 3 (SRC_GPR3)	32	R/W	0000_0000h	21.8.7/764
400F_802C	SRC General Purpose Register 4 (SRC_GPR4)	32	R/W	0000_0000h	21.8.8/765
400F_8030	SRC General Purpose Register 5 (SRC_GPR5)	32	R/W	0000_0000h	21.8.9/765

Table continues on the next page...

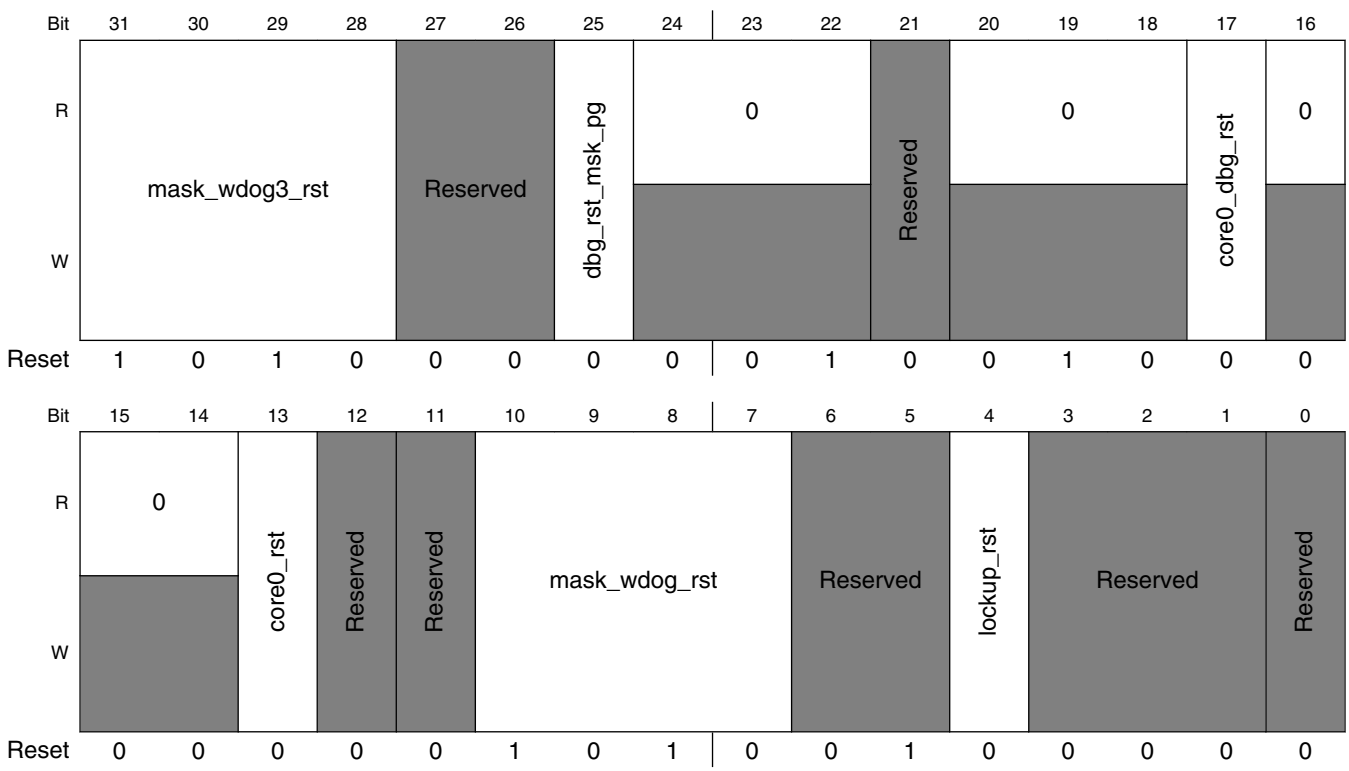
SRC memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400F_8034	SRC General Purpose Register 6 (SRC_GPR6)	32	R/W	0000_0000h	21.8.10/ 766
400F_8038	SRC General Purpose Register 7 (SRC_GPR7)	32	R/W	0000_0000h	21.8.11/ 766
400F_803C	SRC General Purpose Register 8 (SRC_GPR8)	32	R/W	0000_0000h	21.8.12/ 767
400F_8040	SRC General Purpose Register 9 (SRC_GPR9)	32	R/W	0000_0000h	21.8.13/ 767
400F_8044	SRC General Purpose Register 10 (SRC_GPR10)	32	R/W	0000_0000h	21.8.14/ 768

21.8.1 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)
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).

Table continues on the next page...

SRC_SCR field descriptions (continued)

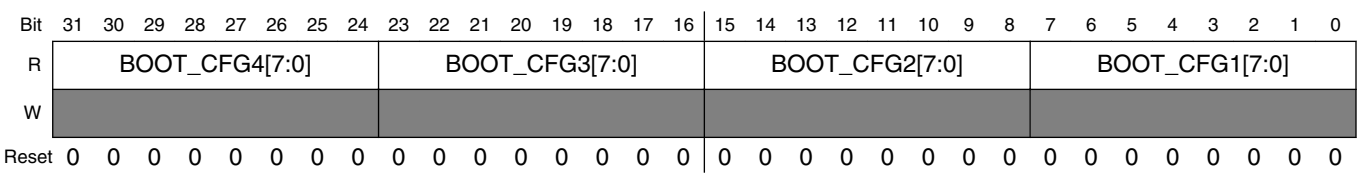
Field	Description
	any other code will be coded to 1010 i.e. wdog_rst_b is not masked 0101 wdog_rst_b is masked 1010 wdog_rst_b is not masked (default)
6–5 -	This field is reserved. Reserved
4 lockup_rst	lockup reset enable bit 0 disabled 1 enabled
3–1 -	This field is reserved. Reserved
0 -	This field is reserved. Reserved

21.8.2 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 registe 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



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.

Table continues on the next page...

SRC_SBMR1 field descriptions (continued)

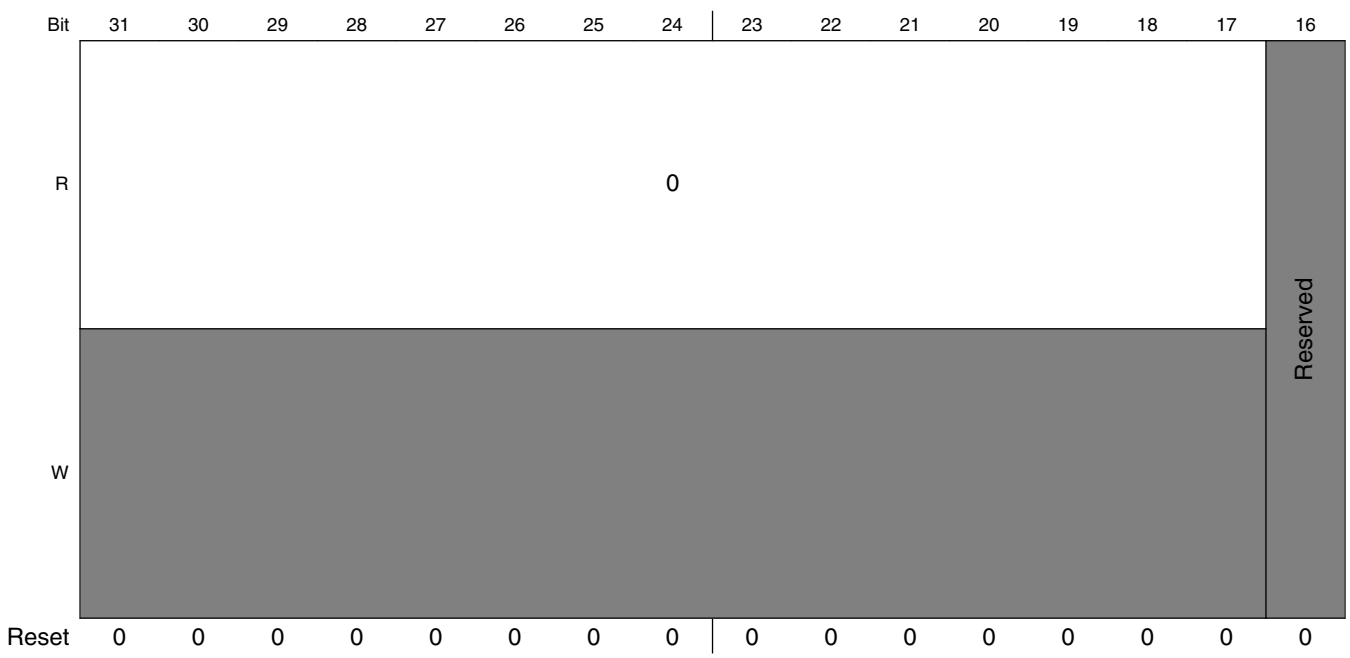
Field	Description
BOOT_ CFG1[7:0]	Refer to fusemap.

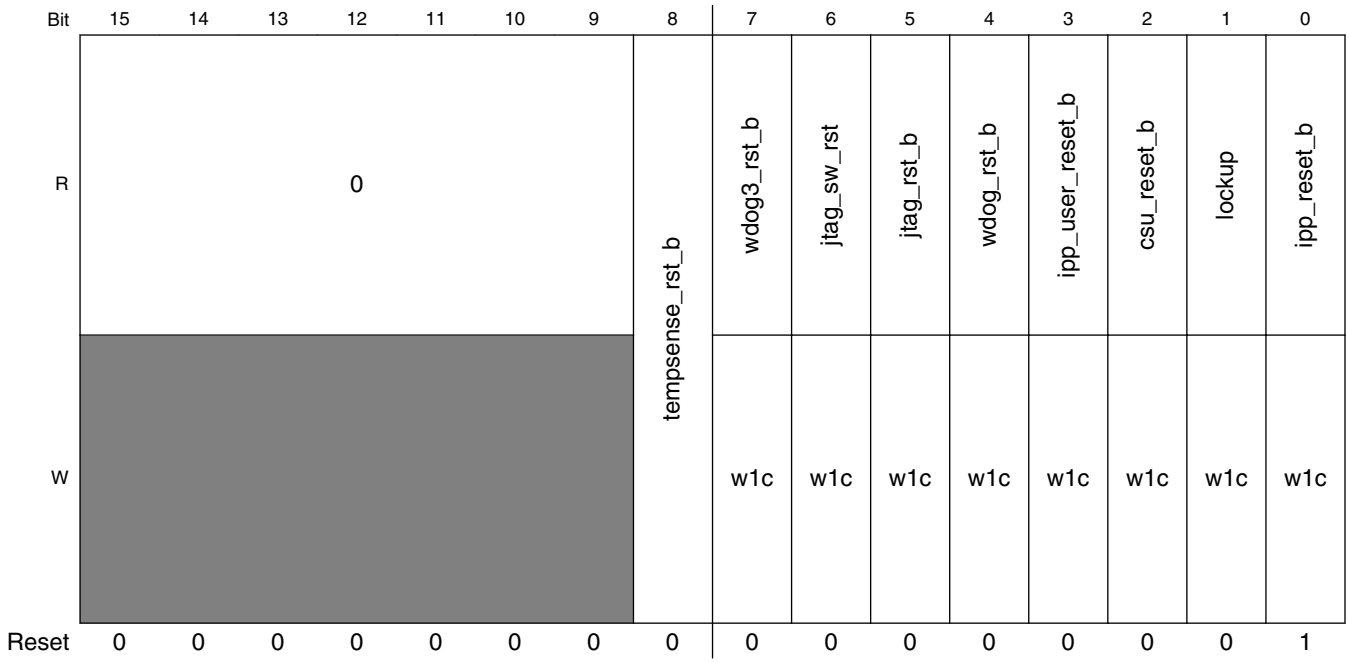
21.8.3 SRC Reset Status Register (SRC_SRSR)

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.
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, or software setting of SYSRESETREQ bit in Application Interrupt and Reset Control Register of the Arm core. SW needs to set some GPR bit before writing SYSRESETREQ bit and use the GPR bit to distinguish if the reset is caused by SYSRESETREQ or from JTAG. 0 Reset is not a result of the mentioned case. 1 Reset is not 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.

Table continues on the next page...

SRC_SRSR field descriptions (continued)

Field	Description
	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	Indicates a reset has been caused by 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.

21.8.4 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	DIR_BT_DIS	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 DIR_BT_DIS	DIR_BT_DIS shows the state of the DIR_BT_DIS fuse. See Chapter 5, Fusemap for additional information on this fuse.
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.

21.8.5 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.

21.8.6 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.

21.8.7 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.

21.8.8 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.

21.8.9 SRC General Purpose Register 5 (SRC_GPR5)

NOTE

This register is used by the ROM code and should not be used by application software.

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.

21.8.10 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.

21.8.11 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

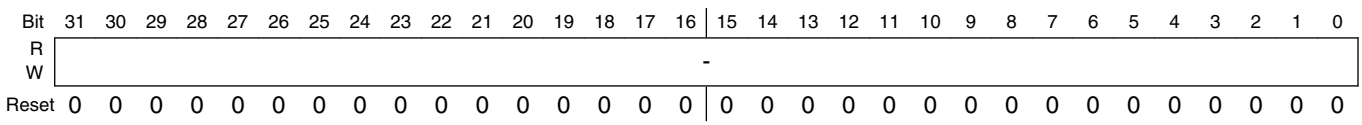
Field	Description
-	Read/write general purpose bits used to store an arbitrary value.

21.8.12 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



SRC_GPR8 field descriptions

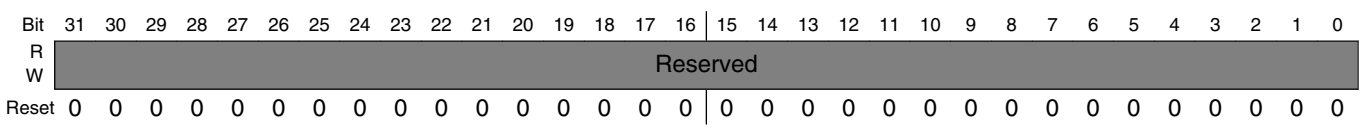
Field	Description
-	Read/write general purpose bits used to store an arbitrary value.

21.8.13 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



SRC_GPR9 field descriptions

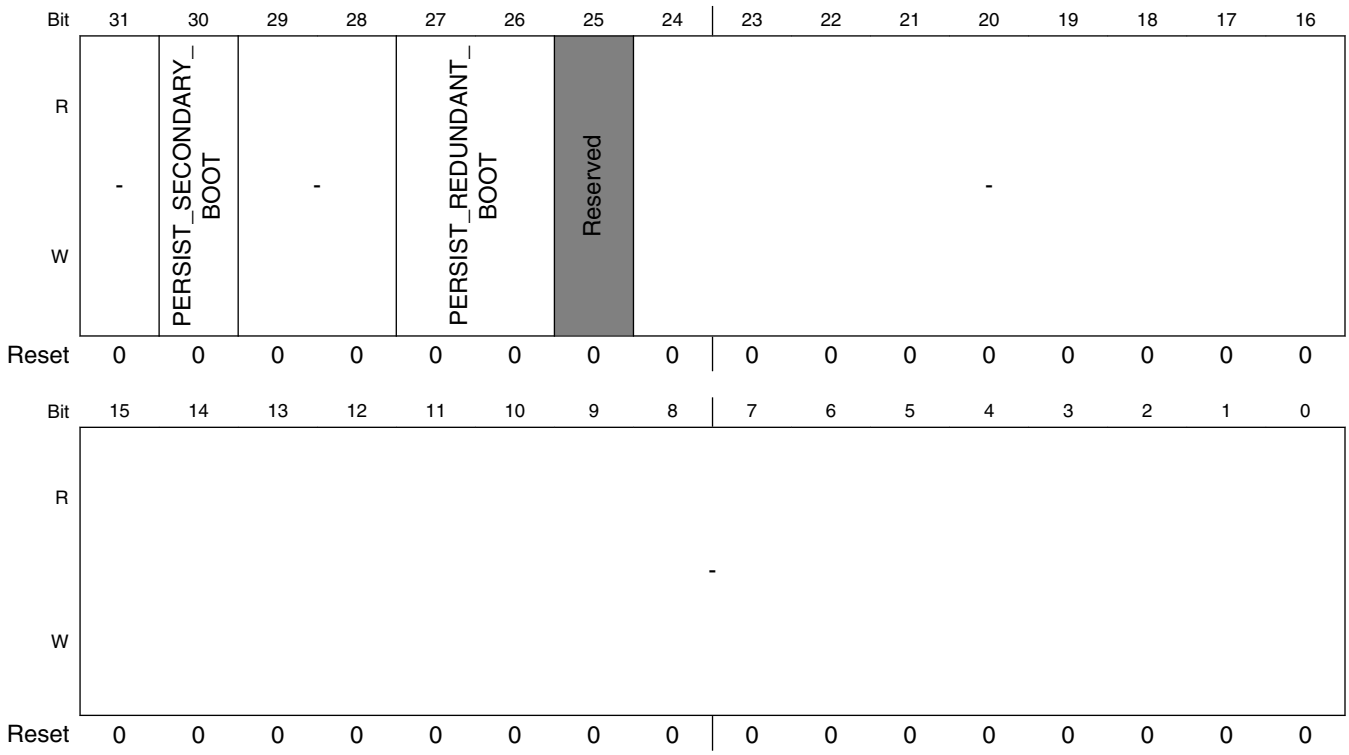
Field	Description
-	This field is reserved. Reserved.

21.8.14 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

NOTE

The fuses marked as “Reserved” are reserved for NXP internal (and future) use only. 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'.

Table 22-2. FlexSPI (Serial NOR) boot fusemap

Address	7	6	5	4	3	2	1	0
0x450[7:0] (BOOT_CFG1)	Reserved	Reserved	Reserved	Reserved	Reserved	FLASH_TYPE: 00-Device supports 3B read by default 01-HyperFlash 3V3 10-MXIC Octal DDR 11 - Micron Octal DDR		EncryptedXIP 0 - Disabled 1 - Enabled
0x450[15:8] (BOOT_CFG2)	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	
0x450[23:16] (BOOT_CFG3)	Reserved							
0x450[31:24]	Reserved							

Table continues on the next page...

Table 22-2. FlexSPI (Serial NOR) boot fusemap (continued)

Address	7	6	5	4	3	2	1	0
(BOOT_CFG4)								
0x460[7:0]	Reserved	UART_INTERRUPT_DISABLE 0 - UART Interrupt Enabled 1 - Uart Interrupt Disabled	FORCE_COLD_BOOT (SBMR)	BT_FUSE_SEL	DIR_BT_DIS	BOOT_FREQ (ARM/ BSU) 0 - 396/132 MHz 1 - 264/132 MHz	SEC_CONFIG[1]	Reserved
0x460[15:8]	Reserved		OTFAD_KEY0_SEL		Reserved (SDR config)			
0x460[23:16]	JTAG_SMODE[1:0]		WDOG_ENABLE '0' - Disabled '1' - Enabled	SJC_DISABLE	DAP_SJC_SWD_SEL	SDP_READ_DISABLE	SDP_DISABLE	FORCE_INTERNAL_BOOT
0x460[31:24]	Reserved		Reserved	Reserved	JTAG_HEO	KTE	Reserved	
0x470[7:0]	Reserved			UART Serial Download Disable: '0' - Not Disable '1' - Disable	Reserved	L1 I-Cache DISABLE	L1 D-Cache DISABLE: 0 - Enable 1 - Disable	Override Pad Settings (using PAD_SETTINGS value)
0x470[15:8]	Reserved					Boot Failure Indicator Pin Select[4]	Reserved	
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]	Reserved	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 22-3. Lock fuses

Address	7	6	5	4	3	2	1	0
0x400[7:0]	Reserved	SJC_RESP_LOCK 0- Unlock '1' - WP,OP,RP	MEM_TRIM_LOCK 00- Unlock '1x' - OP 'x1' - WP		BOOT_CFG_LOCK 00- Unlock '1x' - OP 'x1' - WP		TESTER_LOCK 00 -Unlock '1x' - OP 'x1' - WP	
0x400[15:8]	Reserved	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]	SW_GP2_RLOCK '1' - RP of SW_GP2 fuses	MISC_CONF_LOCK '1' - WP + OP	SW_GP2_LOCK '1' - WP + OP of SW_GP2 fuses	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]	FIELD_RETURN: '1' - Field Returned part '0' - functional part	Reserved	Reserved		GP3_LOCK '1x' - OP 'x1' - WP		Reserved	

NOTE

Do NOT program *Reserved* bits.

22.3 Fusemap Descriptions Table

This section describes the chip fusemap descriptions.

Table 22-4. Fusemap Descriptions

Fuse Address	Fuses Name	Number of Fuses	Fuses Function	Setting	Locked by
0x400[1:0]	TESTER_LOCK	2	Perform lock on Tester programed fuses.	00 - Unlock (The controlled field can be read, sensed, burned or overridden in the corresponded IIM register)	N/A

Table continues on the next page...

Table 22-4. Fusemap Descriptions (continued)

Fuse Address	Fuses Name	Number of Fuses	Fuses Function	Setting	Locked by
				x1 - Write Lock (The controlled field can be read or sensed only) 1x - Operation Lock (The controlled field can't be overridden)	
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]	MEM_TRIM_LOCK	2	Trimming fuses. Burnt on the tester or by customer before the final product shipment.	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[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)	N/A

Table continues on the next page...

Table 22-4. Fusemap Descriptions (continued)

Fuse Address	Fuses Name	Number of Fuses	Fuses Function	Setting	Locked by
				x1 - Write Lock (The controlled field can be read or sensed only) 1x - Operation Lock (The controlled field can't be overridden)	
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]	SW_GP2_LOCK	1	WP and OP Lock for SW_GP2 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[22]	MISC_CONF_LOCK	1			N/A
0x400[23]	SW_GP2_RLOCK	1	RP Lock for SW_GP2 fuse.	0 - Unlock (The controlled field can be read in the	N/A

Table continues on the next page...

Table 22-4. Fusemap Descriptions (continued)

Fuse Address	Fuses Name	Number of Fuses	Fuses Function	Setting	Locked by
				corresponded IIM register) 1 - Lock (The controlled field can't be read in the corresponded IIM register)	
0x400[24]	Reserved	1	Reserved	Reserved	Reserved
0x400[25]	Reserved	1	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 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[29:28]	Reserved	2	Reserved	Reserved	Reserved
0x400[30]	Reserved	1	Reserved	Reserved	Reserved
0x400[31]	FIELD_RETURN	1	Configure device for field return testing. Fuse burning is protected by CSF command, with proper parameter passed. Write / OP protected by FIELD_RETURN_LOCK bit in control register.	0' - Device is in functional / secure mode. '1' - Device is open for 'field-return' testing.	N/A
0x420[10:0], 0x410[31:0]	LOT_NO_ENC[42:0](SJC_CHALL/ UNIQUE_ID[42:0])	43	FSL-wide unique, encoded LOT ID STD II/SJC CHALLENGE/ Unique ID		TESTER_LOCK
0x420[15:11]	WAFER_NO[4:0] (SJC_CHALL[47:43] / UNIQUE_ID[47:43])	5	The wafer number of the wafer on which the device was fabricated/SJC CHALLENGE/ Unique ID		TESTER_LOCK
0x420[23:16]	DIE-Y-CORDINATE[7:0] (SJC_CHALL[55:48] / UNIQUE_ID[55:48])	8	The Y-coordinate of the die location on the wafer/SJC CHALLENGE/ Unique ID		TESTER_LOCK
0x420[31:24]	DIE-X-CORDINATE[7:0] (SJC_CHALL[63:56] / UNIQUE_ID[63:56])	8	The X-coordinate of the die location on the wafer/SJC CHALLENGE/ Unique ID		TESTER_LOCK
0x430[7:0]	Reserved	8	Reserved	Reserved	Reserved
0x430[15:8]	Reserved	8	Reserved	Reserved	Reserved

Table continues on the next page...

Table 22-4. Fusemap Descriptions (continued)

Fuse Address	Fuses Name	Number of Fuses	Fuses Function	Setting	Locked by
0x430[19:16]	SI_REV[3:0]	4	Silicon Revision number.		TESTER_LOCK
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]	PKGINFO[1:0]	2	Burned by tester program, for indicating QSPI boot pin option	00: QFP144 0.5pitch 01: QFP100 0.5pitch 10: Reserved 11: Reserved	TESTER_LOCK

Table continues on the next page...

Table 22-4. 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. **NOTE:** This device only supports "0x0000 - FlexSPI (NOR) boot" option.	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]	SEC_CONFIG[1]	1	Security Configuration (with SEC_CONFIG[0])	00 - FAB (Open) 01 - Open - allows any code to be flashed and executed, even if it has no valid signature. 1x - Closed (Security On)	BOOT_CFG_LOCK
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

Table continues on the next page...

Table 22-4. Fusemap Descriptions (continued)

Fuse Address	Fuses Name	Number of Fuses	Fuses Function	Setting	Locked by
0x460[3]	DIR_BT_DIS	1	Direct External Memory Boot Disable	0 - Direct boot from external memory is allowed 1 - Direct boot from external memory is not allowed	BOOT_CFG_LOCK
0x460[4]	BT_FUSE_SEL	1	Determines, whether using fuses for boot configuration, or GPIO / Serial loader	If boot_mode="10" (internal boot) 0=Boot mode configuration is taken from GPIOs. 1=Boot mode configuration is taken from fuses. If boot_mode="00" (boot from fuses) 0 - Boot using Serila Loader (USB) 1- Boot mode configuration is taken from fuses.	BOOT_CFG_LOCK
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[12]	OTFAD_KEY_SEL0	1	AES key selection for OTFAD	0 - Select [127:0] of key selected by FUSE_OTP_KEY_TO_DCP_DISABLE 1 - Select [255:128] of key selected by FUSE_OTP_KEY_TO_DCP_DISABLE	BOOT_CFG_LOCK
0x460[13]	OTFAD_KEY_SEL1	1	AES key selection for OTFAD	0 - From Key selected by OTFAD_KEY_SEL0	BOOT_CFG_LOCK

Table continues on the next page...

Table 22-4. Fusemap Descriptions (continued)

Fuse Address	Fuses Name	Number of Fuses	Fuses Function	Setting	Locked by
				1 - From FUSE_SW_GPR2[127:0]	
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]	SJC_DISABLE	1	Disable/Enable the Secure JTAG Controller module. This fuse is used to create highest JTAG security level, where JTAG is totally blocked.	0 - Secure JTAG Controller is enabled 1 - Secure JTAG Controller is disabled	BOOT_CFG_LOCK
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]	JTAG_SMODE[1:0]	2	JTAG Security Mode. Controls the security mode of the JTAG debug interface	00 - JTAG enable mode 01 - Secure JTAG mode 11 - No debug mode	BOOT_CFG_LOCK
0x460[25:24]	Reserved	2	Reserved	Reserved	Reserved
0x460[26]	KTE	1	Kill Trace Enable. Enables debug and tracing capability on ETM, and other modules	0 - Debug and tracing are allowed 1 - Debug and tracing are allowed in case security state as defined by Secure JTAG allows it (for example, JTAG_ENABLE)	BOOT_CFG_LOCK
0x460[27]	JTAG_HEO	1	JTAG HAB Enable Override. Disallows HAB JTAG enabling. The HAB may normally enable JTAG debugging by means of the HAB_JDE- bit in the OCOTP SCS	0 - HAB may enable JTAG debug access 1 - HAB JTAG enable is overridden (HAB may not enable JTAG debug access)	BOOT_CFG_LOCK

Table continues on the next page...

Table 22-4. Fusemap Descriptions (continued)

Fuse Address	Fuses Name	Number of Fuses	Fuses Function	Setting	Locked by
			register. The JTAG_HEO-bit can override this behavior		
0x460[28]	Reserved	1			BOOT_CFG_LOCK
0x460[29]	Reserved	1	Reserved	Reserved	Reserved
0x460[31:30]	Reserved	2			BOOT_CFG_LOCK
0x470[0]	Reserved	1			BOOT_CFG_LOCK
0x470[1]	BT_MPU_DISABLE	1	The fuse bit is used for ROM to not enable MMU	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]	Reserved	1			BOOT_CFG_LOCK
0x470[4]	UART Serial Download Disable	1	Disable UART Serial Download	0 - Disable 1 - Enable	BOOT_CFG_LOCK
0x470[6:5]	Reserved	2			BOOT_CFG_LOCK
0x470[7]	Reserved	1	Reserved	Reserved	Reserved
0x470[9:8]	Reserved	2			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[15:11]	Reserved	5			BOOT_CFG_LOCK
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

Table continues on the next page...

Table 22-4. Fusemap Descriptions (continued)

Fuse Address	Fuses Name	Number of Fuses	Fuses Function	Setting	Locked by
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]	Reserved	7			BOOT_CFG_LOCK
0x470[31]	Reserved	1	Reserved	Reserved	Reserved
0x480[5:0]	Reserved	6	Reserved	Reserved	Reserved
0x480[7:6]	Reserved	2	Reserved	Reserved	Reserved
0x480[15:8]	Reserved	8			MEM_TRIM_LOCK
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
0x4F0[31:16]	USB_PID[31:0]	16	USB PID		ANALOG_LOCK
0x500[31:0]	Reserved	256	Reserved	Reserved	Reserved
0x580[31:0]	SRK_HASH[255:0]	256	SRK key, no HW visible lines. NO HW Visible signals available		N/A
0x600[31:0]	SJC_RESP[63:0]	64	Response reference value for the secure JTAG controller		SJC_RESP_LOCK (locks also for read and explicit sense)

Table continues on the next page...

Table 22-4. Fusemap Descriptions (continued)

Fuse Address	Fuses Name	Number of Fuses	Fuses Function	Setting	Locked by
0x620[31:0]	OTFAD_KEY_SCRAMBLE[31:0]	32	Key scramble data value for key blob unwrap		MAC_ADDR_LOCK
0x630[7:0]	OTFAD_KEY_SCRAMBLE_ALIGN[7:0]	8	4x 2-bit key scramble align select for key blob unwrap		MAC_ADDR_LOCK
0x630[8]	OTFAD_ENABLE	1	Reserved enable OTFAD descryption -- Move to GPR		MAC_ADDR_LOCK
0x630[9]	OTFAD_KEY_BLOB_EN	1	Enable key blob unwrap		MAC_ADDR_LOCK
0x630[10]	ENB_OTFAD_KEY_SCRAMBLE	1	Enable KEK scramble during key blob processing		MAC_ADDR_LOCK
0x630[11]	RESTRICT_OTFAD_IPS	1	Restrict IPS access to CR, SR and CRC		MAC_ADDR_LOCK
0x630[12]	Reserved	1	Reserved	Reserved	Reserved
0x630[13]	OTFAD_DISABLE_OVERRIDE	1	Disable OTFAD as if not present		MAC_ADDR_LOCK
0x630[15:14]	Reserved	2	Reserved	Reserved	Reserved
0x630[31:16]	Reserved	16	Reserved	Reserved	Reserved
0x640[31:0]	GP3[31:0]	32	General Purpose fuse register #3		GP3_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]	SW_GP2[127:0]	128	SW general purpose key		SW_GP2_LOCK
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

Table continues on the next page...

Table 22-4. Fusemap Descriptions (continued)

Fuse Address	Fuses Name	Number of Fuses	Fuses Function	Setting	Locked by
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 0011 - Toshiba 19nm RR sequence 0100 - SanDisk 19nm RR sequence 0101 - SanDisk 1ynmRR sequence Others - Reserved	MISC_CONF_LOCK
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_Bank0~3_CFG (O=ORAM,I=ITCM,D=DTCM) 0b0000: 64KB, 32KB, 32KB, {O, O, D, I} 0b0001: 32KB, 32KB, 64KB, {O, D, D, I} 0b0010: 64KB, 0KB, 64KB, {O, O, D, D} 0b0011: 32KB, 0KB, 96KB, {O, D, D, D} 0b0100: 32KB, 64KB, 32KB, {O, D, I, I} 0b0101: 96KB, 0KB, 32KB, {O, O, D, O} 0b0110: 32KB, 96KB,0KB, {O, I, I, I} 0b0111: 64KB, 64KB, 0KB, {O, O, I, I}	MISC_CONF_LOCK

Table continues on the next page...

Table 22-4. Fusemap Descriptions (continued)

Fuse Address	Fuses Name	Number of Fuses	Fuses Function	Setting	Locked by
				0b1000: 96KB, 32KB, 0KB, {O, O, I, O} 0b1001: 128KB, 0KB, 0KB, {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 **NOTE:** This device does NOT support LPSPI boot, so the related bits are reserved.	00 - LPSPI1 01 - LPSPI2	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
0x6D0[31:30]	Reserved	2	Reserved	Reserved	Reserved
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

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

23.2 Overview

This section contains information describing the requirements for the on-chip eFuse OTP controller 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 Features

The OCOTP provides the following features :

- 32-bit word restricted program and read to of 1.5K bit eFuse OTP.
- Loading and housing of fuse content into shadow registers.

- Memory-mapped (restricted) access to shadow registers.
- Generation of HWV_FUSE (hardware visible fuse bus) and the HWV_REG bus which is made of up of volatile PIO register based "fuses". The HWV_REG bits come from the SCS (Software Controllable Signals) register.
- Generation of STICKY_REG which consists of sticky register bits.
- Provides program-protect and read-protect eFuse.
- Provide override and read protection of shadow register.

23.3 Clocks

The table found here describes the clock sources for OCOTP. Please see the chip-specific clocking section (CCM chapter) for clock setting, configuration and gating information.

Table 23-2. OCOTP Clocks

Clock name	Clock Root	Description
ipg_clk	ipg_clk_root	Peripheral clock
ipg_clk_s	ipg_clk_root	Peripheral access clock

23.4 Top-Level Symbol and Functional Overview

The figure below shows the OCOTP system level diagram.

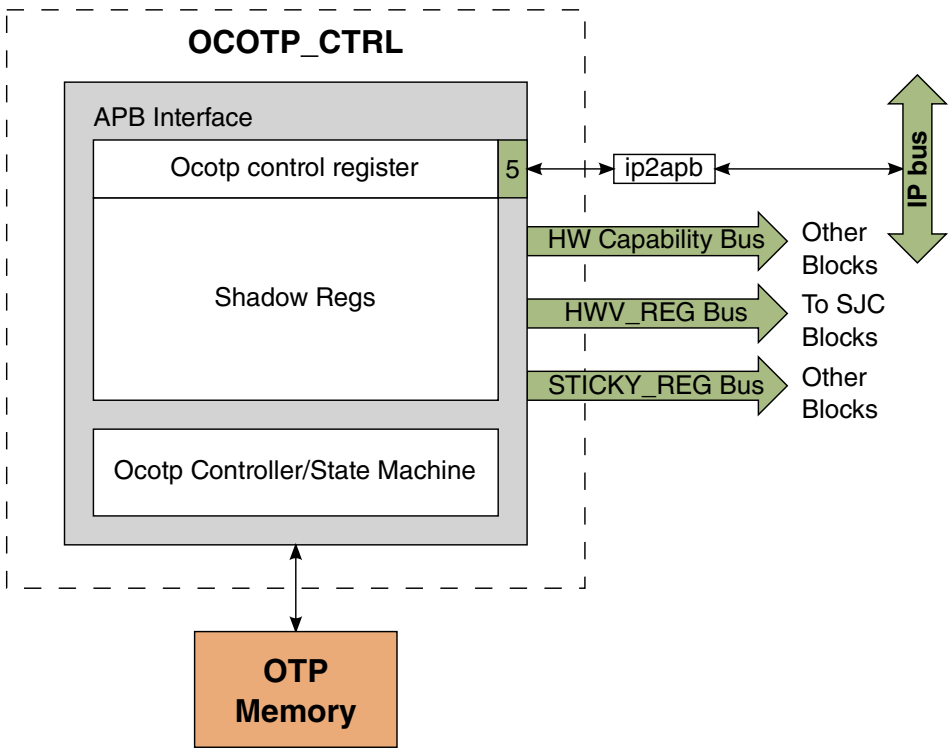


Figure 23-1. OCOTP System Level Diagram

23.4.1 Operation

The IP bus interface of the OCOTP has the following functions.

- Configure control registers for programming and reading fuse wordbank.
- Override and read shadow registers.

OCOTP configuration for program and read are performed on 32-bit words for 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.

23.4.1.1 Shadow Register Reload

All fuse words in efusebox are shadowed. Therefore, fuse information is available through 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 (including HW_OCOTP_LOCK) without having to reset the device. To force a reload, complete the following steps:

1. Set the HW_OCOTP_TIMING[STROBE_READ] field value appropriately.
2. Set the HW_OCOTP_TIMING[RELAX] field value appropriately.
3. Check that HW_OCOTP_CTRL[BUSY] and HW_OCOTP_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 HW_OCOTP_CTRL[RELOAD_SHADOWS] bit. OCOTP will read all the fuses one by one and put it into corresponding shadow register.
5. Wait for HW_OCOTP_CTRL[BUSY] and HW_OCOTP_CTRL[RELOAD_SHADOWS] to be cleared by the controller.

The controller will automatically clear the HW_OCOTP_CTRL[RELOAD_SHADOWS] bit after the successful completion of the operation.

23.4.1.2 Fuse and Shadow register read

All shadow registers are always readable through the APB bus except some secret keys regions. 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 HW_OCOTP_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 locations will still work successfully however.

To read fuse words directly from fusebox correctly complete the following steps:

1. Program HW_OCOTP_TIMING[STROBE_READ] and HW_OCOTP_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 HW_OCOTP_TIMING parameters are set correctly.
2. Check that HW_OCOTP_CTRL[BUSY] and HW_OCOTP_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 address to HW_OCOTP_CTRL[ADDR].

4. Set HW_OCOTP_READ_CTRL[READ_FUSE] to 1. OCOTP will auto read the fuse word according to HW_OCOTP_CTRL[ADDR] in fusebox. Then put read value into HW_OCOTP_READ_FUSE_DATA
5. Once complete, the controller will clear BUSY. A read request to a protected or locked region will result in no OTP access and no setting of HW_OCOTP_CTRL[BUSY]. In addition HW_OCOTP_CTRL[ERROR] will be set. It must be cleared by software before any new access can be issued.
6. Read HW_OCOTP_READ_FUSE_DATA to get fuse word value. HW_OCOTP_READ_FUSE_DATA will be 0xBADABADA when HW_OCOTP_CTRL[ERROR] is set.

23.4.1.3 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 LOCK shadow register also has no shadow 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 correctly complete the following steps:

1. Program the following fields with timing values to match the frequency of ipg_clk:
   - HW_OCOTP_TIMING[STROBE_PROG]
   - HW_OCOTP_TIMING[RELAX]

   OTP writes will work at maximum bus frequencies as long as the HW_OCOTP_TIMING parameters are set correctly.
2. Check that HW_OCOTP_CTRL[BUSY] and HW_OCOTP_CTRL[ERROR] are clear. Overlapped accesses are not supported by the controller. Any pending write or reload must be completed before a write access can be requested.
3. Write the requested address to HW_OCOTP_CTRL[ADDR] and program the unlock code into HW_OCOTP_CTRL[WR_UNLOCK]. This must be programmed for each write access. The lock 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 HW_OCOTP_DATA register. This will automatically set HW_OCOTP_CTRL[BUSY] and clear HW_OCOTP_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 HW_OCOTP_CTRL[ADDR] which cannot be updated until the next write sequence is initiated. This copy guarantees that erroneous writes to

HW_OCOTP_CTRL[ADDR] will not affect an active write operation. During the write operation, HW_OCOTP_DATA cannot be modified.

5. Once complete, the controller will clear BUSY. A write request to a protected or locked region will result in no OTP access and no setting of HW_OCOTP_CTRL[BUSY]. In addition HW_OCOTP_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. Write latencies is based on amount of bit filed which is 1. For example : program half fuse bits in one word need 10 us x 16.

For further details of OTP read/write operations see [eFUSE].

HW_OCOTP_CTRL[ERROR] will be set under the following conditions:

- A write is performed to a shadow register during a shadow reload (essentially, while HW_OCOTP_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 from 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.

23.4.1.4 Write Postamble

Due to internal electrical characteristics of the OTP during writes, all OTP operations following a write must be separated by 2 us after the clearing of HW_OCOTP_CTRL_BUSY following the write. This guarantees programming voltages on-chip to reach a steady state when exiting a write sequence. This includes reads, shadow reloads, or other writes.

A recommended software sequence to meet the postamble requirements is as follows:

• Issue the write and poll for BUSY (as per Fuse Shadow Memory Footprint).

• Once BUSY is clear, use HW_DIGCTL_MICROSECONDS to wait 2 us.

• Perform the next OTP operation.

23.4.2 Fuse Shadow Memory Footprint

The OTP memory footprint shows in the following figure. The registers are grouped by lock region. Their names correspond to the PIO register and fusemap names.

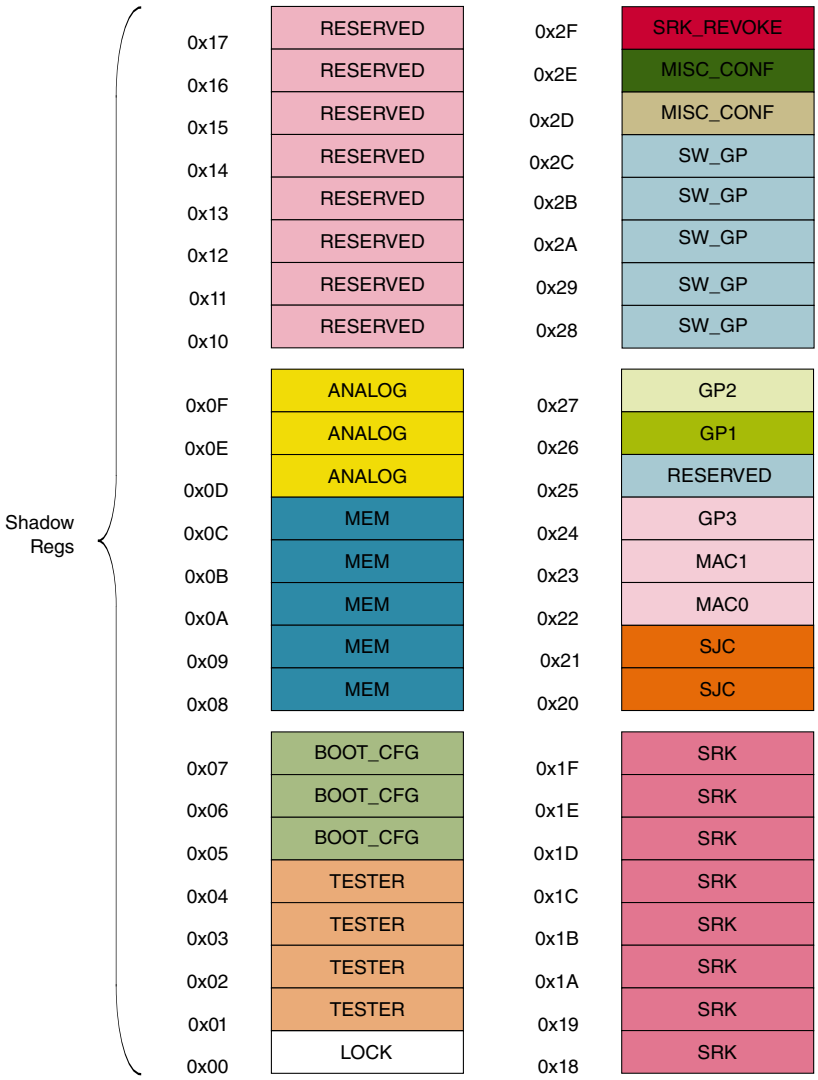


Figure 23-2. OTP Memory Footprint

23.4.3 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
HW_OCOTP_TIMING.WAIT	"WAIT" specifies time interval between auto read and write access in one time program.

Table continues on the next page...

	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.
HW_OCOTP_TIMING2.RELAX1	"RELAX1" is used to configure the setup/hold time for certain timing margin. $t_{SP\_PG\_AVDD}$ = $t_{HP\_PG\_AVDD}$ = (RELAX1 + 1)/ipg_clk_freq, should be ≧ 1000 ns.

23.4.4 Hardware Visible Fuses

The hwv_fuse bus emanates from the OCOTP block and goes to various other blocks inside the chip. This bus is made up of the shadow register bits .

Only a subset of these fuse bits are currently used by the hardware. The fuse bits are initially copied from the banks after reset is deasserted. When all fuse bits are loaded into their shadow registers, the OCOTP asserts the fuse_latched output signal.

The hwv_reg bus also comes from the OCOTP. Its source is the HW_OCOTP_SCS register. This register has 1 defined bit, the HAB_JDE bit, that is connected to the SJC block. The SCS bits are intended to be used as volatile fuse bits under software control. Additional bits will be defined as needed in future implementations.

The system-wide reset sequence must be coordinated by the system reset controller, so that the hwv_fuse and hwv_reg buses are stable and reflect the values of the fuses before they are used by the rest of the system.

23.4.5 Behavior During Reset

The OCOTP is always active. The shadow registers automatically load the appropriate OTP contents after reset is deasserted. During this load-time HW_OCOTP_CTRL_BUSY is set. The load time is similar to that of a "reload shadow" operation.

23.4.6 Secure JTAG control

The JTAG control fuses are used to allow or disallow JTAG access to secured resources.

Three JTAG security levels are envisioned, as shown in the table below.

Table 23-3. JTAG Security Level Control Bits

Security Mode	JTAG_SMODE	Description
No Debug	2'b11	The highest security level.
Secure JTAG	2'b01	Limit the JTAG access by using key based authentication mechanism.
JTAG Enable	2'b00	Low Security, all JTAG features are enabled.

23.5 Fuse Map

See the Fusemap chapter of this reference manual for more information.

23.6 OCOTP Memory Map/Register Definition

The OCOTP Memory Map/Register Definition can be found here.

23.6.1 OCOTP register descriptions

23.6.1.1 OCOTP Memory map

OCOTP base address: 401F_4000h

Offset	Register	Width (In bits)	Access	Reset value
0h	OTP Controller Control Register (HW_OCOTP_CTRL)	32	RW	0000_0000h
4h	OTP Controller Control Register (HW_OCOTP_CTRL_SET)	32	RW	0000_0000h
8h	OTP Controller Control Register (HW_OCOTP_CTRL_CLR)	32	RW	0000_0000h
Ch	OTP Controller Control Register (HW_OCOTP_CTRL_TOG)	32	RW	0000_0000h
10h	OTP Controller Timing Register (HW_OCOTP_TIMING)	32	RW	060D_9755h
20h	OTP Controller Write Data Register (HW_OCOTP_DATA)	32	RW	0000_0000h
30h	OTP Controller Write Data Register (HW_OCOTP_READ_CTRL)	32	RW	0000_0000h
40h	OTP Controller Read Data Register (HW_OCOTP_READ_FUSE_DATA)	32	RW	0000_0000h
50h	Sticky bit Register (HW_OCOTP_SW_STICKY)	32	RW	0000_0000h
60h	Software Controllable Signals Register (HW_OCOTP_SCS)	32	RW	0000_0000h
64h	Software Controllable Signals Register (HW_OCOTP_SCS_SET)	32	RW	0000_0000h
68h	Software Controllable Signals Register (HW_OCOTP_SCS_CLR)	32	RW	0000_0000h
6Ch	Software Controllable Signals Register (HW_OCOTP_SCS_TOG)	32	RW	0000_0000h
90h	OTP Controller Version Register (HW_OCOTP_VERSION)	32	RO	0600_0000h
100h	OTP Controller Timing Register 2 (HW_OCOTP_TIMING2)	32	RW	01C3_0092h
400h	Value of OTP Bank0 Word0 (Lock controls) (HW_OCOTP_LOCK)	32	RW	0000_0000h
410h	Value of OTP Bank0 Word1 (Configuration and Manufacturing Info.) (HW_OCOTP_CFG0)	32	RW	0000_0000h
420h	Value of OTP Bank0 Word2 (Configuration and Manufacturing Info.) (HW_OCOTP_CFG1)	32	RW	0000_0000h
430h	Value of OTP Bank0 Word3 (Configuration and Manufacturing Info.) (HW_OCOTP_CFG2)	32	RW	0000_0000h
440h	Value of OTP Bank0 Word4 (Configuration and Manufacturing Info.) (HW_OCOTP_CFG3)	32	RW	0000_0000h
450h	Value of OTP Bank0 Word5 (Configuration and Manufacturing Info.) (HW_OCOTP_CFG4)	32	RW	0000_0000h
460h	Value of OTP Bank0 Word6 (Configuration and Manufacturing Info.) (HW_OCOTP_CFG5)	32	RW	0000_0000h
470h	Value of OTP Bank0 Word7 (Configuration and Manufacturing Info.) (HW_OCOTP_CFG6)	32	RW	0000_0000h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
480h	Value of OTP Bank1 Word0 (Memory Related Info.) (HW_OCOTP_MEM0)	32	RW	0000_0000h
490h	Value of OTP Bank1 Word1 (Memory Related Info.) (HW_OCOTP_MEM1)	32	RW	0000_0000h
4A0h	Value of OTP Bank1 Word2 (Memory Related Info.) (HW_OCOTP_MEM2)	32	RW	0000_0000h
4B0h	Value of OTP Bank1 Word3 (Memory Related Info.) (HW_OCOTP_MEM3)	32	RW	0000_0000h
4C0h	Value of OTP Bank1 Word4 (Memory Related Info.) (HW_OCOTP_MEM4)	32	RW	0000_0000h
4D0h	Value of OTP Bank1 Word5 (Analog Info.) (HW_OCOTP_ANA0)	32	RW	0000_0000h
4E0h	Value of OTP Bank1 Word6 (Analog Info.) (HW_OCOTP_ANA1)	32	RW	0000_0000h
4F0h	Value of OTP Bank1 Word7 (Analog Info.) (HW_OCOTP_ANA2)	32	RW	0000_0000h
580h	Shadow Register for OTP Bank3 Word0 (SRK Hash) (HW_OCOTP_SRK0)	32	RW	0000_0000h
590h	Shadow Register for OTP Bank3 Word1 (SRK Hash) (HW_OCOTP_SRK1)	32	RW	0000_0000h
5A0h	Shadow Register for OTP Bank3 Word2 (SRK Hash) (HW_OCOTP_SRK2)	32	RW	0000_0000h
5B0h	Shadow Register for OTP Bank3 Word3 (SRK Hash) (HW_OCOTP_SRK3)	32	RW	0000_0000h
5C0h	Shadow Register for OTP Bank3 Word4 (SRK Hash) (HW_OCOTP_SRK4)	32	RW	0000_0000h
5D0h	Shadow Register for OTP Bank3 Word5 (SRK Hash) (HW_OCOTP_SRK5)	32	RW	0000_0000h
5E0h	Shadow Register for OTP Bank3 Word6 (SRK Hash) (HW_OCOTP_SRK6)	32	RW	0000_0000h
5F0h	Shadow Register for OTP Bank3 Word7 (SRK Hash) (HW_OCOTP_SRK7)	32	RW	0000_0000h
600h	Value of OTP Bank4 Word0 (Secure JTAG Response Field) (HW_OCOTP_SJC_RESP0)	32	RW	0000_0000h
610h	Value of OTP Bank4 Word1 (Secure JTAG Response Field) (HW_OCOTP_SJC_RESP1)	32	RW	0000_0000h
620h	Value of OTP Bank4 Word2 (MAC Address) (HW_OCOTP_MAC0)	32	RW	0000_0000h
630h	Value of OTP Bank4 Word3 (MAC Address) (HW_OCOTP_MAC1)	32	RW	0000_0000h
640h	Value of OTP Bank4 Word4 (MAC Address) (HW_OCOTP_GP3)	32	RW	0000_0000h
660h	Value of OTP Bank4 Word6 (General Purpose Customer Defined Info) (HW_OCOTP_GP1)	32	RW	0000_0000h
670h	Value of OTP Bank4 Word7 (General Purpose Customer Defined Info) (HW_OCOTP_GP2)	32	RW	0000_0000h
680h	Value of OTP Bank5 Word0 (SW GP1) (HW_OCOTP_SW_GP1)	32	RW	0000_0000h
690h	Value of OTP Bank5 Word1 (SW GP2) (HW_OCOTP_SW_GP20)	32	RW	0000_0000h
6A0h	Value of OTP Bank5 Word2 (SW GP2) (HW_OCOTP_SW_GP21)	32	RW	0000_0000h
6B0h	Value of OTP Bank5 Word3 (SW GP2) (HW_OCOTP_SW_GP22)	32	RW	0000_0000h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
6C0h	Value of OTP Bank5 Word4 (SW GP2) (HW_OCOTP_SW_GP23)	32	RW	0000_0000h
6D0h	Value of OTP Bank5 Word5 (Misc Conf) (HW_OCOTP_MISC_CONF0)	32	RW	0000_0000h
6E0h	Value of OTP Bank5 Word6 (Misc Conf) (HW_OCOTP_MISC_CONF1)	32	RW	0000_0000h
6F0h	Value of OTP Bank5 Word7 (SRK Revoke) (HW_OCOTP_SRK_REVOKE)	32	RW	0000_0000h

23.6.1.2 OTP Controller Control Register (HW_OCOTP_CTRL)

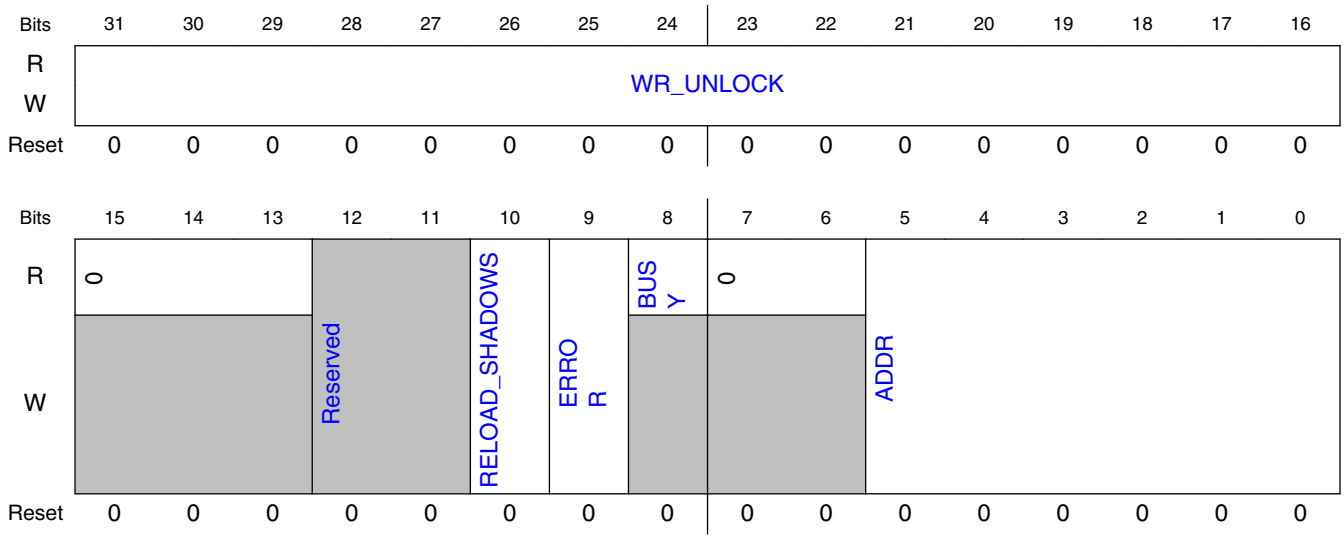
23.6.1.2.1 Offset

Register	Offset
HW_OCOTP_CTRL	0h

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 and BUSY/ERROR may be used in conjuction with the HW_OCOTP_DATA register to perform write operations. Read operations to the On-Chip OTP are involving ADDR, BUSY/ERROR bit field and HW_OCOTP_READ_CTRL register. Read value is saved in HW_OCOTP_READ_FUSE_DATA register.

23.6.1.2.3 Diagram



23.6.1.2.4 Fields

Field	Function
31-16 WR_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 HW_OCOTP_DATA is written), so the UNLOCK bitfield must contain the correct key value during all writes to HW_OCOTP_DATA, otherwise a write shall not be initiated. This field is automatically cleared after a successful write completion (clearing of BUSY).
15-13 —	Reserved
12-11 —	Reserved
10 RELOAD_SHADOWS	Set to force re-loading the shadow registers (HW/SW capability and LOCK). This operation will automatically set BUSY. Once the shadow registers have been re-loaded, BUSY and RELOAD_SHADOWS are automatically cleared by the controller.
9 ERROR	Set by the controller when an access to a locked region(OTP or shadow register) is requested. Must be cleared before any further access can be performed. This bit can only be set by the controller. This bit is also set if the Pin interface is active and software requests an access to the OTP. In this instance, the ERROR bit cannot be cleared until the Pin interface access has completed. Reset this bit by writing a one to the SCT clear address space and not by a general write.
8 BUSY	OTP controller status bit. When active, no new write access or read access to OTP(including RELOAD_SHADOWS) can be performed. Cleared by controller when access complete. 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.
7-6 —	Reserved
5-0 ADDR	OTP write and read access address register. Specifies one of 128 word address locations (0x00 - 0x7f). If a valid access is accepted by the controller, the controller makes an internal copy of this value. This internal copy will not update until the access is complete.

23.6.1.3 OTP Controller Control Register (HW_OCOTP_CTRL_SET)

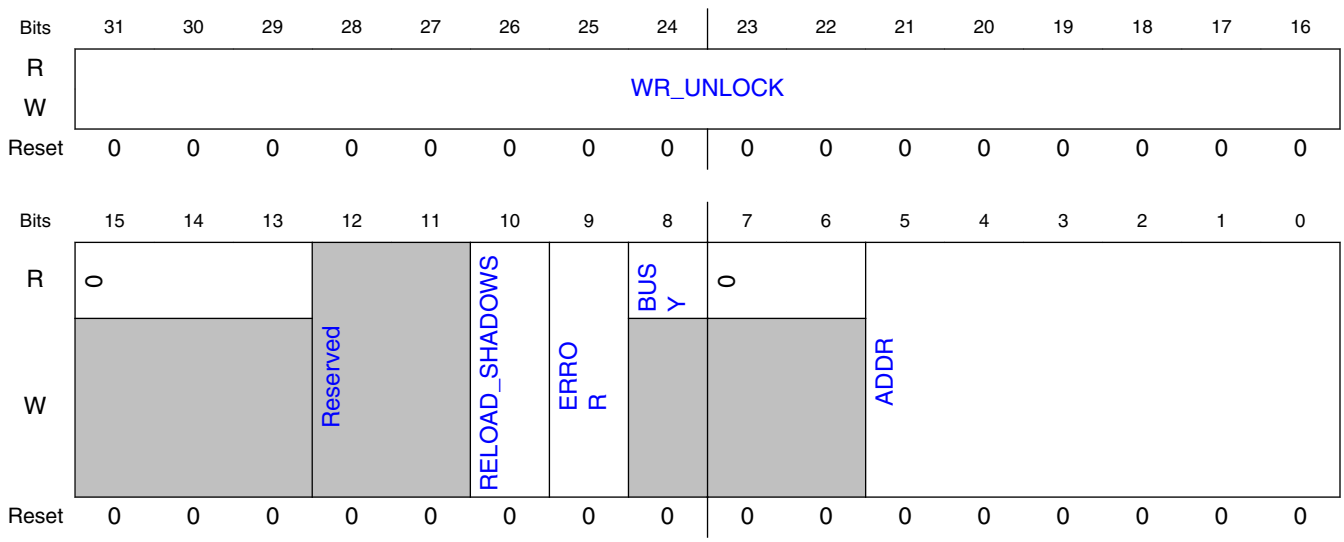
23.6.1.3.1 Offset

Register	Offset
HW_OCOTP_CTRL_SET	4h

23.6.1.3.2 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 and BUSY/ERROR may be used in conjuction with the HW_OCOTP_DATA register to perform write operations. Read operations to the On-Chip OTP are involving ADDR, BUSY/ERROR bit field and HW_OCOTP_READ_CTRL register. Read value is saved in HW_OCOTP_READ_FUSE_DATA register.

23.6.1.3.3 Diagram



23.6.1.3.4 Fields

Field	Function
31-16 WR_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 HW_OCOTP_DATA is written), so the UNLOCK bitfield must contain the

Table continues on the next page...

Field	Function
	correct key value during all writes to HW_OCOTP_DATA, otherwise a write shall not be initiated. This field is automatically cleared after a successful write completion (clearing of BUSY).
15-13 —	Reserved
12-11 —	Reserved
10 RELOAD_SHADOWS	Set to force re-loading the shadow registers (HW/SW capability and LOCK). This operation will automatically set BUSY. Once the shadow registers have been re-loaded, BUSY and RELOAD_SHADOWS are automatically cleared by the controller.
9 ERROR	Set by the controller when an access to a locked region(OTP or shadow register) is requested. Must be cleared before any further access can be performed. This bit can only be set by the controller. This bit is also set if the Pin interface is active and software requests an access to the OTP. In this instance, the ERROR bit cannot be cleared until the Pin interface access has completed. Reset this bit by writing a one to the SCT clear address space and not by a general write.
8 BUSY	OTP controller status bit. When active, no new write access or read access to OTP(including RELOAD_SHADOWS) can be performed. Cleared by controller when access complete. 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.
7-6 —	Reserved
5-0 ADDR	OTP write and read access address register. Specifies one of 128 word address locations (0x00 - 0x7f). If a valid access is accepted by the controller, the controller makes an internal copy of this value. This internal copy will not update until the access is complete.

23.6.1.4 OTP Controller Control Register (HW_OCOTP_CTRL_CLR)

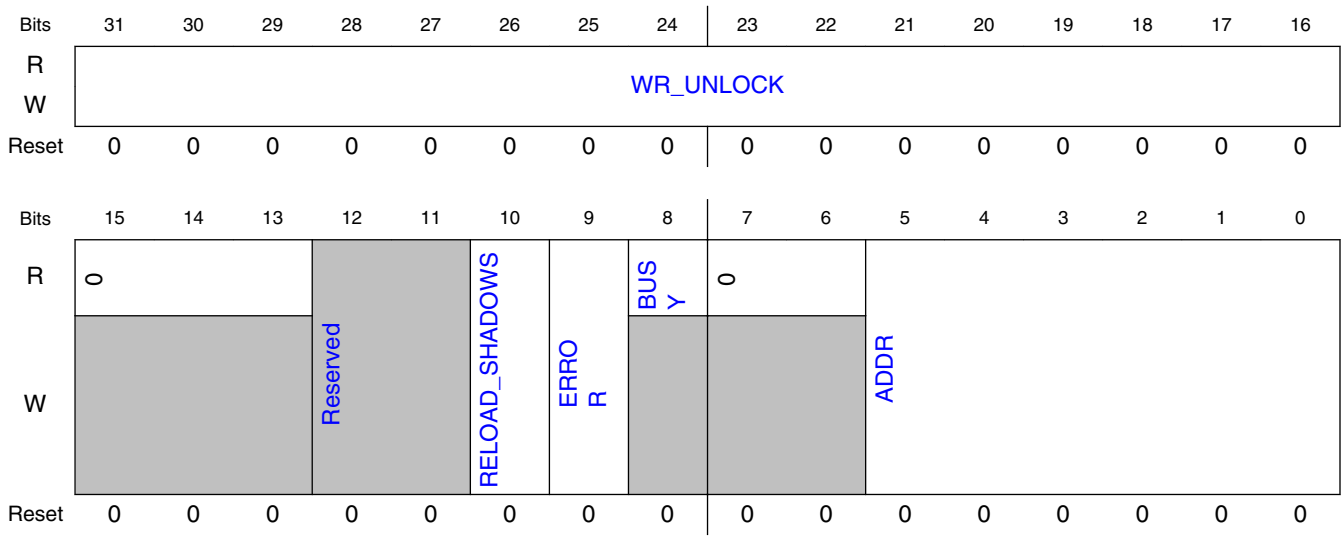
23.6.1.4.1 Offset

Register	Offset
HW_OCOTP_CTRL_CLR	8h

23.6.1.4.2 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 and BUSY/ERROR may be used in conjuction with the HW_OCOTP_DATA register to perform write operations. Read operations to the On-Chip OTP are involving ADDR, BUSY/ERROR bit field and HW_OCOTP_READ_CTRL register. Read value is saved in HW_OCOTP_READ_FUSE_DATA register.

23.6.1.4.3 Diagram



23.6.1.4.4 Fields

Field	Function
31-16 WR_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 HW_OCOTP_DATA is written), so the UNLOCK bitfield must contain the correct key value during all writes to HW_OCOTP_DATA, otherwise a write shall not be initiated. This field is automatically cleared after a successful write completion (clearing of BUSY).
15-13 —	Reserved
12-11 —	Reserved
10 RELOAD_SHADOWS	Set to force re-loading the shadow registers (HW/SW capability and LOCK). This operation will automatically set BUSY. Once the shadow registers have been re-loaded, BUSY and RELOAD_SHADOWS are automatically cleared by the controller.
9 ERROR	Set by the controller when an access to a locked region(OTP or shadow register) is requested. Must be cleared before any further access can be performed. This bit can only be set by the controller. This bit is also set if the Pin interface is active and software requests an access to the OTP. In this instance, the ERROR bit cannot be cleared until the Pin interface access has completed. Reset this bit by writing a one to the SCT clear address space and not by a general write.
8 BUSY	OTP controller status bit. When active, no new write access or read access to OTP(including RELOAD_SHADOWS) can be performed. Cleared by controller when access complete. 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.
7-6 —	Reserved
5-0 ADDR	OTP write and read access address register. Specifies one of 128 word address locations (0x00 - 0x7f). If a valid access is accepted by the controller, the controller makes an internal copy of this value. This internal copy will not update until the access is complete.

23.6.1.5 OTP Controller Control Register (HW_OCOTP_CTRL_TOG)

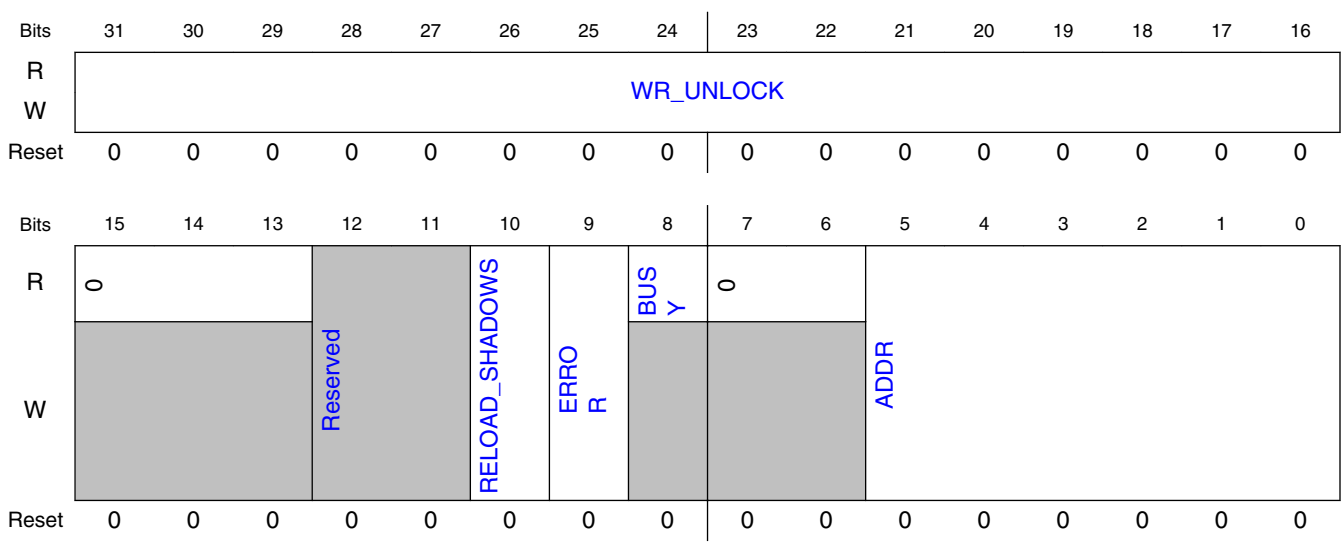
23.6.1.5.1 Offset

Register	Offset
HW_OCOTP_CTRL_TOG	Ch

23.6.1.5.2 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 and BUSY/ERROR may be used in conjuction with the HW_OCOTP_DATA register to perform write operations. Read operations to the On-Chip OTP are involving ADDR, BUSY/ERROR bit field and HW_OCOTP_READ_CTRL register. Read value is saved in HW_OCOTP_READ_FUSE_DATA register.

23.6.1.5.3 Diagram



23.6.1.5.4 Fields

Field	Function
31-16 WR_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 HW_OCOTP_DATA is written), so the UNLOCK bitfield must contain the correct key value during all writes to HW_OCOTP_DATA, otherwise a write shall not be initiated. This field is automatically cleared after a successful write completion (clearing of BUSY).
15-13 —	Reserved
12-11 —	Reserved
10 RELOAD_SHADOWS	Set to force re-loading the shadow registers (HW/SW capability and LOCK). This operation will automatically set BUSY. Once the shadow registers have been re-loaded, BUSY and RELOAD_SHADOWS are automatically cleared by the controller.
9 ERROR	Set by the controller when an access to a locked region(OTP or shadow register) is requested. Must be cleared before any further access can be performed. This bit can only be set by the controller. This bit is also set if the Pin interface is active and software requests an access to the OTP. In this instance, the ERROR bit cannot be cleared until the Pin interface access has completed. Reset this bit by writing a one to the SCT clear address space and not by a general write.
8 BUSY	OTP controller status bit. When active, no new write access or read access to OTP(including RELOAD_SHADOWS) can be performed. Cleared by controller when access complete. 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.
7-6 —	Reserved
5-0 ADDR	OTP write and read access address register. Specifies one of 128 word address locations (0x00 - 0x7f). If a valid access is accepted by the controller, the controller makes an internal copy of this value. This internal copy will not update until the access is complete.

23.6.1.6 OTP Controller Timing Register (HW_OCOTP_TIMING)

23.6.1.6.1 Offset

Register	Offset
HW_OCOTP_TIMING	10h

23.6.1.6.2 Function

This register specifies timing parameters for programming and reading the OCOTP fuse array.

23.6.1.6.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0				WAIT						STROBE_READ					
W																
Reset	0	0	0	0	0	1	1	0	0	0	0	0	1	1	0	1

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

23.6.1.6.4 Fields

Field	Function
31-28 —	Reserved
27-22 WAIT	This count value specifies time interval between auto read and write access in one time program. It is given in number of ipg_clk periods.
21-16 STROBE_READ	This count value specifies the strobe period in one time read OTP. Trd = ((STROBE_READ+1)-2*(RELAX+1)) /ipg_clk_freq. It is given in number of ipg_clk periods.
15-12 RELAX	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	This count value specifies the strobe period in one time write OTP. Tpgm = ((STROBE_PROG+1)-2*(RELAX+1)) /ipg_clk_freq. It is given in number of ipg_clk periods.

23.6.1.7 OTP Controller Write Data Register (HW_OCOTP_DATA)

23.6.1.7.1 Offset

Register	Offset
HW_OCOTP_DATA	20h

23.6.1.7.2 Function

This register is used in conjuction with HW_OCOTP_CTRL to perform one-time writes to the OTP. Please see the "Software Write Sequence" section for operating details.

23.6.1.7.3 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

23.6.1.7.4 Fields

Field	Function
31-0 DATA	Used to initiate a write to OTP. Please see the "Software Write Sequence" section for operating details.

23.6.1.8 OTP Controller Write Data Register (HW_OCOTP_READ_CTRL)

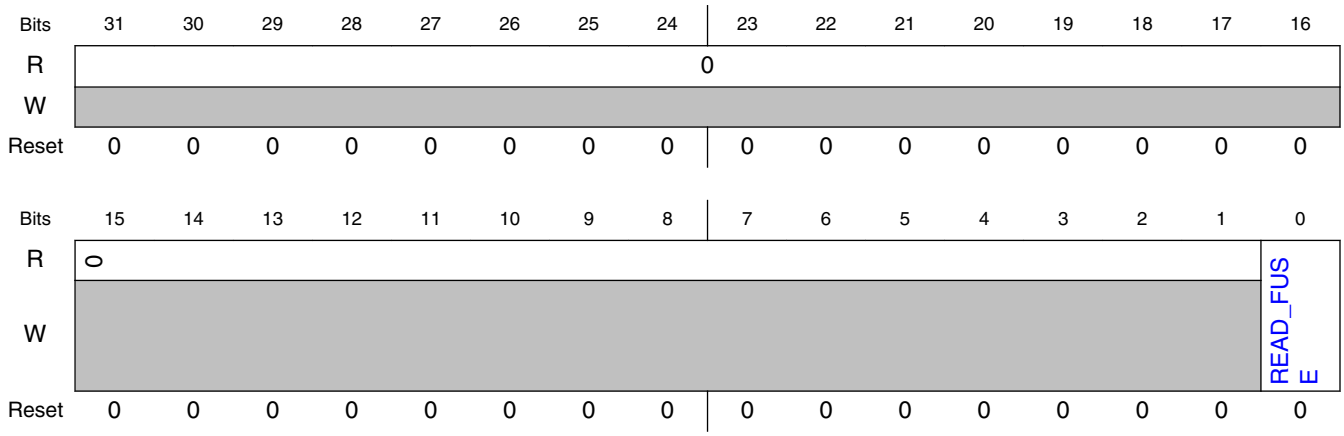
23.6.1.8.1 Offset

Register	Offset
HW_OCOTP_READ_CTRL	30h

23.6.1.8.2 Function

This register is used in conjuction with HW_OCOTP_CTRL to perform one time read to the OTP. Please see the "Software read Sequence" section for operating details.

23.6.1.8.3 Diagram



23.6.1.8.4 Fields

Field	Function
31-1 —	Reserved
0 READ_FUSE	Used to initiate a read to OTP. Please see the "Software read Sequence" section for operating details.

23.6.1.9 OTP Controller Read Data Register (HW_OCOTP_READ_FUSE_DATA)

23.6.1.9.1 Offset

Register	Offset
HW_OCOTP_READ_FUSE_DATA	40h

23.6.1.9.2 Function

The data read from OTP

23.6.1.9.3 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

23.6.1.9.4 Fields

Field	Function
31-0 DATA	The data read from OTP

23.6.1.10 Sticky bit Register (HW_OCOTP_SW_STICKY)

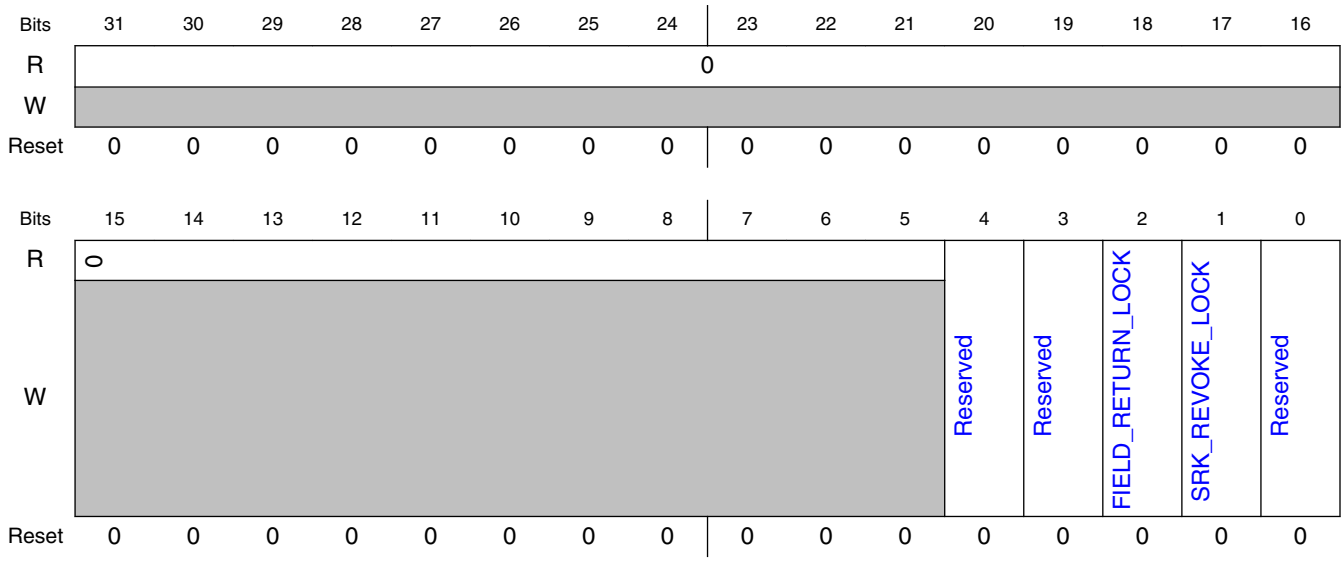
23.6.1.10.1 Offset

Register	Offset
HW_OCOTP_SW_STICKY	50h

23.6.1.10.2 Function

Some sticky bits are used by SW to lock some fuse area , shadow registers and other features.

23.6.1.10.3 Diagram



23.6.1.10.4 Fields

Field	Function
31-5 —	Reserved
4 —	Reserved
3 —	Reserved
2 FIELD_RETURN_LOCK	Shadow register write and OTP write lock for FIELD_RETURN region. When set, the writing of this region's shadow register and OTP fuse word are blocked.Once this bit is set, it is always high unless a POR is issued.
1 SRK_REVOKE_LOCK	Shadow register write and OTP write lock for SRK_REVOKE region. When set, the writing of this region's shadow register and OTP fuse word are blocked. Once this bit is set, it is always high unless a POR is issued.
0 —	Reserved

23.6.1.11 Software Controllable Signals Register (HW_OCOTP_SCS)

23.6.1.11.1 Offset

Register	Offset
HW_OCOTP_SCS	60h

23.6.1.11.2 Function

This register holds volatile configuration values that can be set and locked by trusted software. All values are returned to their defualt values after POR.

23.6.1.11.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	LOCK	SPARE														
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	SPARE															HAB_JDE
W																
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 LOCK	When set, all of the bits in this register are locked and can not be changed through SW programming. This bit is only reset after a POR is issued.
30-1 SPARE	Unallocated read/write bits for implementation specific software use.
0 HAB_JDE	HAB JTAG Debug Enable. This bit is used by the HAB to enable JTAG debugging, assuming that a properlay signed command to do so is found and validated by the HAB. The HAB must lock the register before passing control to the OS whether or not JTAG debugging has been enabled. Once JTAG is enabled by this bit, it can not be disabled unless the system is reset by POR. 0: JTAG debugging is not enabled by the HAB (it may still be enabled by other mechanisms). 1: JTAG debugging is enabled by the HAB (though this signal may be gated off).

23.6.1.12 Software Controllable Signals Register (HW_OCOTP_SCS_SET)

23.6.1.12.1 Offset

Register	Offset
HW_OCOTP_SCS_SET	64h

23.6.1.12.2 Function

This register holds volatile configuration values that can be set and locked by trusted software. All values are returned to their defualt values after POR.

23.6.1.12.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	LOCK	SPARE														
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	SPARE															HAB_JDE
W																
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 LOCK	When set, all of the bits in this register are locked and can not be changed through SW programming. This bit is only reset after a POR is issued.
30-1 SPARE	Unallocated read/write bits for implementation specific software use.
0 HAB_JDE	HAB JTAG Debug Enable. This bit is used by the HAB to enable JTAG debugging, assuming that a properlay signed command to do so is found and validated by the HAB. The HAB must lock the register before passing control to the OS whether or not JTAG debugging has been enabled. Once JTAG is enabled by this bit, it can not be disabled unless the system is reset by POR. 0: JTAG debugging is not enabled by the HAB (it may still be enabled by other mechanisms). 1: JTAG debugging is enabled by the HAB (though this signal may be gated off).

23.6.1.13 Software Controllable Signals Register (HW_OCOTP_SCS_CLR)

23.6.1.13.1 Offset

Register	Offset
HW_OCOTP_SCS_CLR	68h

23.6.1.13.2 Function

This register holds volatile configuration values that can be set and locked by trusted software. All values are returned to their defualt values after POR.

23.6.1.13.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	LOCK	SPARE														
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	SPARE															HAB_JDE
W																
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 LOCK	When set, all of the bits in this register are locked and can not be changed through SW programming. This bit is only reset after a POR is issued.
30-1 SPARE	Unallocated read/write bits for implementation specific software use.
0 HAB_JDE	HAB JTAG Debug Enable. This bit is used by the HAB to enable JTAG debugging, assuming that a properlay signed command to do so is found and validated by the HAB. The HAB must lock the register before passing control to the OS whether or not JTAG debugging has been enabled. Once JTAG is enabled by this bit, it can not be disabled unless the system is reset by POR. 0: JTAG debugging is not enabled by the HAB (it may still be enabled by other mechanisms). 1: JTAG debugging is enabled by the HAB (though this signal may be gated off).

23.6.1.14 Software Controllable Signals Register (HW_OCOTP_SCS_TOG)

23.6.1.14.1 Offset

Register	Offset
HW_OCOTP_SCS_TOG	6Ch

23.6.1.14.2 Function

This register holds volatile configuration values that can be set and locked by trusted software. All values are returned to their defualt values after POR.

23.6.1.14.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	LOCK	SPARE														
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	SPARE															HAB_JDE
W																
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 LOCK	When set, all of the bits in this register are locked and can not be changed through SW programming. This bit is only reset after a POR is issued.
30-1 SPARE	Unallocated read/write bits for implementation specific software use.
0 HAB_JDE	HAB JTAG Debug Enable. This bit is used by the HAB to enable JTAG debugging, assuming that a properlay signed command to do so is found and validated by the HAB. The HAB must lock the register before passing control to the OS whether or not JTAG debugging has been enabled. Once JTAG is enabled by this bit, it can not be disabled unless the system is reset by POR. 0: JTAG debugging is not enabled by the HAB (it may still be enabled by other mechanisms). 1: JTAG debugging is enabled by the HAB (though this signal may be gated off).

23.6.1.15 OTP Controller Version Register (HW_OCOTP_VERSION)

23.6.1.15.1 Offset

Register	Offset
HW_OCOTP_VERSION	90h

23.6.1.15.2 Function

This register indicates the RTL version in use.

23.6.1.15.3 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	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	STEP															
W																
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
31-24 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.
15-0 STEP	Fixed read-only value reflecting the stepping of the RTL version.

23.6.1.16 OTP Controller Timing Register 2 (HW_OCOTP_TIMING2)

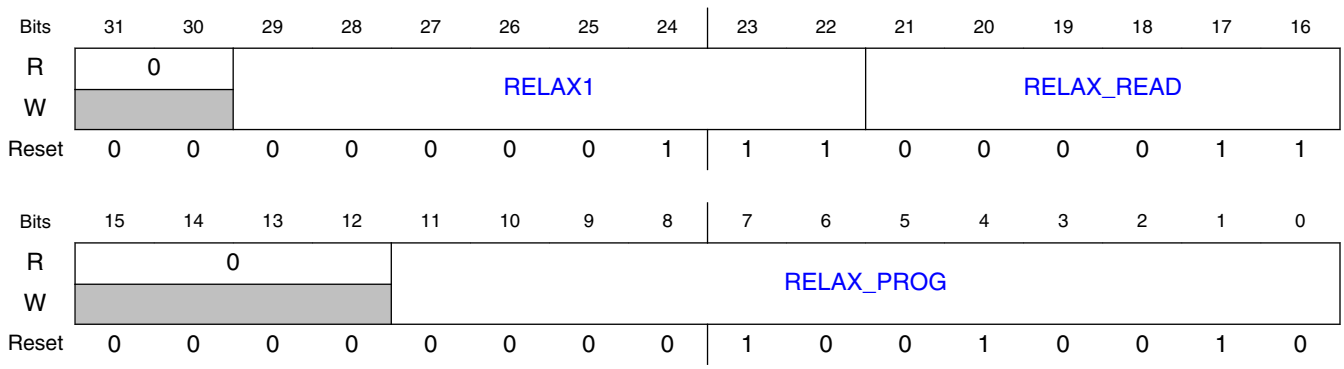
23.6.1.16.1 Offset

Register	Offset
HW_OCOTP_TIMING2	100h

23.6.1.16.2 Function

This register specifies timing parameters for programming and reading the OCOTP fuse array.

23.6.1.16.3 Diagram



23.6.1.16.4 Fields

Field	Function
31-30 —	Reserved
29-22 RELAX1	This count value specifies time interval between auto read and write access in one time program. It is given in number of ipg_clk periods.
21-16 RELAX_READ	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	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.

23.6.1.17 Value of OTP Bank0 Word0 (Lock controls) (HW_OCOTP_LOCK)

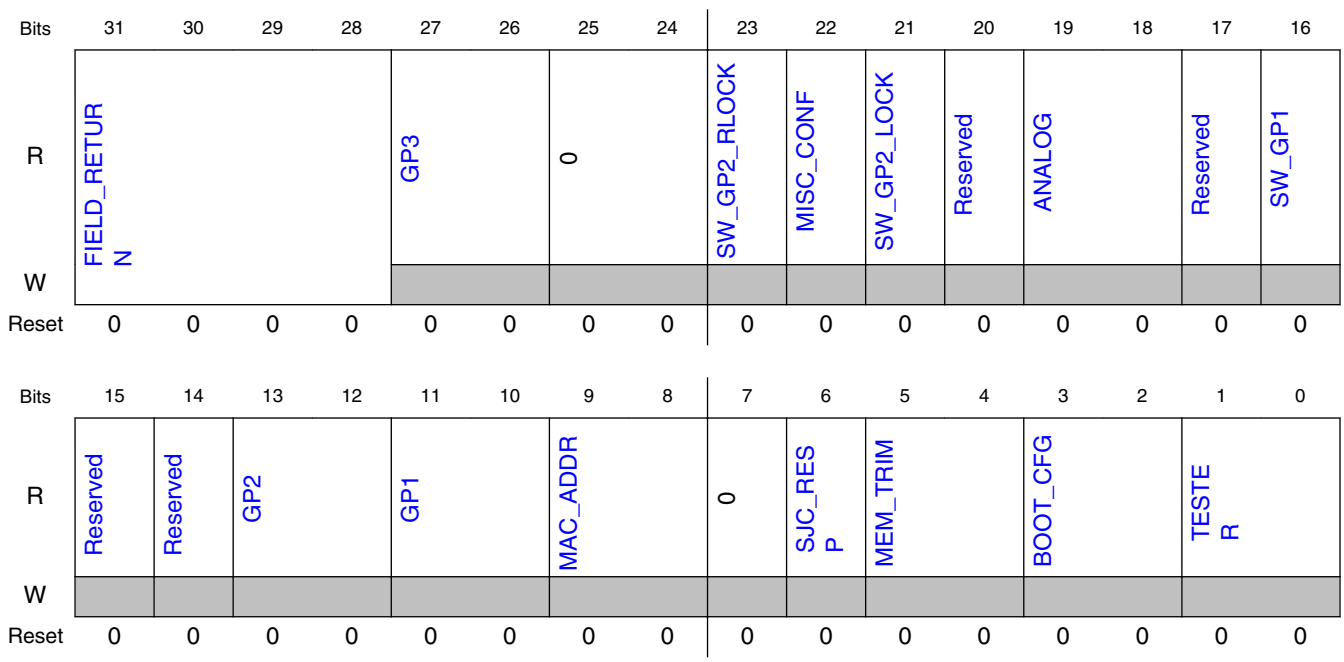
23.6.1.17.1 Offset

Register	Offset
HW_OCOTP_LOCK	400h

23.6.1.17.2 Function

Shadowed memory mapped access to OTP Bank 0, word 0 (ADDR = 0x00).

23.6.1.17.3 Diagram



23.6.1.17.4 Fields

Field	Function
31-28 FIELD_RETURN	Reserved
27-26 GP3	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.

Table continues on the next page...

Field	Function
25-24 —	Reserved
23 SW_GP2_RLOCK	Status of shadow register and OTP read lock for sw_gp2 region. When set, the reading of this region's shadow register and OTP fuse word are blocked.
22 MISC_CONF	Status of shadow register and OTP write lock for misc_conf region. When set, the writing of this region's shadow register and OTP fuse word are blocked.
21 SW_GP2_LOCK	Status of shadow register and OTP write lock for sw_gp2 region. When set, the writing of this region's shadow register and OTP fuse word are blocked.
20 -	Reserved
19-18 ANALOG	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	Status of shadow register and OTP write lock for sw_gp1 region. When set, the writing of this region's shadow register and OTP fuse word are blocked.
15 -	Reserved
14 —	Reserved
13-12 GP2	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	Status of shadow register and OTP write lock for gp1 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	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	Status of shadow register read and write, OTP read and write lock for sjc_resp region. 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 MEM_TRIM	Status of shadow register and OTP write lock for mem_trim 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.
3-2 BOOT_CFG	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 TESTER	Status of shadow register and OTP write lock for tester 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.

23.6.1.18 Value of OTP Bank0 Word1 (Configuration and Manufacturing Info.) (HW_OCOTP_CFG0)

23.6.1.18.1 Offset

Register	Offset
HW_OCOTP_CFG0	410h

23.6.1.18.2 Function

Shadowed memory mapped access to OTP Bank 0, word 1 (ADDR = 0x01).

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	This register contains 32 bits of the Unique ID and SJC_CHALLENGE field. Reflects value of OTP Bank 0, word 1 (ADDR = 0x01). These bits become read-only after the HW_OCOTP_LOCK_TESTER[1] bit is set.

23.6.1.19 Value of OTP Bank0 Word2 (Configuration and Manufacturing Info.) (HW_OCOTP_CFG1)

23.6.1.19.1 Offset

Register	Offset
HW_OCOTP_CFG1	420h

23.6.1.19.2 Function

shadowed memory mapped access to OTP Bank 0, word 2 (ADDR = 0x02).

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	This register contains 32 bits of the Unique ID and SJC_CHALLENGE field. Reflects value of OTP Bank 0, word 2 (ADDR = 0x02). These bits become read-only after the HW_OCOTP_LOCK_TESTER[1] bit is set.

23.6.1.20 Value of OTP Bank0 Word3 (Configuration and Manufacturing Info.) (HW_OCOTP_CFG2)

23.6.1.20.1 Offset

Register	Offset
HW_OCOTP_CFG2	430h

23.6.1.20.2 Function

Shadowed memory mapped access to OTP Bank 0, word 3 (ADDR = 0x03).

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	Reflects value of OTP Bank 0, word 3 (ADDR = 0x03). These bits become read-only after the HW_OCOTP_LOCK_TESTER[1] bit is set.

23.6.1.21 Value of OTP Bank0 Word4 (Configuration and Manufacturing Info.) (HW_OCOTP_CFG3)

23.6.1.21.1 Offset

Register	Offset
HW_OCOTP_CFG3	440h

23.6.1.21.2 Function

Non-shadowed memory mapped access to OTP Bank 0, word 4 (ADDR = 0x04).

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	Reflects value of OTP Bank 0, word 4 (ADDR = 0x04). These bits become read-only after the HW_OCOTP_LOCK_TESTER[1] bit is set.

23.6.1.22 Value of OTP Bank0 Word5 (Configuration and Manufacturing Info.) (HW_OCOTP_CFG4)

23.6.1.22.1 Offset

Register	Offset
HW_OCOTP_CFG4	450h

23.6.1.22.2 Function

Shadowed memory mapped access to OTP Bank 0, word 5 (ADDR = 0x05).

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	Reflects value of OTP Bank 0, word 5 (ADDR = 0x05). These bits become read-only after the HW_OCOTP_LOCK_BOOT_CFG[1] bit is set.

23.6.1.23 Value of OTP Bank0 Word6 (Configuration and Manufacturing Info.) (HW_OCOTP_CFG5)

23.6.1.23.1 Offset

Register	Offset
HW_OCOTP_CFG5	460h

23.6.1.23.2 Function

Shadowed memory mapped access to OTP Bank 0, word 6 (ADDR = 0x06).

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	Reflects value of OTP Bank 0, word 6 (ADDR = 0x06). These bits become read-only after the HW_OCOTP_LOCK_BOOT_CFG[1] bit is set.

23.6.1.24 Value of OTP Bank0 Word7 (Configuration and Manufacturing Info.) (HW_OCOTP_CFG6)

23.6.1.24.1 Offset

Register	Offset
HW_OCOTP_CFG6	470h

23.6.1.24.2 Function

Shadowed memory mapped access to OTP Bank 0, word 7 (ADDR = 0x07).

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	Reflects value of OTP Bank 0, word 7 (ADDR = 0x07). These bits become read-only after the HW_OCOTP_LOCK_BOOT_CFG[1] bit is set.

23.6.1.25 Value of OTP Bank1 Word0 (Memory Related Info.) (HW_OCOTP_MEM0)

23.6.1.25.1 Offset

Register	Offset
HW_OCOTP_MEM0	480h

23.6.1.25.2 Function

Shadowed memory mapped access to OTP bank 1, word 0 (ADDR = 0x08).

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	Reflects value of OTP bank 1, word 0 (ADDR = 0x08). These bits become read-only after the HW_OCOTP_LOCK_MEM_TRIM[1] bit is set.

23.6.1.26 Value of OTP Bank1 Word1 (Memory Related Info.) (HW_OCOTP_MEM1)

23.6.1.26.1 Offset

Register	Offset
HW_OCOTP_MEM1	490h

23.6.1.26.2 Function

Shadowed memory mapped access to OTP bank 1, word 1 (ADDR = 0x09).

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	Reflects value of OTP bank 1, word 1 (ADDR = 0x09). These bits become read-only after the HW_OCOTP_LOCK_MEM_TRIM[1] bit is set.

23.6.1.27 Value of OTP Bank1 Word2 (Memory Related Info.) (HW_OCOTP_MEM2)

23.6.1.27.1 Offset

Register	Offset
HW_OCOTP_MEM2	4A0h

23.6.1.27.2 Function

Shadowed memory mapped access to OTP bank 1, word 2 (ADDR = 0x0A).

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	Reflects value of OTP bank 1, word 2 (ADDR = 0x0A). These bits become read-only after the HW_OCOTP_LOCK_MEM_TRIM[1] bit is set.

23.6.1.28 Value of OTP Bank1 Word3 (Memory Related Info.) (HW_OCOTP_MEM3)

23.6.1.28.1 Offset

Register	Offset
HW_OCOTP_MEM3	4B0h

23.6.1.28.2 Function

Shadowed memory mapped access to OTP bank 1, word 3 (ADDR = 0x0B).

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	Reflects value of OTP bank 1, word 3 (ADDR = 0x0B). These bits become read-only after the HW_OCOTP_LOCK_MEM_TRIM[1] bit is set.

23.6.1.29 Value of OTP Bank1 Word4 (Memory Related Info.) (HW_OCOTP_MEM4)

23.6.1.29.1 Offset

Register	Offset
HW_OCOTP_MEM4	4C0h

23.6.1.29.2 Function

Shadowed memory mapped access to OTP bank 1, word 4 (ADDR = 0x0C).

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	Reflects value of OTP bank 1, word 4 (ADDR = 0x0C). These bits become read-only after the HW_OCOTP_LOCK_MEM_TRIM[1] bit is set.

23.6.1.30 Value of OTP Bank1 Word5 (Analog Info.) (HW_OCOTP_ANA0)

23.6.1.30.1 Offset

Register	Offset
HW_OCOTP_ANA0	4D0h

23.6.1.30.2 Function

Shadowed memory mapped access to OTP bank 1, word 5 (ADDR = 0x0D).

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	Reflects value of OTP bank 1, word 5 (ADDR = 0x0D). These bits become read-only after the HW_OCOTP_LOCK_ANALOG[1] bit is set.

23.6.1.31 Value of OTP Bank1 Word6 (Analog Info.) (HW_OCOTP_ANA1)

23.6.1.31.1 Offset

Register	Offset
HW_OCOTP_ANA1	4E0h

23.6.1.31.2 Function

Shadowed memory mapped access to OTP bank 1, word 6 (ADDR = 0x0E).

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	Reflects value of OTP bank 1, word 6 (ADDR = 0x0E). These bits become read-only after the HW_OCOTP_LOCK_ANALOG[1] bit is set.

23.6.1.32 Value of OTP Bank1 Word7 (Analog Info.) (HW_OCOTP_ANA2)

23.6.1.32.1 Offset

Register	Offset
HW_OCOTP_ANA2	4F0h

23.6.1.32.2 Function

Shadowed memory mapped access to OTP bank 1, word 7 (ADDR = 0x0F).

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	Reflects value of OTP bank 1, word 7 (ADDR = 0x0F). These bits become read-only after the HW_OCOTP_LOCK_ANALOG[1] bit is set.

23.6.1.33 Shadow Register for OTP Bank3 Word0 (SRK Hash) (HW_OCOTP_SRK0)

23.6.1.33.1 Offset

Register	Offset
HW_OCOTP_SRK0	580h

23.6.1.33.2 Function

Shadowed memory mapped access to OTP Bank 3, word 0 (ADDR = 0x18).

23.6.1.33.3 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

23.6.1.33.4 Fields

Field	Function
31-0 BITS	Shadow register for the hash of the Super Root Key word0 (Copy of OTP Bank 3, word 0 (ADDR = 0x1C)).

23.6.1.34 Shadow Register for OTP Bank3 Word1 (SRK Hash) (HW_OCOTP_SRK1)

23.6.1.34.1 Offset

Register	Offset
HW_OCOTP_SRK1	590h

23.6.1.34.2 Function

Shadowed memory mapped access to OTP Bank 3, word 1 (ADDR = 0x19).

23.6.1.34.3 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

23.6.1.34.4 Fields

Field	Function
31-0 BITS	Shadow register for the hash of the Super Root Key word1 (Copy of OTP Bank 3, word 1 (ADDR = 0x1D)).

23.6.1.35 Shadow Register for OTP Bank3 Word2 (SRK Hash) (HW_OCOTP_SRK2)

23.6.1.35.1 Offset

Register	Offset
HW_OCOTP_SRK2	5A0h

23.6.1.35.2 Function

Shadowed memory mapped access to OTP Bank 3, word 2 (ADDR = 0x1A).

23.6.1.35.3 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

23.6.1.35.4 Fields

Field	Function
31-0 BITS	Shadow register for the hash of the Super Root Key word2 (Copy of OTP Bank 3, word 2 (ADDR = 0x1E)).

23.6.1.36 Shadow Register for OTP Bank3 Word3 (SRK Hash) (HW_OCOTP_SRK3)

23.6.1.36.1 Offset

Register	Offset
HW_OCOTP_SRK3	5B0h

23.6.1.36.2 Function

Shadowed memory mapped access to OTP Bank 3, word 3 (ADDR = 0x1B).

23.6.1.36.3 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

23.6.1.36.4 Fields

Field	Function
31-0 BITS	Shadow register for the hash of the Super Root Key word3 (Copy of OTP Bank 3, word 3 (ADDR = 0x1F)).

23.6.1.37 Shadow Register for OTP Bank3 Word4 (SRK Hash) (HW_OCOTP_SRK4)

23.6.1.37.1 Offset

Register	Offset
HW_OCOTP_SRK4	5C0h

23.6.1.37.2 Function

Shadowed memory mapped access to OTP Bank 3, word 4 (ADDR = 0x1C).

23.6.1.37.3 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

23.6.1.37.4 Fields

Field	Function
31-0 BITS	Shadow register for the hash of the Super Root Key word4 (Copy of OTP Bank 3, word 4 (ADDR = 0x20)).

23.6.1.38 Shadow Register for OTP Bank3 Word5 (SRK Hash) (HW_OCOTP_SRK5)

23.6.1.38.1 Offset

Register	Offset
HW_OCOTP_SRK5	5D0h

23.6.1.38.2 Function

Shadowed memory mapped access to OTP Bank 3, word 5 (ADDR = 0x1D).

23.6.1.38.3 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

23.6.1.38.4 Fields

Field	Function
31-0 BITS	Shadow register for the hash of the Super Root Key word5 (Copy of OTP Bank 3, word 5 (ADDR = 0x21)).

23.6.1.39 Shadow Register for OTP Bank3 Word6 (SRK Hash) (HW_OCOTP_SRK6)

23.6.1.39.1 Offset

Register	Offset
HW_OCOTP_SRK6	5E0h

23.6.1.39.2 Function

Shadowed memory mapped access to OTP Bank 3, word 6 (ADDR = 0x1E).

23.6.1.39.3 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

23.6.1.39.4 Fields

Field	Function
31-0 BITS	Shadow register for the hash of the Super Root Key word6 (Copy of OTP Bank 3, word 6 (ADDR = 0x22)).

23.6.1.40 Shadow Register for OTP Bank3 Word7 (SRK Hash) (HW_OCOTP_SRK7)

23.6.1.40.1 Offset

Register	Offset
HW_OCOTP_SRK7	5F0h

23.6.1.40.2 Function

Shadowed memory mapped access to OTP Bank 3, word 7 (ADDR = 0x1F).

23.6.1.40.3 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

23.6.1.40.4 Fields

Field	Function
31-0 BITS	Shadow register for the hash of the Super Root Key word7 (Copy of OTP Bank 3, word 7 (ADDR = 0x23)).

23.6.1.41 Value of OTP Bank4 Word0 (Secure JTAG Response Field) (HW_OCOTP_SJC_RESP0)

23.6.1.41.1 Offset

Register	Offset
HW_OCOTP_SJC_RESP0	600h

23.6.1.41.2 Function

Shadowed memory mapped access to OTP Bank 4, word 0 (ADDR = 0x20).

23.6.1.41.3 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

23.6.1.41.4 Fields

Field	Function
31-0 BITS	Shadow register for the SJC_RESP Key word0 (Copy of OTP Bank 4, word 0 (ADDR = 0x20)). These bits can be not read and wrotten after the HW_OCOTP_LOCK_SJC_RESP bit is set. If read, returns 0xBADA_BADA and sets HW_OCOTP_CTRL[ERROR].

23.6.1.42 Value of OTP Bank4 Word1 (Secure JTAG Response Field) (HW_OCOTP_SJC_RESP1)

23.6.1.42.1 Offset

Register	Offset
HW_OCOTP_SJC_RESP1	610h

23.6.1.42.2 Function

Shadowed memory mapped access to OTP Bank 4, word 1 (ADDR = 0x21).

23.6.1.42.3 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

23.6.1.42.4 Fields

Field	Function
31-0 BITS	Shadow register for the SJC_RESP Key word1 (Copy of OTP Bank 4, word 1 (ADDR = 0x21)). These bits can be not read and wrotten after the HW_OCOTP_LOCK_SJC_RESP bit is set. If read, returns 0xBADA_BADA and sets HW_OCOTP_CTRL[ERROR].

23.6.1.43 Value of OTP Bank4 Word2 (MAC Address) (HW_OCOTP_MAC0)

23.6.1.43.1 Offset

Register	Offset
HW_OCOTP_MAC0	620h

23.6.1.43.2 Function

Shadowed memory mapped access to OTP Bank 4, word 2 (ADDR = 0x22).

23.6.1.43.3 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

23.6.1.43.4 Fields

Field	Function
31-0 BITS	Reflects value of OTP Bank 4, word 2 (ADDR = 0x22).

23.6.1.44 Value of OTP Bank4 Word3 (MAC Address) (HW_OCOTP_MAC1)

23.6.1.44.1 Offset

Register	Offset
HW_OCOTP_MAC1	630h

23.6.1.44.2 Function

Shadowed memory mapped access to OTP Bank 4, word 3 (ADDR = 0x23).

23.6.1.44.3 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

23.6.1.44.4 Fields

Field	Function
31-0 BITS	Reflects value of OTP Bank 4, word 3 (ADDR = 0x23).

23.6.1.45 Value of OTP Bank4 Word4 (MAC Address) (HW_OCOTP_GP3)

23.6.1.45.1 Offset

Register	Offset
HW_OCOTP_GP3	640h

23.6.1.45.2 Function

Shadowed memory mapped access to OTP Bank 4, word 4 (ADDR = 0x24).

23.6.1.45.3 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

23.6.1.45.4 Fields

Field	Function
31-0 BITS	Reflects value of OTP Bank 4, word 4 (ADDR = 0x24).

23.6.1.46 Value of OTP Bank4 Word6 (General Purpose Customer Defined Info) (HW_OCOTP_GP1)

23.6.1.46.1 Offset

Register	Offset
HW_OCOTP_GP1	660h

23.6.1.46.2 Function

Shadowed memory mapped access to OTP Bank 4, word 6 (ADDR = 0x26).

23.6.1.46.3 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

23.6.1.46.4 Fields

Field	Function
31-0 BITS	Reflects value of OTP Bank 4, word 6 (ADDR = 0x26).

23.6.1.47 Value of OTP Bank4 Word7 (General Purpose Customer Defined Info) (HW_OCOTP_GP2)

23.6.1.47.1 Offset

Register	Offset
HW_OCOTP_GP2	670h

23.6.1.47.2 Function

Shadowed memory mapped access to OTP Bank 4, word 7 (ADDR = 0x27).

23.6.1.47.3 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

23.6.1.47.4 Fields

Field	Function
31-0 BITS	Reflects value of OTP Bank 4, word 7 (ADDR = 0x27).

23.6.1.48 Value of OTP Bank5 Word0 (SW GP1) (HW_OCOTP_SW_GP1)

23.6.1.48.1 Offset

Register	Offset
HW_OCOTP_SW_GP1	680h

23.6.1.48.2 Function

Shadowed memory mapped access to OTP Bank 5, word 0 (ADDR = 0x28).

23.6.1.48.3 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

23.6.1.48.4 Fields

Field	Function
31-0 BITS	Reflects value of OTP Bank 5, word 0 (ADDR = 0x28).

23.6.1.49 Value of OTP Bank5 Word1 (SW GP2) (HW_OCOTP_SW_GP20)

23.6.1.49.1 Offset

Register	Offset
HW_OCOTP_SW_GP20	690h

23.6.1.49.2 Function

Shadowed memory mapped access to OTP Bank 5, word 1 (ADDR = 0x29).

23.6.1.49.3 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

23.6.1.49.4 Fields

Field	Function
31-0 BITS	Reflects value of OTP Bank 5, word 1 (ADDR = 0x29).

23.6.1.50 Value of OTP Bank5 Word2 (SW GP2) (HW_OCOTP_SW_GP21)

23.6.1.50.1 Offset

Register	Offset
HW_OCOTP_SW_GP21	6A0h

23.6.1.50.2 Function

Shadowed memory mapped access to OTP Bank 5, word 2 (ADDR = 0x2a).

23.6.1.50.3 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

23.6.1.50.4 Fields

Field	Function
31-0 BITS	Reflects value of OTP Bank 5, word 2 (ADDR = 0x2a).

23.6.1.51 Value of OTP Bank5 Word3 (SW GP2) (HW_OCOTP_SW_GP22)

23.6.1.51.1 Offset

Register	Offset
HW_OCOTP_SW_GP22	6B0h

23.6.1.51.2 Function

Shadowed memory mapped access to OTP Bank 5, word 3 (ADDR = 0x2b).

23.6.1.51.3 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

23.6.1.51.4 Fields

Field	Function
31-0 BITS	Reflects value of OTP Bank 5, word 3 (ADDR = 0x2b).

23.6.1.52 Value of OTP Bank5 Word4 (SW GP2) (HW_OCOTP_SW_GP23)

23.6.1.52.1 Offset

Register	Offset
HW_OCOTP_SW_GP23	6C0h

23.6.1.52.2 Function

Shadowed memory mapped access to OTP Bank 5, word 4 (ADDR = 0x2c).

23.6.1.52.3 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

23.6.1.52.4 Fields

Field	Function
31-0 BITS	Reflects value of OTP Bank 5, word 4 (ADDR = 0x2c).

23.6.1.53 Value of OTP Bank5 Word5 (Misc Conf) (HW_OCOTP_MISC_CONF0)

23.6.1.53.1 Offset

Register	Offset
HW_OCOTP_MISC_CONF0	6D0h

23.6.1.53.2 Function

Shadowed memory mapped access to OTP Bank 5, word 5 (ADDR = 0x2d).

23.6.1.53.3 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

23.6.1.53.4 Fields

Field	Function
31-0 BITS	Reflects value of OTP Bank 5, word 5 (ADDR = 0x2d).

23.6.1.54 Value of OTP Bank5 Word6 (Misc Conf) (HW_OCOTP_MISC_CONF1)

23.6.1.54.1 Offset

Register	Offset
HW_OCOTP_MISC_CONF1	6E0h

23.6.1.54.2 Function

Shadowed memory mapped access to OTP Bank 5, word 6 (ADDR = 0x2e).

23.6.1.54.3 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

23.6.1.54.4 Fields

Field	Function
31-0 BITS	Reflects value of OTP Bank 5, word 6 (ADDR = 0x2e).

23.6.1.55 Value of OTP Bank5 Word7 (SRK Revoke) (HW_OCOTP_SRK_REVOKE)

23.6.1.55.1 Offset

Register	Offset
HW_OCOTP_SRK_REVOKE	6F0h

23.6.1.55.2 Function

Shadowed memory mapped access to OTP Bank 5, word 7 (ADDR = 0x2f).

23.6.1.55.3 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

23.6.1.55.4 Fields

Field	Function
31-0 BITS	Reflects value of OTP Bank 5, word 7 (ADDR = 0x2f).

Chapter 24
External Memory Controllers

24.1 Overview

This chip has these external memory interfaces and controllers:

- Quad Serial Peripheral Interface (QSPI)

24.2 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 133MHz SDR Mode and 66MHz DDR Mode (with internal DQS loopback mode)

Chapter 25
FlexSPI Controller

25.1 Chip-specific FlexSPI information

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.

NOTE

ARM AXIM access to FlexSPI is converted to AHB command access of FlexSPI, on this device.

Master ID allocation

This table summarizes the master IDs for all system master modules:

Table 25-1. Master IDs

Module	Master ID
Core Platform	000b
eDMA	001b
DCP	010b
All others	011b

Table 25-2. 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

Table continues on the next page...

Table 25-2. Reference links to related information (continued)

Topic	Related module(s)	Reference
		Power Management Unit
Signal multiplexing	IOMUX	External Signals and Pin Multiplexing IOMUX

25.2 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.

25.2.1 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
  - Serial NAND Flash
  - HyperBus device (HyperFlash/HyperRAM)
  - 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. Total AHB RX Buffer size: 128 * 64 Bits
  - 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. AHB TX Buffer size: 8 * 64 Bits
- 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: 16 * 64 Bits
  - IP TX FIFO implemented to buffer all Write data to External device. FIFO size: 16 * 64 Bits
  - 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

25.2.2 Block diagram

The block diagram of FlexSPI is indicated in following figure:

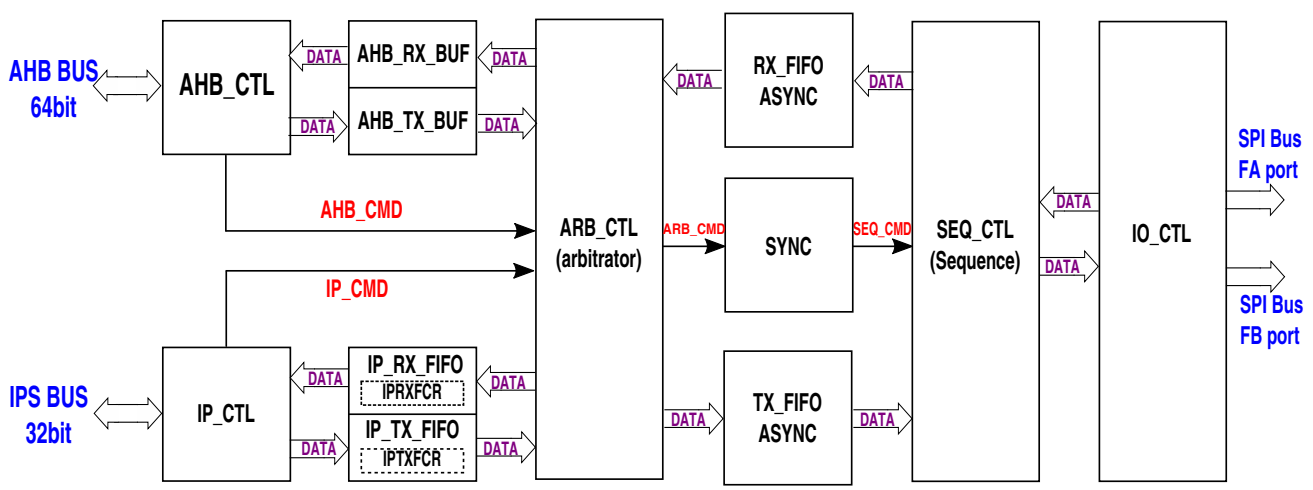


Figure 25-1. FlexSPI block diagram

25.2.3 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.

25.3 Glossary for FlexSPI module

Table 25-3. Glossary

Term	Definition
AHB Command	AHB Command is one or several Serial Flash Memory access command sequences triggered by AHB read/write access to AHB address ranged mapped for Serial Flash Memory.
ASFM_BASE	Base address of AHB address space mapped to Serial Flash Memory.
ARDF_BASE	Base address of AHB address space mapped to IP RX FIFO.
ATDF_BASE	Base address of AHB address space mapped to IP TX FIFO.
Set	Write 1 to register bit to establish logic level one on the bit.
Clear	Write 0 to register bit to establish logic level zero on the bit.
Command Sequence	A command sequence is 4*32 bits sequence code consisting of up to 8 instructions. Command sequence should be programmed in LUT according to Flash command. When a command sequence executed, the chip selection signal becomes asserted. When a command sequence execution finished, the chip selection signal becomes negated.
DDR	Dual data transfer rate for flash access mode Flash receive data on both SCLK rise and fall edge and transmit data on both SCLK rise and fall edge.
Endianness	Byte Ordering scheme.
Field	Two or more register bits grouped together.
Fill	To add entries to a FIFO by software or hardware.
Instruction Code	16 bits code defining the type of command to be executed on FlexSPI interface.
IP Command	IP Command is one or several Serial Flash Memory access command sequences triggered by set register bit IPCMD.TRG.
Negated	A signal that is negated is in its inactive state. An active low signal changes from logic level 0 to logic level 1 when negated, and an active high signal changes from logic level 1 to logic level 0.
SDR	Single data transfer rate for flash access mode Flash receive data on SCLK rise edge and transmit data on SCLK fall edge.
Set	To set a bit or bits means to establish logic level one on the bit or bits.
SFM	Serial Flash Memory
Individual Mode	Access to a single, individual serial flash device at a time.
Parallel Mode	Read/Program Access to two serial flash devices in parallel (Port A and Port B). FlexSPI will split flash program data before transmitting to Flash or merge flash read data before putting into IP RX FIFO or AHB RX Buffer automatically.
Memory Command	Serial access Command for reading/programming/configuring. A memory command may consist one or several command sequences, for each command sequence the Chip selection signal will be asserted and deasserted once.
LUT	Look-up table to preserve command sequence. LUT is programmed by software with command sequences which is used to issue memory commands.

25.4 External Signal Description

This section provides the external signal information of the FlexSPI module.

Table 25-4. 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. 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:

Table continues on the next page...

Table 25-4. External Signal List (continued)

Signal Name	Function	Direction	Description
			• 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) such as HyperRAM/ HyperFlash. • 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 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

25.5 Functional description

The following sections describe functional details of the FlexSPI module.

25.5.1 Clocks

This section describes clocks and special clocking requirements of the FlexSPI module.

Table 25-5. 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	-

Table continues on the next page...

Table 25-5. Clock Usage (continued)

Clock Name	Description	Comment
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.

25.5.2 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 error interrupt**

  When AHB bus response timeout, or AHB read access during OTFAD is enabled and FlexSPI is still fetching the key blob, there will be interrupt generated if INTEN[AHBBUSERROREN] 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.

- **Key done interrupt**

When OTFAD is enabled and key blob processing is enabled, a interrupt will be generated after key blob processing is done and no error detected if INTEN[KEYDONEEN] is set to 0x1.

- **Key error interrupt**

  When OTFAD is enabled and key blob processing is enabled, a interrupt will be generated if error happened during key blob processing if INTEN[KEYERROREN] is set to 0x1. Once this interrupt is set, it can only be cleared by reset.

25.5.3 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:

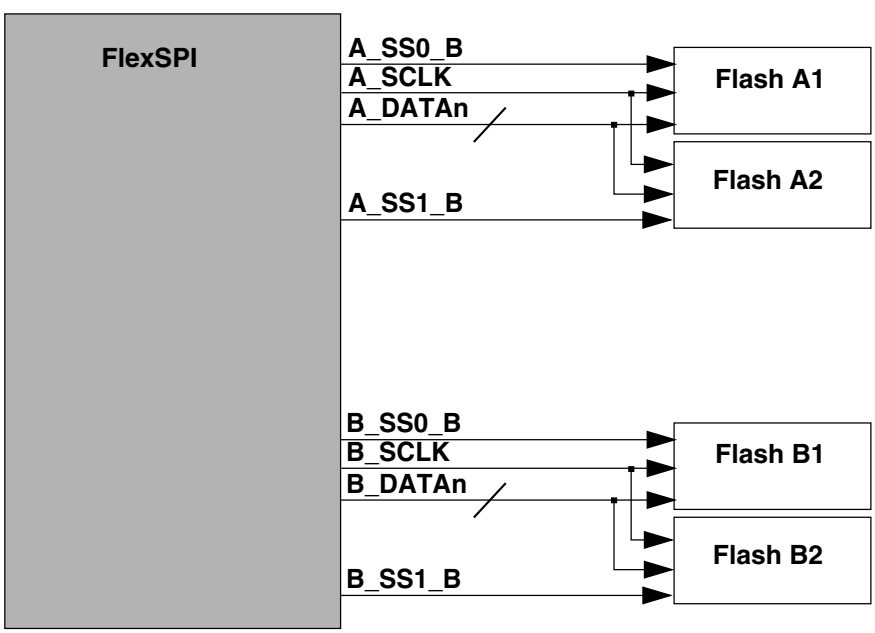


Figure 25-2. Flash connection diagram with four devices

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. The connection diagram for this combination mode is as following:

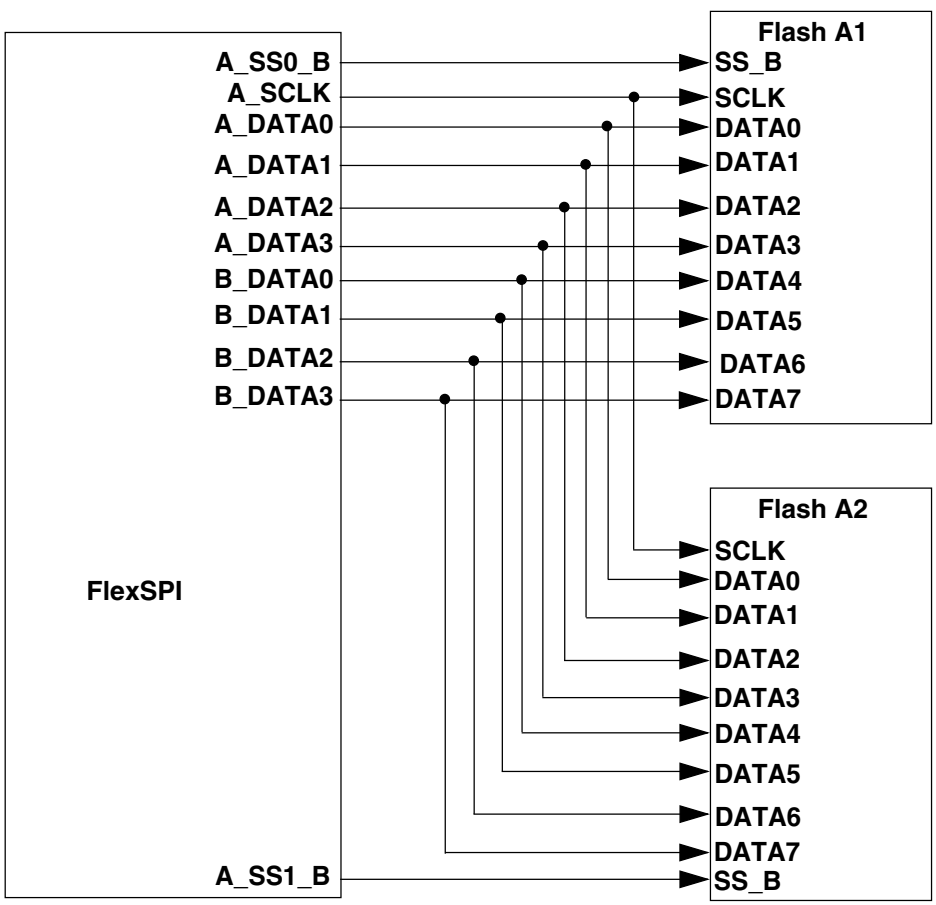


Figure 25-3. Flash connection diagram with combination mode

25.5.4 Flash Access mode

This section describes flash access mode.

25.5.4.1 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.

25.5.4.2 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 mode and parallel mode is determined statically by register field IPCR1[IPAREN] (for IP command) or AHBCR[APAREN] (for AHB command).

25.5.4.3 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.

25.5.4.4 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.

25.5.5 Flash memory map

Flash memory map in individual and parallel mode is as following:

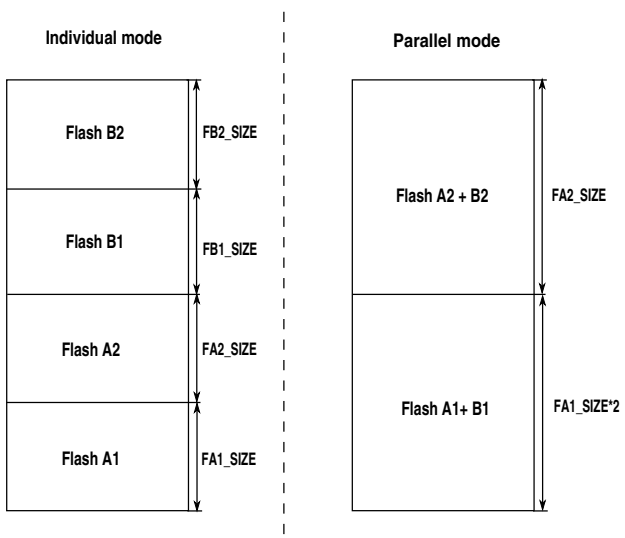


Figure 25-4. Flash memory map in individual and parallel mode

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.

25.5.6 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 25-6. 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, or OTFAD is enabled, FlexSPI will guarantee flash access start address and data size are 64 bit aligned by hardware.

   When AHBCR[READADDROPT] is set to 1, or OTFAD is enabled, 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.

25.5.7 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.

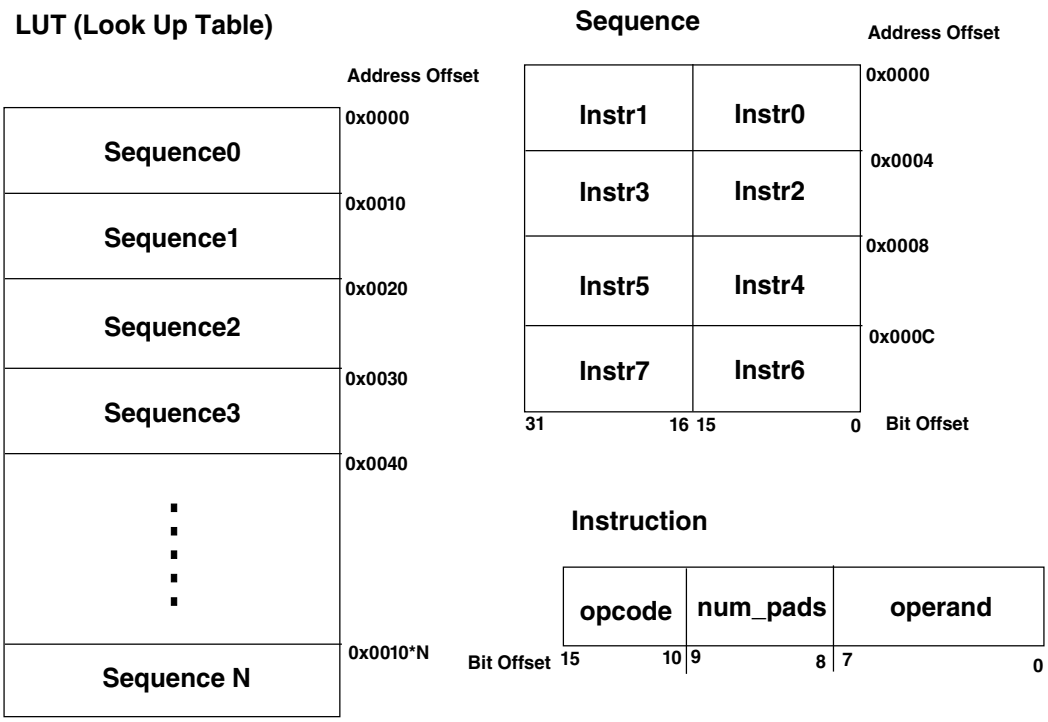


Figure 25-5. LUT and sequence structure

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].

25.5.8 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 25-7. Instruction set

Name	Opcode	Num_pads	Action on SPI interface	Transmit Data	Bits/Bytes/Cycle Number
CMD_SDR/	6'h01/	2'h0 - one pad (Single mode)	Transmit Command code to Flash	Command code:	Bit number: 8

Table continues on the next page...

Table 25-7. Instruction set (continued)

Name	Opcode	Num_pads	Action on SPI interface	Transmit Data	Bits/Bytes/Cycle Number
CMD_DDR	6'h21	2'h1 - two pad (Dual mode) 2'h2 - four pad (Quad mode) 2'h3 - eight pad (Octal mode)		Operand[7:0]	
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
WRITE_SDR/ WRITE_DDR	6'h08/ 6'h28		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.	-	Byte number: operand[7:0] Never set operand to zero for LEARN instruction.
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	-	Dummy cycle number (in serial root clock): Operand[7:0] **Dummy cycle (N), described in the Flash device datasheet is in number of**

Table continues on the next page...

Table 25-7. Instruction set (continued)

Name	Opcode	Num_pads	Action on SPI interface	Transmit Data	Bits/Bytes/Cycle Number
			will determine the number of pads in input mode.		**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.**
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. DQS pins is called "RWDS" in HyperBus specification. Refer Dummy instruction for more details. **Set operand as "Latency count" for HyperBus devices.**	-	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	-	No transaction on SPI interface.

Table continues on the next page...

Table 25-7. Instruction set (continued)

Name	Opcode	Num_pads	Action on SPI interface	Transmit Data	Bits/Bytes/Cycle Number
			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]).		
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.

25.5.8.1 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 date 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 f(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 date 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_DDR could be used for HyperBus device which use "RWDS" pin to indicate whether extra latency count needed. DUMMY_RWDS_SDR is reserved for future. FlexSPI controller checks DQS pin input level at the 4th cycle after SCLK output toggling is enabled. DQS pins is called "RWDS" in HyperBus specification. 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.

25.5.8.2 Flash access sequence example

Following is an example for SDR single I/O Read sequence (Cypress Serial Nor Flash S25FS512S) in individual mode.

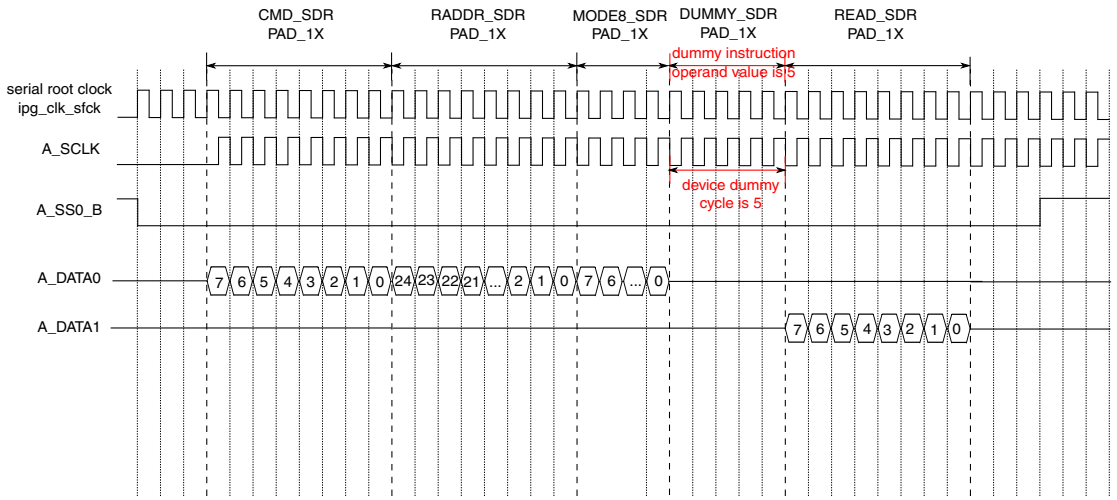


Figure 25-6. Flash access sequence example (SDR Single I/O Read sequence)

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.

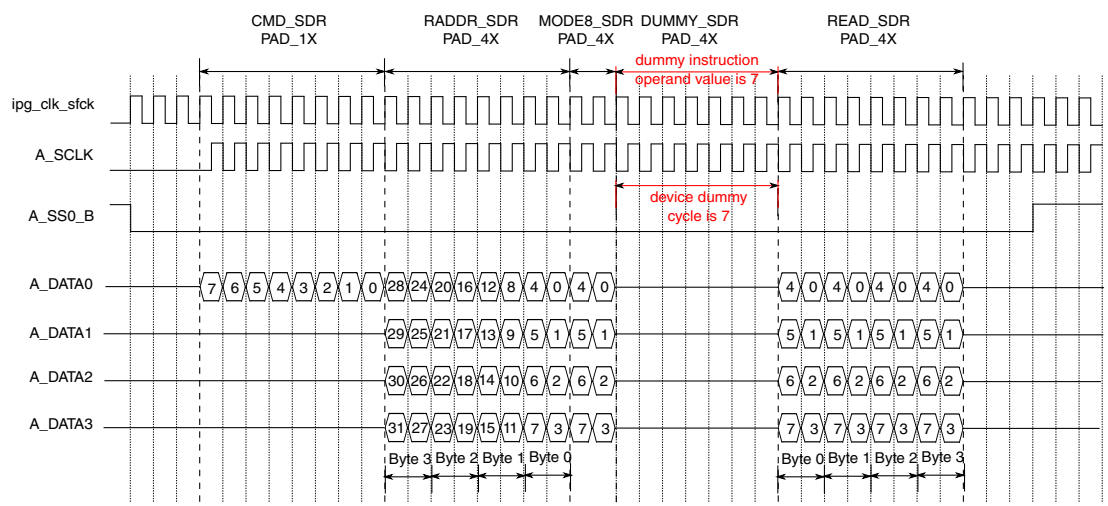


Figure 25-7. Flash access sequence example (SDR Quad I/O Read sequence)

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.

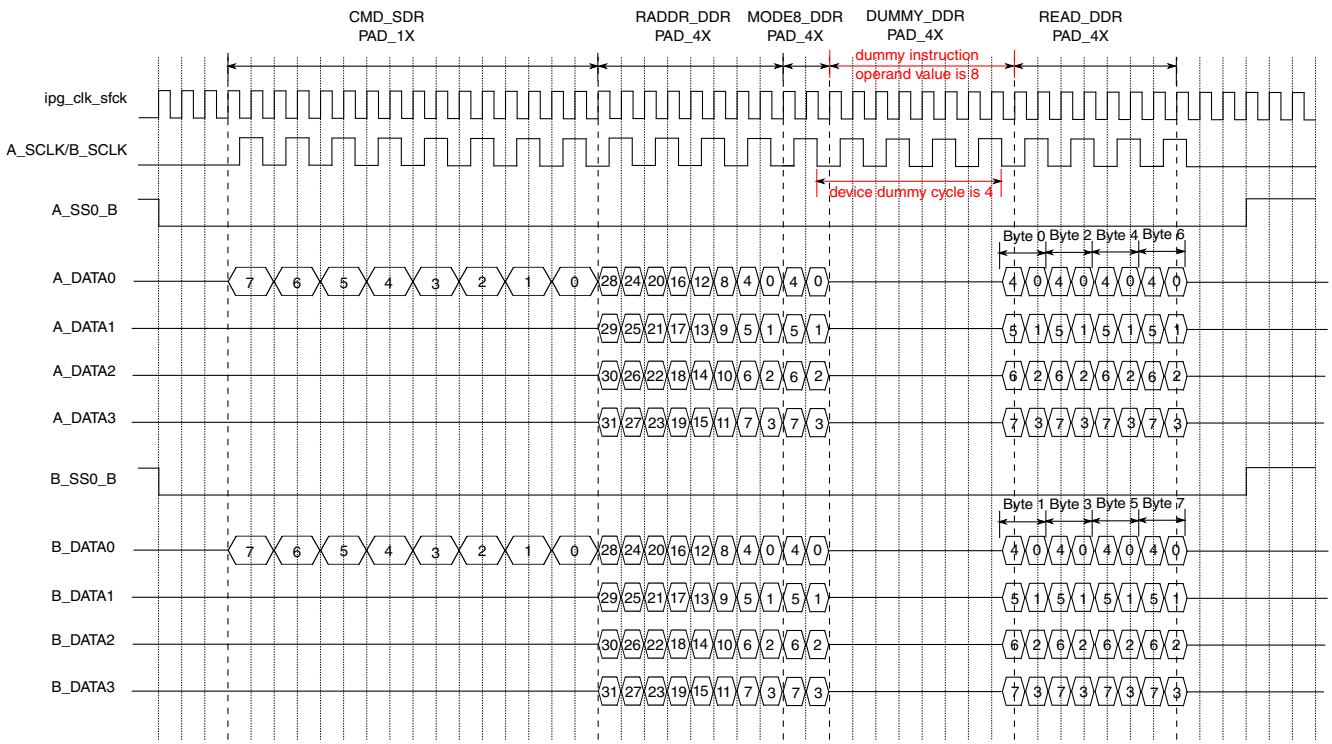


Figure 25-8. Flash access sequence example (DDR Quad I/O Read sequence)

Following is an example for HyperBus device read transaction (Single latency count) in individual mode.

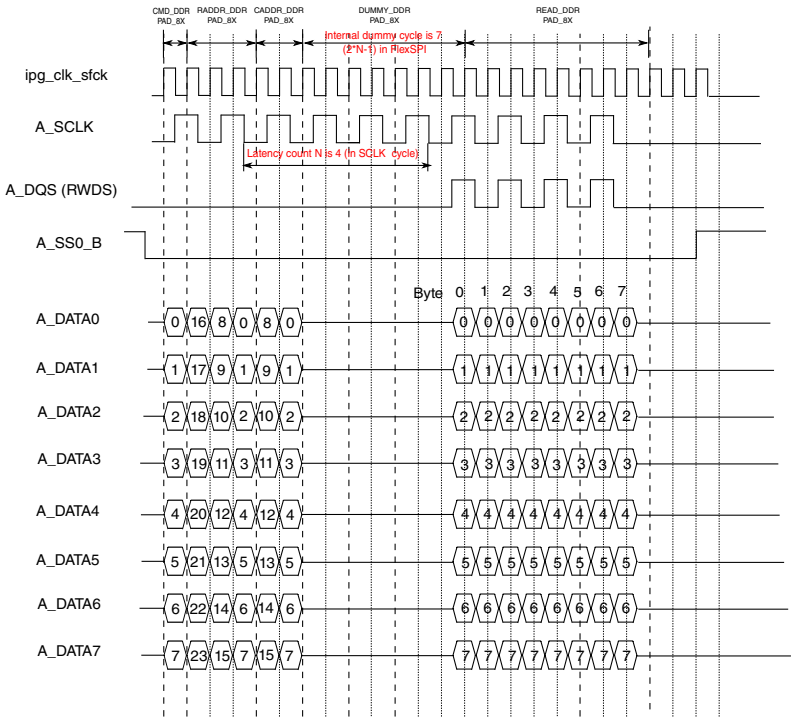


Figure 25-9. HyperBus device read transaction with single latency count

Following is an example for HyperBus device read transaction (Additional latency count) in individual mode.

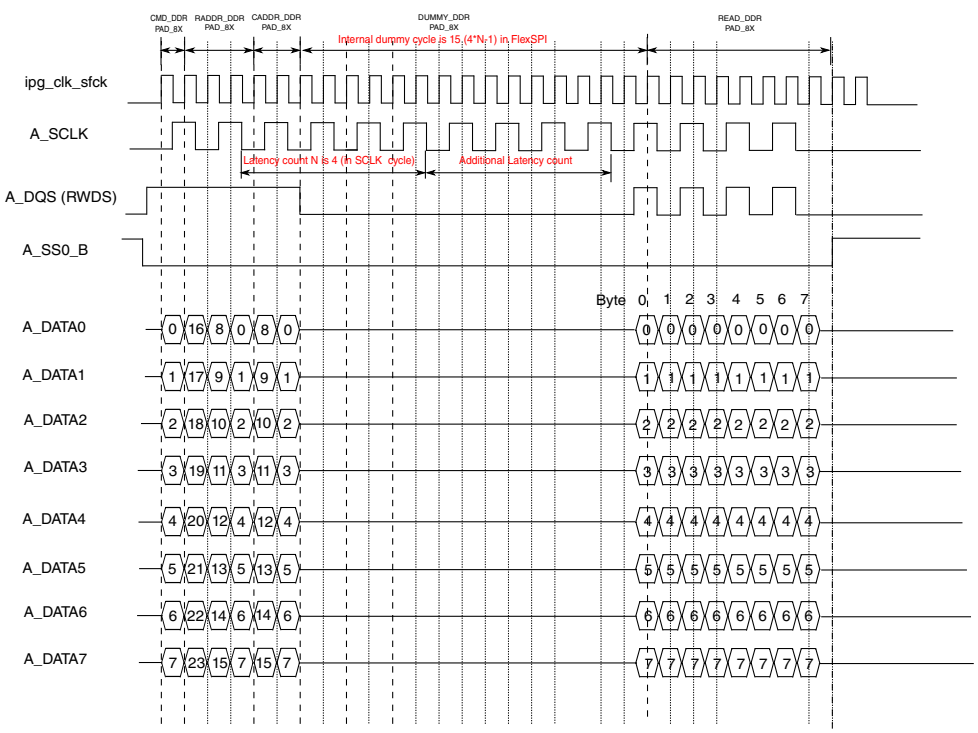


Figure 25-10. HyperBus device read transaction with additional latency count

Following is an example for HyperBus device write transaction (Single latency count) in individual mode.

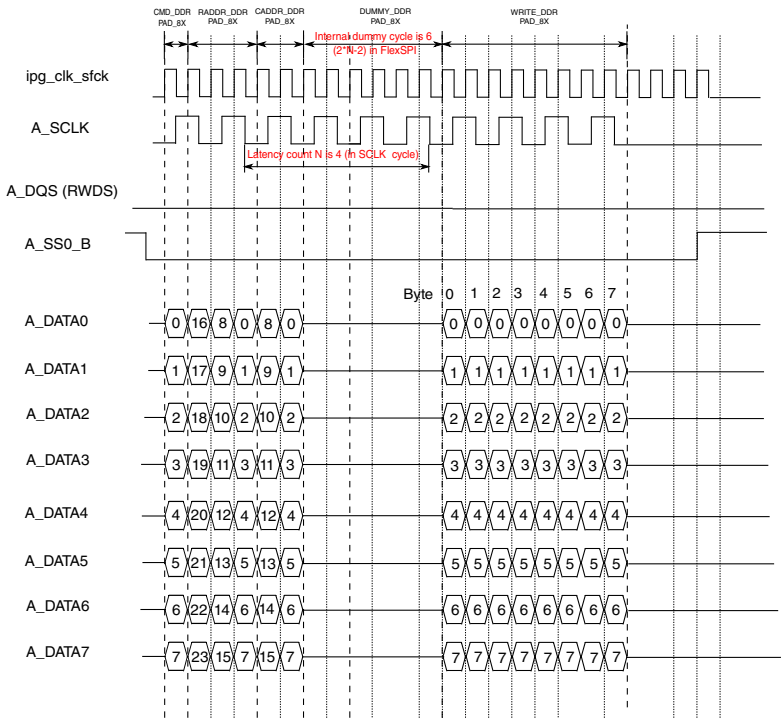


Figure 25-11. HyperBus device write transaction with single latency count

Following is an example for HyperBus device write transaction (Additional latency count) in individual mode.

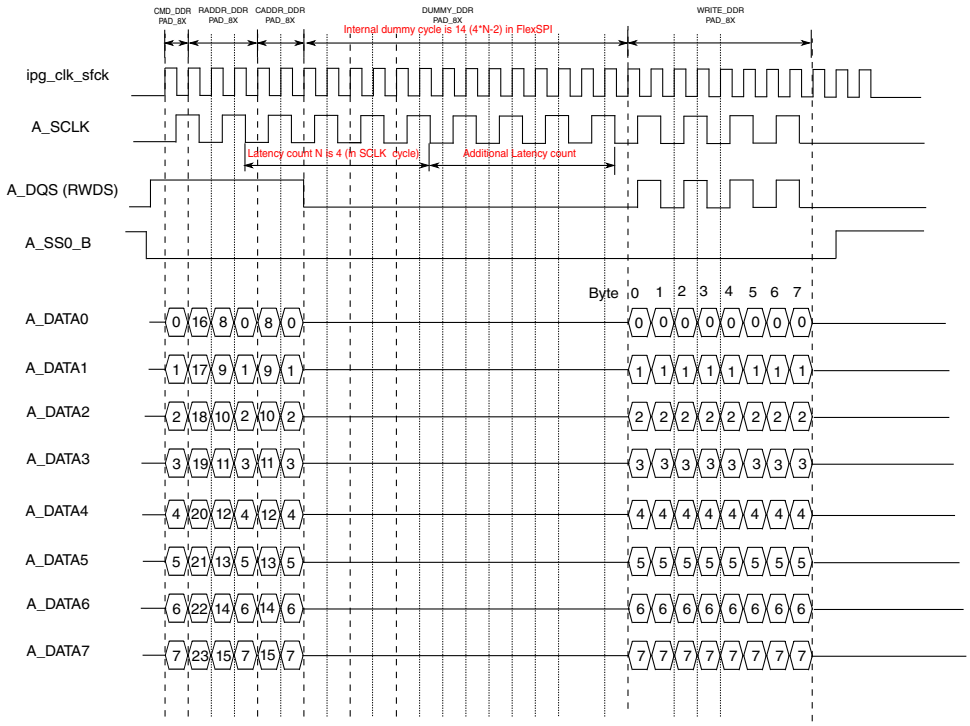


Figure 25-12. HyperBus device write transaction with additional latency count

25.5.9 OTFAD Support

This section describes OTFAD(On-The-Fly AES Decryption) support of the FlexSPI module.

After power up if the OTFAD and key fetches are enabled in the system, FlexSPI automatically starts fetching the key blobs from location 0 of the flashA1. The key fetches continue until the OTFAD receives and processes all key blobs. During the key blob fetch duration any AHB access is responded to with an error response and INT[AHBBUSERROR] is set. During Key blob fetch any IPS access is not allowed. Key blob processing result will be represented by INT[KEYDONE] or INT[KEYERROR]

Once the OTFAD processes all key blobs, any reads from the FlexSPI are first decrypted by the OTFAD engine and then returned on the AHB bus. In case OTFAD is enabled, the FlexSPI will always enable the prefetch, only AHBCR[READSZALIGN] can disable the prefetch. Read Resume function will also be disabled. All read access address will be 8 bytes aligned automatically regardless the setting of AHBCR[ADDROPT]. If OTFAD is enabled, the prefetch never crosses the 1KB boundary.

25.5.10 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 IP RX FIFO is not full. Please refer to SCLK stop feature for more detail.

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.

25.5.10.1 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.

- 0x100 - 0x180 (by IPS Bus)
- 0x7FC00000 - 0x7FC00080 (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 digram indicates the reading flow from IP RX FIFO by processor.

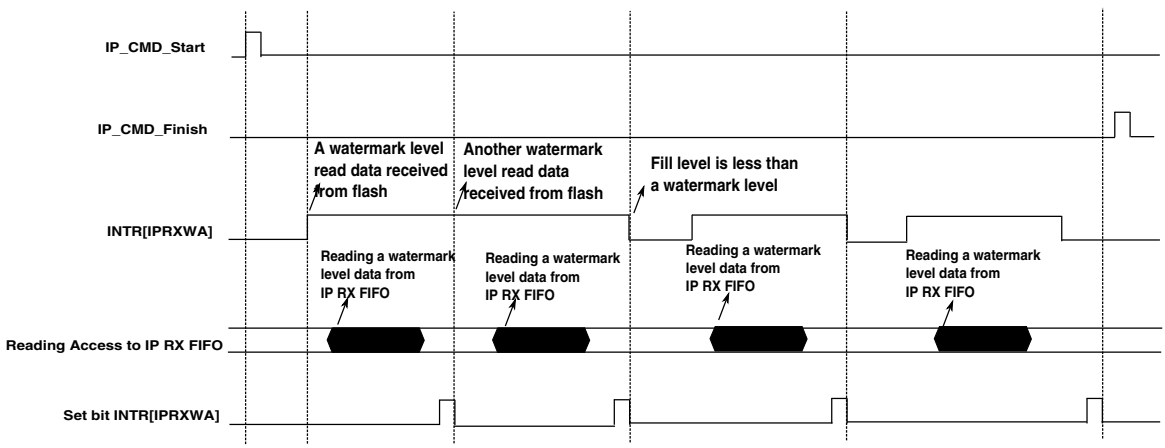


Figure 25-13. Reading IP RX FIFO by processor

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.

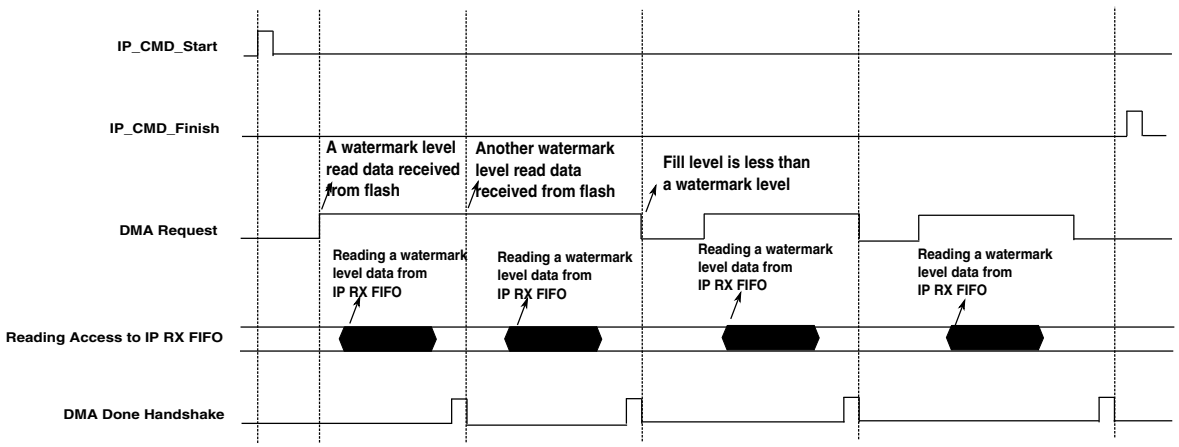


Figure 25-14. Reading from IP RX FIFO by DMA

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.

25.5.10.2 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.

- 0x180 - 0x200 (by IP Bus)
- 0x7F800000 - 0x7F800080 (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.

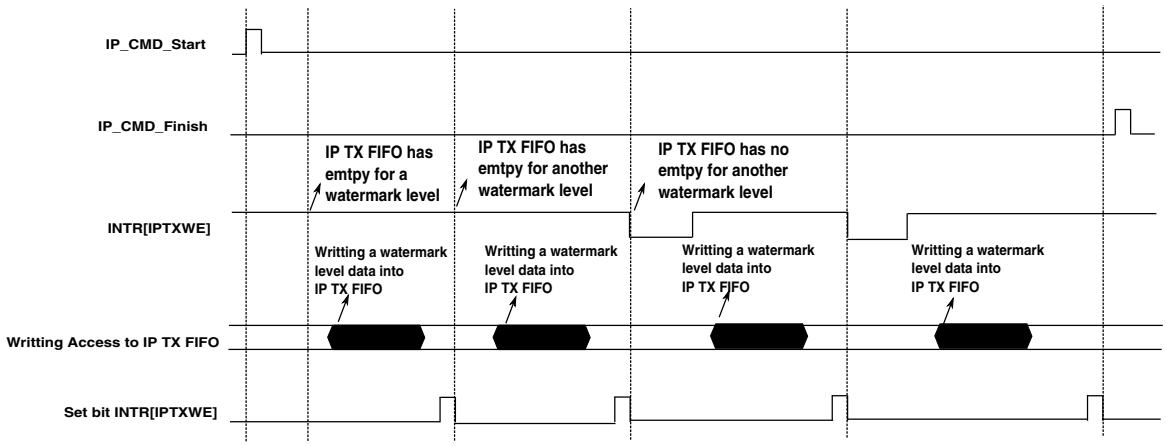


Figure 25-15. Filling IP TX FIFO by processor

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.

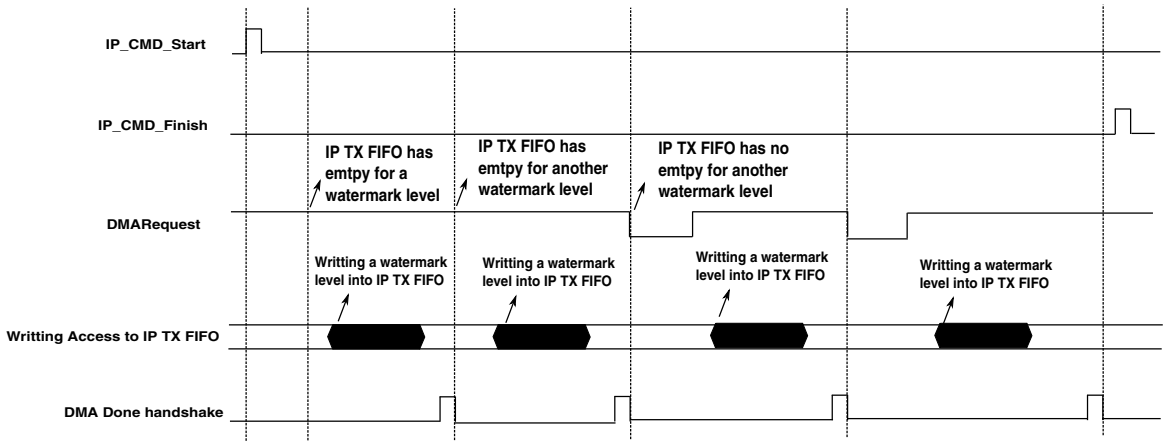


Figure 25-16. Filling from IP TX FIFO by DMA

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.

25.5.11 Flash access by AHB Command

Flash could be accessed by AHB bus directly on AHB address space: 60000000~0x80000000. This address space is mapped to Serial Flash Memory in FlexSPI. AHB bus access to this address space may trigger Flash access command sequence 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.

Following section described AHB command 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.

25.5.11.1 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.

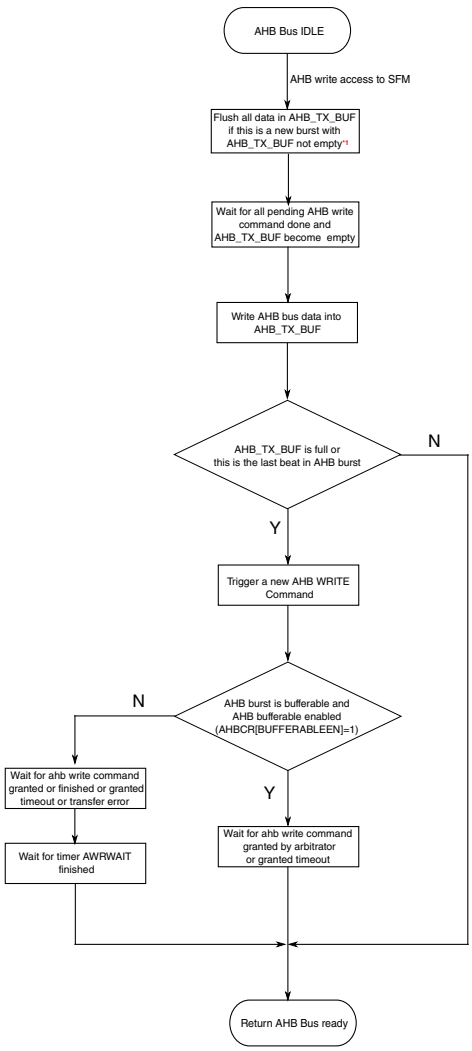


Figure 25-17. Hardware operation in responsee to AHB write access to Flash

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 25-18.

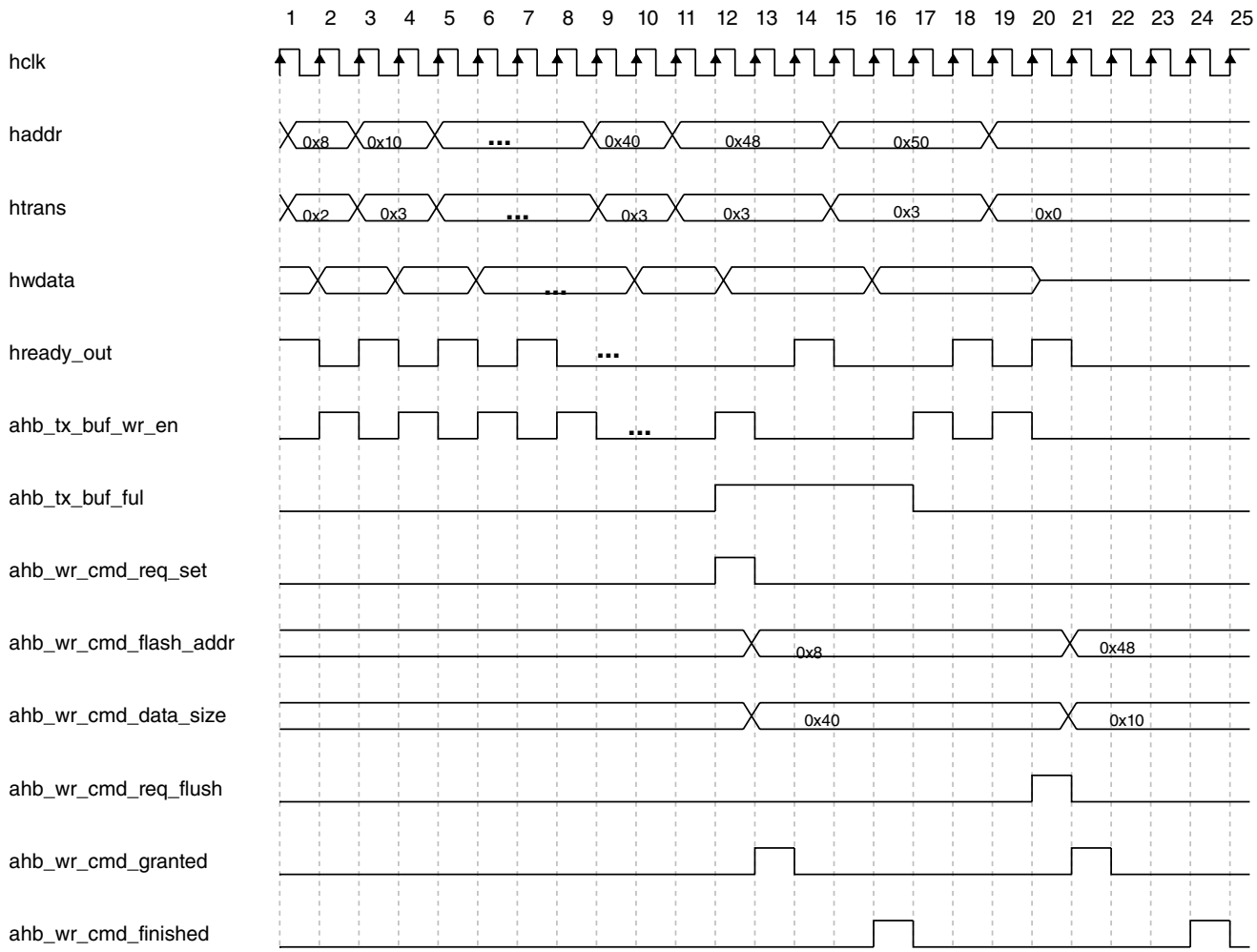


Figure 25-18. AHB write access (INCR/64bit/Bufferable)

- 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 25-19.

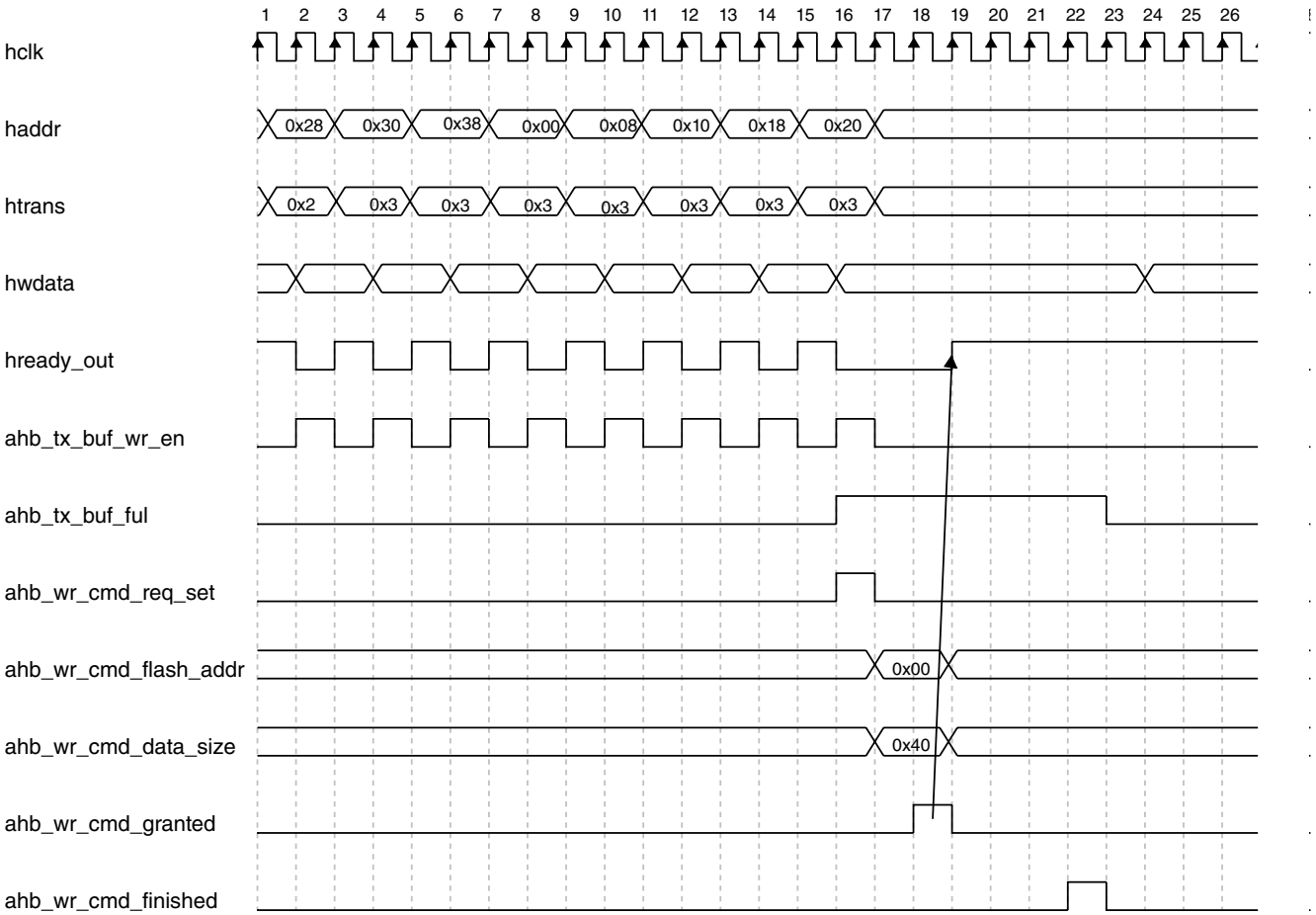


Figure 25-19. AHB write access (WRAP8/64bit/Bufferable)

- 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;

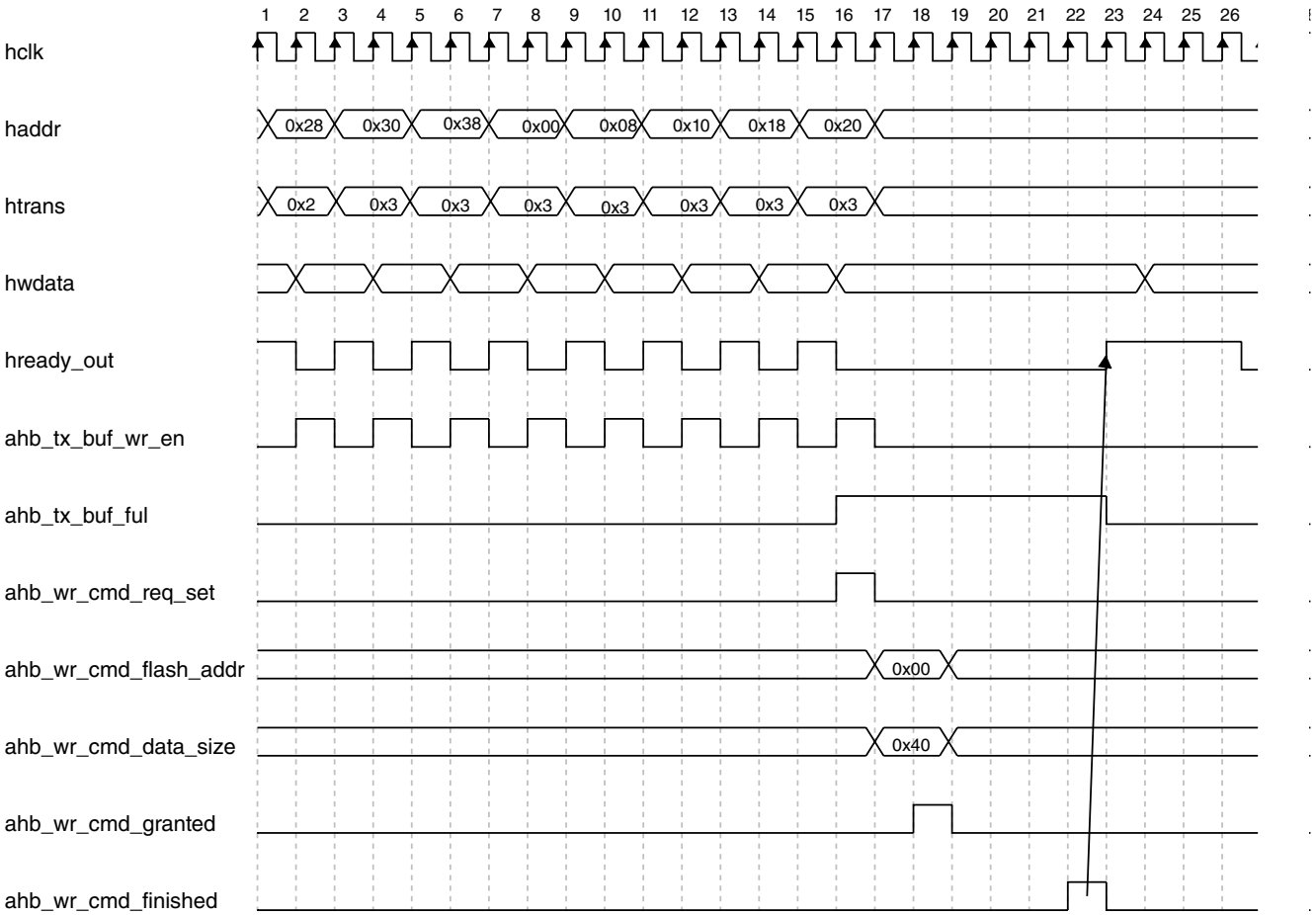


Figure 25-20. AHB write access (WRAP8/64bit/Non-Bufferable)

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.

25.5.11.2 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.

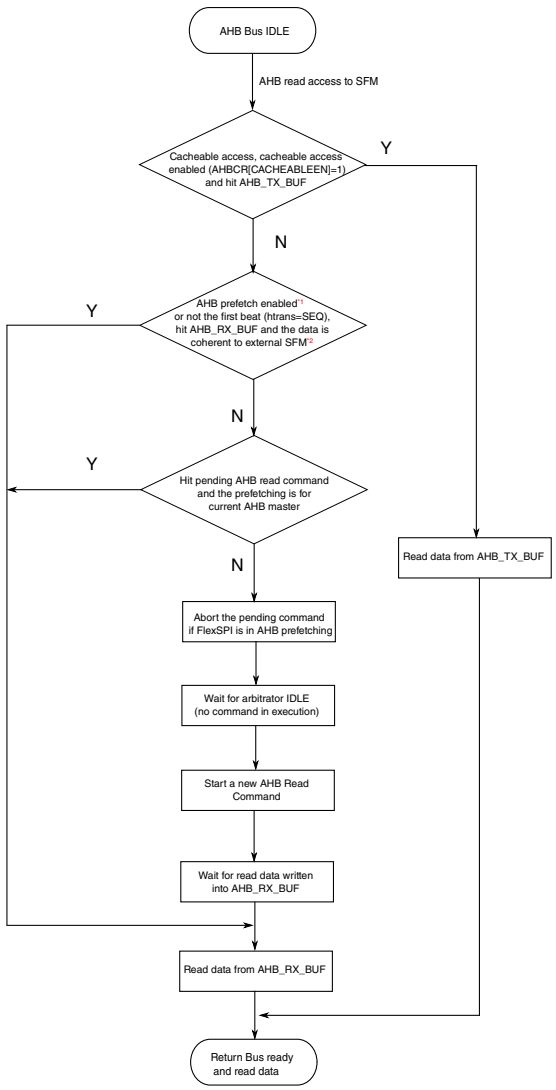


Figure 25-21. Hardware operation in response to AHB read access to Flash

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) or OTFAD is enabled.
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 25-8. 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. Refer MCR2[SAMEDEVICEEN] for more detail.
- 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, refer 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.

25.5.11.3 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.

AHB Rx Buffers total size is 1 KBytes . 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.
- 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. Refer Command Abort and Suspend.

25.5.12 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.

25.5.12.1 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).

Following is an example indicating command abort/suspend/resume flow:

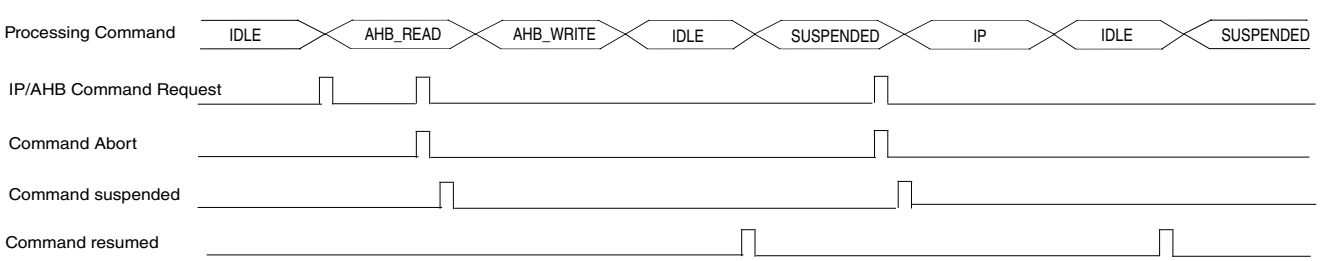


Figure 25-22. Command Instructions Execution on FlexSPI interface

25.5.13 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.

25.5.14 FlexSPI Output Timing

This section describes the output timing in FlexSPI.

25.5.14.1 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:

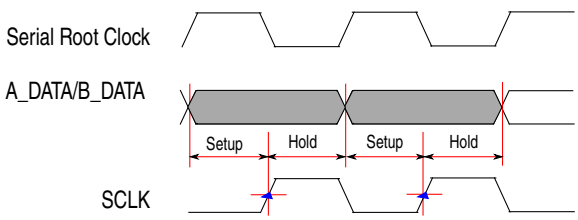


Figure 25-23. SDR instruction in SDR sequence

- **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:

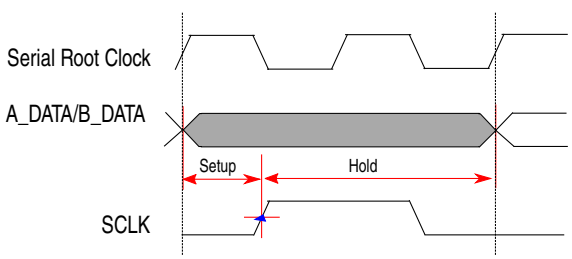


Figure 25-24. SDR instruction in DDR sequence

- **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:

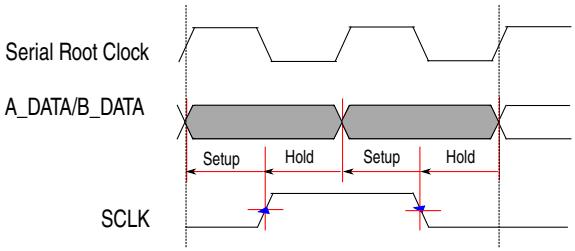


Figure 25-25. DDR instruction in DDR sequence

25.5.14.2 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:

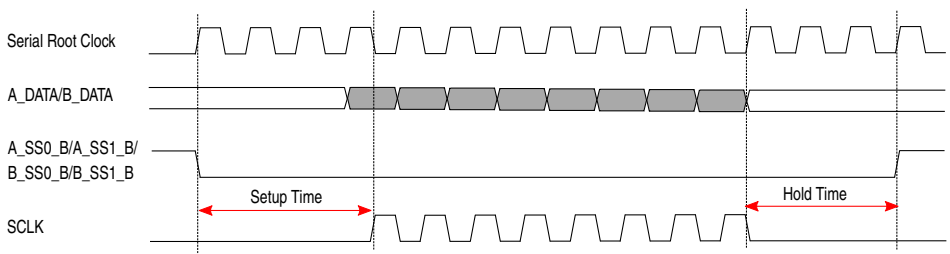


Figure 25-26. Chip selection output timing for SDR sequence

- **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:

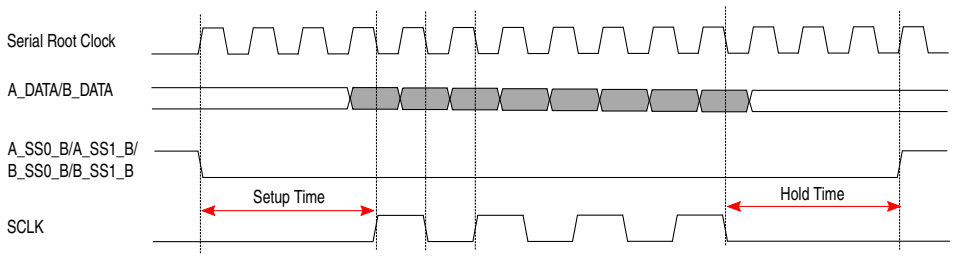


Figure 25-27. Chip selection output timing for DDR sequence

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.

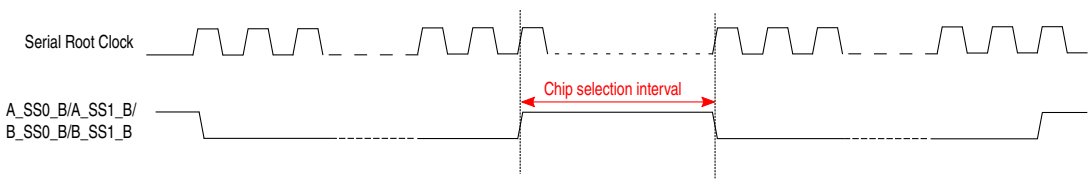


Figure 25-28. Chip Selection Valid interval

25.5.15 FlexSPI Input Timing

This section describes the input timing of FlexSPI.

25.5.15.1 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.

25.5.15.2 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

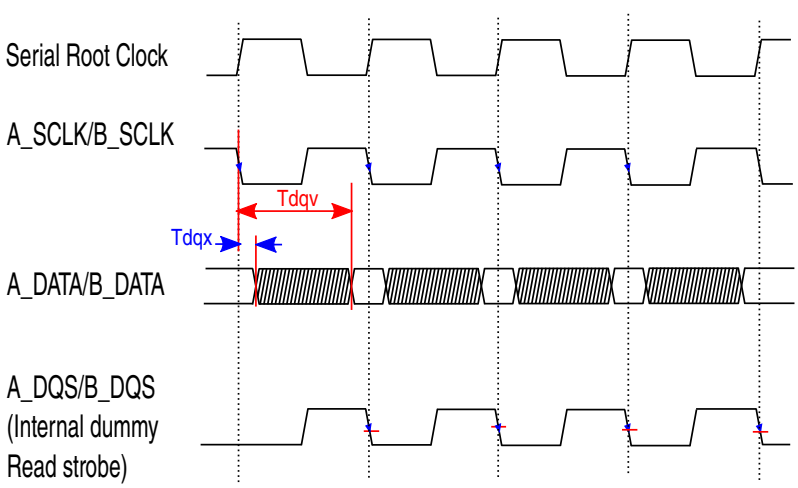


Figure 25-29. Input Timing for sampling with dummy read strobe in SDR mode

- **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

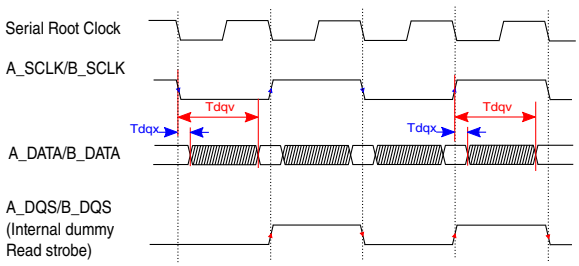


Figure 25-30. Input Timing for sampling with dummy read strobe in DDR mode

25.5.15.3 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:

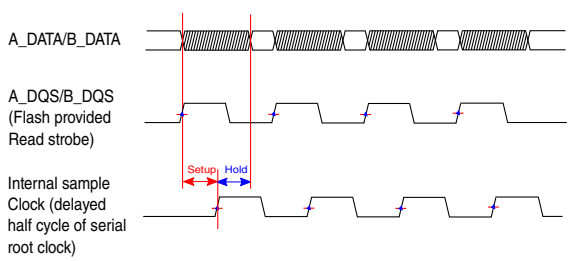


Figure 25-31. Input Timing 2 for Flash provided read strobe in SDR mode

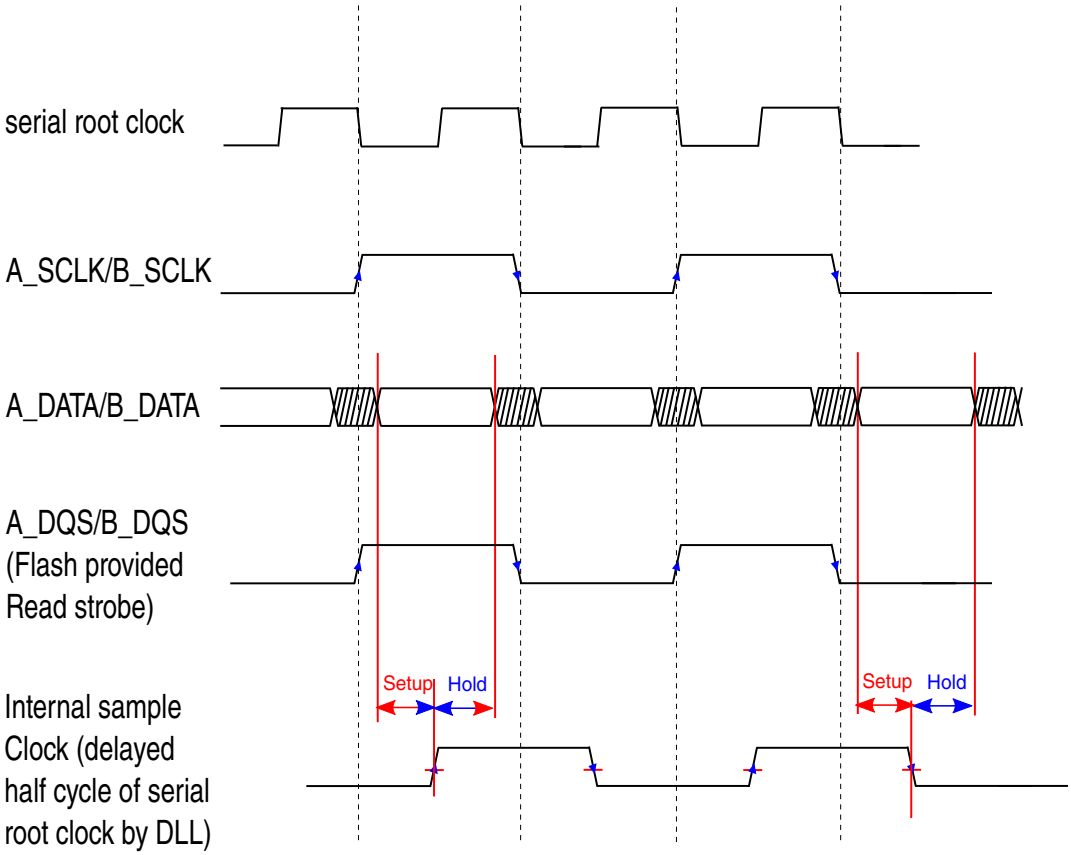


Figure 25-32. Input Timing 2 for Flash provided read strobe in DDR mode

25.5.15.4 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 = 17,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).

25.5.16 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.

Following diagram indicates XIP operation with Flash XIP Enhanced mode:

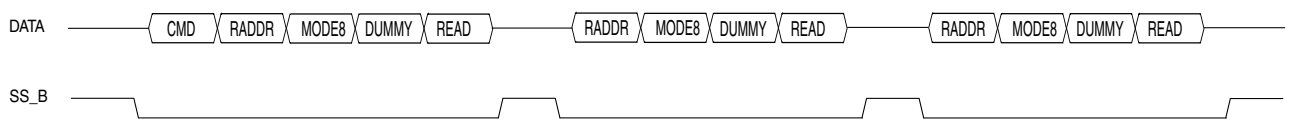


Figure 25-33. XIP Enhanced Mode operation

25.6 Application information

This section describes applications supported by the FlexSPI module.

25.6.1 FlexSPI Initialization

FlexSPI controller initialization sequence is as following:

- Enable controller clocks (AHB clock/IP Bus clock/Serial root clock) in System level.
- Power the FlexSPI SRAM using PDRUNCFG registers.
- Release FlexSPI from reset using PRSTCRL registers.
- 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.

25.6.2 Overview of Error Flags

The following table gives an overview of error category, flags and triggered source.

Table 25-9. 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.	INTR[IPCMDERR] will be set

Table continues on the next page...

Table 25-9. Error category and flags in FlexSPI (continued)

Error Category	Triggered Source	Description	Error Flags
		• Instruction DUMMY_SDR/ DUMMY_RWDS_SDR used in DDR sequence. • Instruction DUMMY_DDR/ DUMMY_RWDS_DDR used in SDR sequence. • Flash boundary across.	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 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

25.6.3 Application on Serial NOR Flash device

This section provides the example sequences for serial NOR flash device (Cypress Flash S25FS128S).

25.6.3.1 Write Enable command

The following table shows WRITE ENABLE command sequence.

Table 25-10. 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	

25.6.3.2 Write Registers command

The following table shows Write Registers command sequence.

Table 25-11. 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	

25.6.3.3 Page Program command

The following table shows Page Program command sequence.

Table 25-12. 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 25-13. 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	

25.6.3.4 Read Status 1 command

The following table shows READ STATUS 1 command sequence.

Table 25-14. 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	

25.6.3.5 Read command

The following table shows READ command sequence.

Table 25-15. 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
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	

25.6.3.6 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 25-16. 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	

25.6.3.7 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 25-17. 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 25-18. 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.

Table continues on the next page...

Table 25-18. Dual IO FAST_READ command (Non-continuous Read mode) (continued)

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
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	

25.6.3.8 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 25-19. 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 25-20. 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.
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 25-21. 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	

25.6.3.9 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 25-22. 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 25-23. 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.

Table continues on the next page...

Table 25-23. 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
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	

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 25-24. 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	

25.6.4 Application on HyperBus device

This section provides the example sequences for HyperBus device (Cypress RPC flash/ HyperRam/HyperFlash).

25.6.4.1 HyperFlash

This section provides the example sequences for HyperFlash devices (Cypress S26KS series).

The following table shows Read Status command sequence.

Table 25-25. Read Status command

Seq No.	Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0 (Write - Addr=0x555, Data=0x70)	0	CMD_DDR	0x3 (Octal)	0x20	CA bit 47: (R/W#) = 0x0 CA bit 46: (Target) = 0x0 CA bit 45: (Burst Type) = 0x1 CA bit 44-40: All reserved = 0x0
	1	CMD_DDR	0x3 (Octal)	0x00	Row Address: 0x0000AA (24 bit)
	2	CMD_DDR	0x3 (Octal)	0x00	
	3	CMD_DDR	0x3 (Octal)	0xAA	
	4	CMD_DDR	0x3 (Octal)	0x00	Column Address: 0x05 (13 zero bits + 3 valid bits)
	5	CMD_DDR	0x3 (Octal)	0x05	
	6	CMD_DDR	0x3 (Octal)	0x00	Write Data: 0x0070
	7	CMD_DDR	0x3 (Octal)	0x70	
1 (Read - Addr=xxx, Data= Status register data)	0	CMD_DDR	0x3 (Octal)	0xA0	CA bit 47: (R/W#) = 0x1 CA bit 46: (Target) = 0x0 CA bit 45: (Burst Type) = 0x1 CA bit 44-40: All reserved = 0x0
	1	RADDR_DDR	0x3 (Octal)	0x18	Row Address: 24 bits
	2	CADDR_DDR	0x3 (Octal)	0x10	Column Address: (13 zero bits + 3 valid bits)
	3	DUMMY_RWDS _DDR	0x3 (Octal)	0x0B	In case of latency count=11
	4	READ_DDR	0x3 (Octal)	0x4	4 Byte read
	5~7	STOP (0x00)	0x0	0x00	

The following table shows Read (memory) command sequence.

Table 25-26. Read (memory) command

Seq No.	Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0 (Read - Addr=xxx, Data= memory data)	0	CMD_DDR	0x3 (Octal)	0xA0	CA bit 47: (R/W#) = 0x1 CA bit 46: (Target) = 0x0 CA bit 45: (Burst Type) = 0x1 CA bit 44-40: All reserved = 0x0
	1	RADDR_DDR	0x3 (Octal)	0x18	Row Address: 24 bits
	2	CADDR_DDR	0x3 (Octal)	0x10	Column Address: (13 zero bits + 3 valid bits)
	3	DUMMY_RWDS_DDR	0x3 (Octal)	0x0B	In case of latency count=11
	4	READ_DDR	0x3 (Octal)	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 Word Program command sequence.

Table 25-27. Word Program command

Seq No.	Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0 (Write - Addr=0x555, Data=0xAA)	0	CMD_DDR	0x3 (Octal)	0x20	CA bit 47: (R/W#) = 0x0 CA bit 46: (Target) = 0x0 CA bit 45: (Burst Type) = 0x1 CA bit 44-40: All reserved = 0x0
	1	CMD_DDR	0x3 (Octal)	0x00	Row Address: 0x0000AA (24 bit)
	2	CMD_DDR	0x3 (Octal)	0x00	
	3	CMD_DDR	0x3 (Octal)	0xAA	
	4	CMD_DDR	0x3 (Octal)	0x00	Column Address: 0x05 (13 zero bits + 3 valid bits)
	5	CMD_DDR	0x3 (Octal)	0x05	
	6	CMD_DDR	0x3 (Octal)	0x00	Write Data: 0x00AA
	7	CMD_DDR	0x3 (Octal)	0xAA	
1 (Write - Addr=0x2AA, Data=0x55)	0	CMD_DDR	0x3 (Octal)	0x20	CA bit 47: (R/W#) = 0x0 CA bit 46: (Target) = 0x0 CA bit 45: (Burst Type) = 0x1 CA bit 44-40: All reserved = 0x0
	1	CMD_DDR	0x3 (Octal)	0x00	Row Address: 0x000055 (24 bit)
	2	CMD_DDR	0x3 (Octal)	0x00	
	3	CMD_DDR	0x3 (Octal)	0x55	
	4	CMD_DDR	0x3 (Octal)	0x00	Column Address: 0x02 (13 zero bits + 3 valid bits)

Table continues on the next page...

Table 25-27. Word Program command (continued)

Seq No.	Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
	5	CMD_DDR	0x3 (Octal)	0x02	
	6	CMD_DDR	0x3 (Octal)	0x00	Write Data: 0x0055
	7	CMD_DDR	0x3 (Octal)	0x55	
2 (Write - Addr=0x555, Data=0xA0)	0	CMD_DDR	0x3 (Octal)	0x20	CA bit 47: (R/W#) = 0x0 CA bit 46: (Target) = 0x0 CA bit 45: (Burst Type) = 0x1 CA bit 44-40: All reserved = 0x0
	1	CMD_DDR	0x3 (Octal)	0x00	Row Address: 0x0000AA (24 bit)
	2	CMD_DDR	0x3 (Octal)	0x00	
	3	CMD_DDR	0x3 (Octal)	0xAA	
	4	CMD_DDR	0x3 (Octal)	0x00	Column Address: 0x05 (13 zero bits + 3 valid bits)
	5	CMD_DDR	0x3 (Octal)	0x55	
	6	CMD_DDR	0x3 (Octal)	0x00	Write Data: 0x00A0
	7	CMD_DDR	0x3 (Octal)	0xA0	
3 (Word Program)	0	CMD_DDR	0x3 (Octal)	0x20	CA bit 47: (R/W#) = 0x0 CA bit 46: (Target) = 0x0 CA bit 45: (Burst Type) = 0x1 CA bit 44-40: All reserved = 0x0
	1	RADDR_DDR	0x3 (Octal)	0x18	Row Address: 24 bits
	2	CADDR_DDR	0x3 (Octal)	0x10	Column Address: (13 zero bits + 3 valid bits)
	3	WRITE_DDR	0x3 (Octal)	0x02	2 Byte written data
	4-7	STOP (0x0)	0x0	0x00	

The following table shows Written-to-Buffer and Program-Buffer-to-Flash command sequence.

Table 25-28. Written-to-Buffer and Program-Buffer-to-Flash command

Seq No.	Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0 (Write - Addr=0x555, Data=0xAA)	0	CMD_DDR	0x3 (Octal)	0x20	CA bit 47: (R/W#) = 0x0 CA bit 46: (Target) = 0x0 CA bit 45: (Burst Type) = 0x1 CA bit 44-40: All reserved = 0x0
	1	CMD_DDR	0x3 (Octal)	0x00	Row Address: 0x0000AA (24 bit)
	2	CMD_DDR	0x3 (Octal)	0x00	
	3	CMD_DDR	0x3 (Octal)	0xAA	

Table continues on the next page...

Table 25-28. Written-to-Buffer and Program-Buffer-to-Flash command (continued)

Seq No.	Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
	4	CMD_DDR	0x3 (Octal)	0x00	Column Address: 0x05 (13 zero bits + 3 valid bits)
	5	CMD_DDR	0x3 (Octal)	0x05	
	6	CMD_DDR	0x3 (Octal)	0x00	Write Data: 0x00AA
	7	CMD_DDR	0x3 (Octal)	0xAA	
1 (Write - Addr=0x2AA, Data=0x55)	0	CMD_DDR	0x3 (Octal)	0x20	CA bit 47: (R/W#) = 0x0 CA bit 46: (Target) = 0x0 CA bit 45: (Burst Type) = 0x1 CA bit 44-40: All reserved = 0x0
	1	CMD_DDR	0x3 (Octal)	0x00	Row Address: 0x000055 (24 bit)
	2	CMD_DDR	0x3 (Octal)	0x00	
	3	CMD_DDR	0x3 (Octal)	0x55	
	4	CMD_DDR	0x3 (Octal)	0x00	Column Address: 0x02 (13 zero bits + 3 valid bits)
	5	CMD_DDR	0x3 (Octal)	0x02	
	6	CMD_DDR	0x3 (Octal)	0x00	Write Data: 0x0055
	7	CMD_DDR	0x3 (Octal)	0x55	
2 (Write - Addr=SA, Data=0x25)	0	CMD_DDR	0x3 (Octal)	0x20	CA bit 47: (R/W#) = 0x0 CA bit 46: (Target) = 0x0 CA bit 45: (Burst Type) = 0x1 CA bit 44-40: All reserved = 0x0
	1	RADDR_DDR	0x3 (Octal)	0x18	Row Address: SA (24 bit) SA is sector address. Please set IPCR0[SFAR]=SA
	2	CADDR_DDR	0x3 (Octal)	0x10	Column Address: 13 zero bits + 3 valid bits
	3	CMD_DDR	0x3 (Octal)	0x00	Write Data: 0x0025
	4	CMD_DDR	0x3 (Octal)	0x25	
2 (Write - Addr=SA, Data=WC)	0	CMD_DDR	0x3 (Octal)	0x20	CA bit 47: (R/W#) = 0x0 CA bit 46: (Target) = 0x0 CA bit 45: (Burst Type) = 0x1 CA bit 44-40: All reserved = 0x0
	1	RADDR_DDR	0x3 (Octal)	0x18	Row Address: SA (24 bit) SA is sector address. Please set IPCR0[SFAR]=SA
	2	CADDR_DDR	0x3 (Octal)	0x10	Column Address: 13 zero bits + 3 valid bits
	3	CMD_DDR	0x3 (Octal)	WC	Write Data: WC
	4	CMD_DDR	0x3 (Octal)		WC is word count
3 - N	0	CMD_DDR	0x3 (Octal)	0x20	CA bit 47: (R/W#) = 0x0 CA bit 46: (Target) = 0x0

Table continues on the next page...

Table 25-28. Written-to-Buffer and Program-Buffer-to-Flash command (continued)

Seq No.	Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
(Write - Addr=WBL, Data=PD) N is the word count + 2					CA bit 45: (Burst Type) = 0x1 CA bit 44-40: All reserved = 0x0
	1	RADDR_DDR	0x3 (Octal)	0x18	Row Address: WBL (24 bit) WBL is write buffer location. Please set IPCR0[SFAR]=WBL
	2	CADDR_DDR	0x3 (Octal)	0x10	Column Address: 13 zero bits + 3 valid bits
	3	WRITE_DDR	0x3 (Octal)	0x02	2 Byte write data
	4-7	STOP (0x0)	0x0	0x00	
N+1 (Write - Addr=SA, Data=29) Program Buffer to Flash	0	CMD_DDR	0x3 (Octal)	0x20	CA bit 47: (R/W#) = 0x0 CA bit 46: (Target) = 0x0 CA bit 45: (Burst Type) = 0x1 CA bit 44-40: All reserved = 0x0
	1	RADDR_DDR	0x3 (Octal)	0x18	Row Address: SA (24 bit) SA is sector address. Please set IPCR0[SFAR]=SA
	2	CADDR_DDR	0x3 (Octal)	0x10	Column Address: 13 zero bits + 3 valid bits
	3	CMD_DDR	0x3 (Octal)	0x00	Write Data: 0x29
	4	CMD_DDR	0x3 (Octal)	0x29	
	5-7	STOP (0x0)	0x0	0x00	

25.6.4.2 HyperRAM

This section provides the example sequences for HyperRAM (Cypress S27KL series).

Read (memory) command sequence is same as HyperFlash. The following table shows Write (memory) command sequence.

Table 25-29. Write (memory) command

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_DDR	0x3 (Octal)	0x20	CA bit 47: (R/W#) = 0x0 CA bit 46: (Target) = 0x0 CA bit 45: (Burst Type) = 0x1 CA bit 44-40: All reserved = 0x0
1	RADDR_DDR	0x3 (Octal)	0x18	Row Address: 24 bits

Table continues on the next page...

Table 25-29. Write (memory) command (continued)

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
2	CADDR_DDR	0x3 (Octal)	0x10	Column Address: (13 zero bits + 3 valid bits)
3	DUMMY_RWDS_DDR	0x3 (Octal)	0x0B	In case of latency count=11
4	WRITE_DDR	0x3 (Octal)	Any non-zero value	This operand value could be used as default write data size if IPCR1[IDATSZ] is zero. This value is ignored for AHB command.
5~7	STOP (0x00)	0x0	0x00	

25.6.5 Application on Serial NAND Flash device

This section provides the example sequences for serial NAND flash device (Micron Flash MT29 series). The operation to serial NAND flash is quite similar to serial NOR flash.

READ operation sequence is as following:

- Page Read (Transfer the data from the NAND Flash array to the cache register)
- Get Feature to read the status
- Random Data Read

The following table shows Page Read command sequence.

Table 25-30. Page Read command

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0x1 (Single)	0x13	Command code: 0x13
1	RADDR_SDR	0x1 (Single)	0x18	Row Address: 24 bit
2-7	STOP (0x0)	0x0	0x00	

The following table shows Get Feature command sequence.

Table 25-31. Get Feature command

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0x1 (Single)	0x0F	Command code: 0x0F
1	CMD_SDR	0x1 (Single)	0xC0	Status register address (0xC0)
2	READ_SDR	0x1 (Single)	0x02	2 Byte read data
3-7	STOP (0x0)	0x0	0x00	

The following table shows Random Data Read command sequence.

Table 25-32. Read From Cache x4 command

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0x1 (Single)	0x6B	Command code: 0x6B
1	MODE4_SDR	0x1 (Single)	0x0 or 0x1	Software should decode the flash address and set mode bits as 0x1 if plane selection is one, or 0x0 if plane selection is zero. Plane selection bit is the 18th bit of flash address. If NAND flash size is less than 4Gbit, plane selection will always be zero.
2	CADDR_SDR	0x1 (Single)	0x0C	Column address: 12 bit
3	DUMMY_SDR	0x2 (Quad)	0x08	Dummy cycle number: 8 (serial root clock)
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.
5-7	STOP (0x0)	0x0	0x00	

Program operation sequence is as following:

- Write Enable
- Program Load (Transfer the write data to the cache register)
- Program Execute
- Get Feature to read the status

The following table shows Program Load command sequence.

Table 25-33. Program Load command

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0x1 (Single)	0x02	Command code: 0x02
1	MODE4_SDR	0x1 (Single)	0x0 or 0x1	Software should decode the flash address and set mode bits as 0x1 if plane selection is one, or 0x0 if plane selection is zero. Plane selection bit is the 18th bit of flash address. If NAND flash size is less than 4Gbit, plane selection will always be zero.
2	CADDR_SDR	0x1 (Single)	0x0C	Column address: 12 bit
3	WRITE_SDR	0x1 (Single)	Any non-zero value	This operand value could be used as default write data size if IPCR1[IDATSZ] is zero.
4-7	STOP (0x0)	0x0	0x00	

The following table shows Program Execute command sequence.

Table 25-34. Program Execute command

Instruction No.	Instruction opcode[5:0]	Instruction num_pads[1:0]	Instruction operand[7:0]	Comment
0	CMD_SDR	0x1 (Single)	0x10	Command code: 0x10
1	RADDR_SDR	0x1 (Single)	0x18	Row Address: 24 bit
2-7	STOP (0x0)	0x0	0x00	

25.6.6 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.

25.7 Memory Map and register definition

This section includes the FlexSPI module memory map and detailed descriptions of all registers.

25.7.1 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.

NOTE

For usage that FlexSPI is capable of accessing sensitive memory contents, protection should be done to FlexSPI controller using resource isolation (e.g. XRDC2, XRDC, RDC) or any kind of equivalent partition control via SCFW, to make sure only trusted software be allowed to access the FlexSPI controller register interface.

25.7.2 FlexSPI register descriptions

25.7.2.1 FlexSPI memory map

FlexSPI base address: 400A_0000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Module Control Register 0 (MCR0)	32	RW	FFFF_80C2h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
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	RW	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
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 25-34

25.7.2.2 Module Control Register 0 (MCR0)

25.7.2.2.1 Offset

Register	Offset
MCR0	0h

25.7.2.2.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	AHBGRANTWAIT								IPGRANTWAIT							
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	Reserved	SCKFREERUNEN	COMBINATIONEN	DOZEEN	HSEN	SERCLKDIV			ATDFEN	ARDFEN	RXCLKSRC		Reserved		MDIS	SWRESET
Reset	1	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0

25.7.2.2.3 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

Table continues on the next page...

Field	Function
	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]). **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 . Refer Clocks chapter 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	Sample Clock source selection for Flash Reading

Table continues on the next page...

Field	Function
RXCLKSRC	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.

25.7.2.3 Module Control Register 1 (MCR1)

25.7.2.3.1 Offset

Register	Offset
MCR1	4h

25.7.2.3.2 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

25.7.2.3.3 Fields

Field	Function
31-16	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

Table continues on the next page...

Field	Function
SEQWAIT	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[AHBBUSERROREN]) if this interrupt is enabled (INTR[AHBBUSERROREN]) is set 0x1) and AHB command is ignored by arbitrator. **NOTE:** It is not allowed to set this field to value 0x0.

25.7.2.4 Module Control Register 2 (MCR2)

25.7.2.4.1 Offset

Register	Offset
MCR2	8h

25.7.2.4.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	RESUMEWAIT								Reserved				SCKBDIFFOPT	Reserved		Reserved
Reset	0	0	1	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	SAMEDEVICEEN	CLRLEARNPHASE	Reserved	Reserved	CLRAHBBUFOPT	Reserved		Reserved								
Reset	1	0	0	0	0	0	0	1	1	1	1	1	0	1	1	1

25.7.2.4.3 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 SAMEDEVICEEN	All external devices are same devices (both in types and size) for A1/A2/B1/B2. 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 CLRLEARNPHASE	The sampling clock phase selection will be reset to phase 0 when this bit is written with 0x1. This bit will be auto-cleared immediately.
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

25.7.2.5 AHB Bus Control Register (AHBCR)

25.7.2.5.1 Offset

Register	Offset
AHBCR	Ch

25.7.2.5.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	Reserved			Reserved										Reserved	Reserved	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	Reserved		Reserved	Reserved	READSZALIGN	Reserved	Reserved	Reserved	READADDROPT	PREFETCHEN	BUFFERABLEEN	CACHABLEEN	CLRAHBTXBUF	CLRAHBRXBUF	APAREN
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0

25.7.2.5.3 Fields

Field	Function
31-29 —	Reserved
28-19 —	Reserved
18 —	Reserved
17 —	Reserved
16 —	Reserved
15 —	Reserved
14-13 —	Reserved
12	Reserved

Table continues on the next page...

Field	Function
—	
11 —	Reserved
10 READSZALIGN	AHB Read Size Alignment **NOTE:** When this bit is set to 1, data prefetching will be disabled regardless any other setting(PREFETCH_EN,OTFAD_EN...). And read size will be up sized to 8 bytes aligned. For example, if current AHB read size is 12 bytes, it will be aligned to 16 bytes. This bit is expected to be used when fetch size need to be 8 bytes aligned but no speculative fetching is wanted. 0b - AHB read size will be decided by other register setting like PREFETCH_EN,OTFAD_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. **NOTE:** When OTFAD is enabled, FlexSPI will work as READADDROPT set as 1 regardless the value of this bit. 0b - There is AHB read burst start address alignment limitation when flash is accessed in parallel mode or flash is wordaddressable. 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. **NOTE:** When OTFAD enabled, AHB prefetch function will be enabled regardless of the value of this field.
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 CLRAHBTXBUF	Clear the status/pointers of AHB TX Buffer. Auto-cleared.
1 CLRAHBRXBUF	Clear the status/pointers of AHB RX Buffer. Auto-cleared.

Table continues on the next page...

Field	Function
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.

25.7.2.6 Interrupt Enable Register (INTEN)

25.7.2.6.1 Offset

Register	Offset
INTEN	10h

25.7.2.6.2 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		KEYERRORE N	KEYDONEEN	SEQTIMEOUTEN	AHBBUSERRORE N	SCKSTOPBYWREN	SCKSTOPBYRDEN	Reserved	IPTXWEE N	IPRXWAEN	AHBCMDERREN	IPCMDERREN	AHBCMDGEEN	IPCMDGEEN	IPCMDDONEEN
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

25.7.2.6.3 Fields

Field	Function
31-16 —	Reserved
15-14 —	Reserved
13 KEYERROREN	OTFAD key blob processing error interrupt enable.Refer Interrupts chapter for more details.

Table continues on the next page...

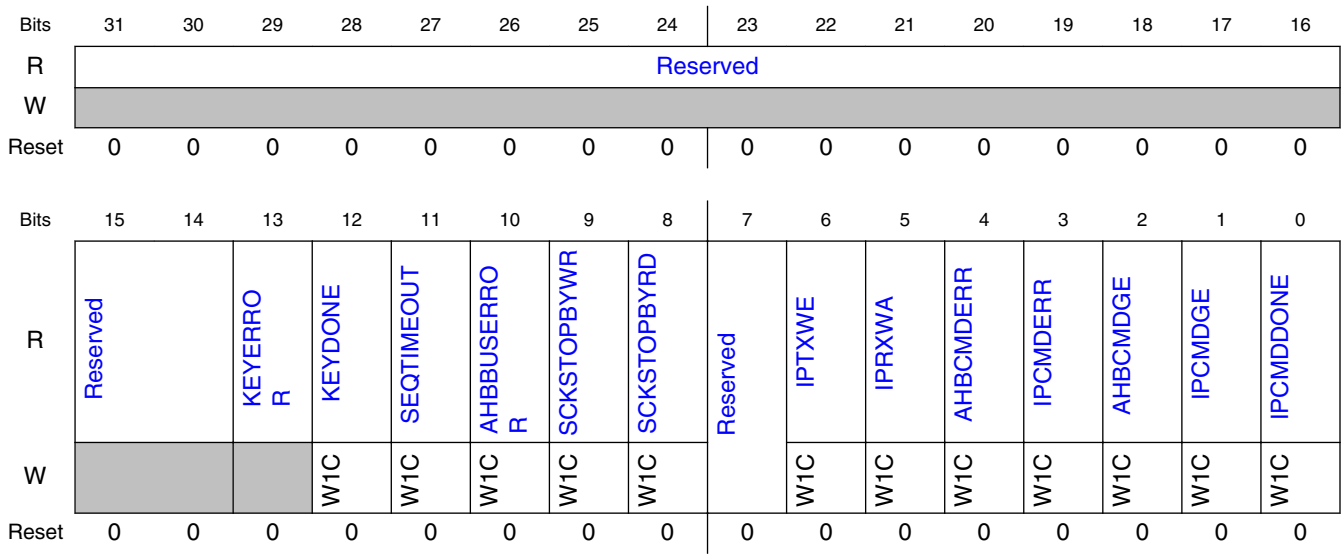
Field	Function
12 KEYDONEEN	OTFAD key blob processing done interrupt enable.Refer Interrupts chapter for more details.
11 SEQTIMEOUTEN	Sequence execution timeout interrupt enable.Refer Interrupts chapter for more details.
10 AHBBUSERROREN	AHB Bus error interrupt enable.Refer Interrupts chapter for more details.
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.

25.7.2.7 Interrupt Register (INTR)

25.7.2.7.1 Offset

Register	Offset
INTR	14h

25.7.2.7.2 Diagram



25.7.2.7.3 Fields

Field	Function
31-16 —	Reserved
15-14 —	Reserved
13 KEYERROR	OTFAD key blob processing error interrupt.
12 KEYDONE	OTFAD key blob processing done interrupt.
11 SEQTIMEOUT	Sequence execution timeout interrupt.
10 AHBBUSERROR	AHB Bus timeout or AHB bus illegal access Flash during OTFAD key blob processing interrupt.
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.
7 —	Reserved

Table continues on the next page...

Field	Function
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.

25.7.2.8 LUT Key Register (LUTKEY)

25.7.2.8.1 Offset

Register	Offset
LUTKEY	18h

25.7.2.8.2 Function

The LUT Key Register contains the key to lock and unlock LUT. Refer to Look Up Table for details.

25.7.2.8.3 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

25.7.2.8.4 Fields

Field	Function
31-0 KEY	The Key to lock or unlock LUT. The key is 0x5AF05AF0. Read value is always 0x5AF05AF0.

25.7.2.9 LUT Control Register (LUTCR)

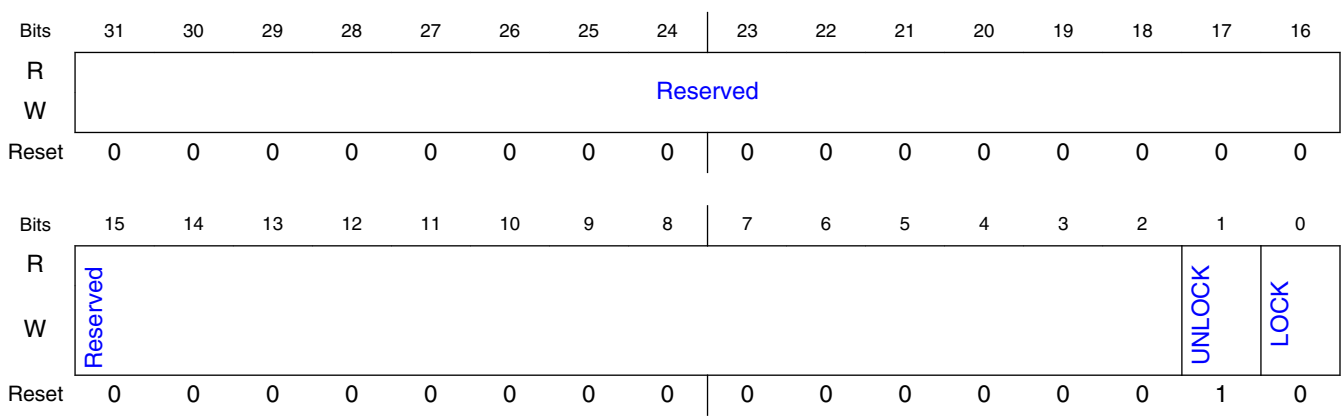
25.7.2.9.1 Offset

Register	Offset
LUTCR	1Ch

25.7.2.9.2 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.

25.7.2.9.3 Diagram



25.7.2.9.4 Fields

Field	Function
31-2 —	Reserved
1 UNLOCK	Unlock LUT
0 LOCK	Lock LUT

25.7.2.10 AHB RX Buffer 0 Control Register 0 (AHBRXBUF0CR0)

25.7.2.10.1 Offset

Register	Offset
AHBRXBUF0CR0	20h

25.7.2.10.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PREFETCHEN	Reserved					PRIORITY		Reserved				MSTRID			
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	Reserved								BUFSZ							
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

25.7.2.10.3 Fields

Field	Function
31 PREFETCHEN	AHB Read Prefetch Enable for current AHB RX Buffer corresponding Master.

Table continues on the next page...

Field	Function
	The prefetch feature is disabled when AHBCR[PREFETCHEN] is set 0. This field allows prefetch disable/ enable separately for each master.
30-26 —	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.

25.7.2.11 AHB RX Buffer 1 Control Register 0 (AHBRXBUF1CR0)

25.7.2.11.1 Offset

Register	Offset
AHBRXBUF1CR0	24h

25.7.2.11.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PREFETCHEN	Reserved					PRIORITY		Reserved				MSTRID			
Reset	1	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 W	Reserved								BUFSZ							
Reset	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

25.7.2.11.3 Fields

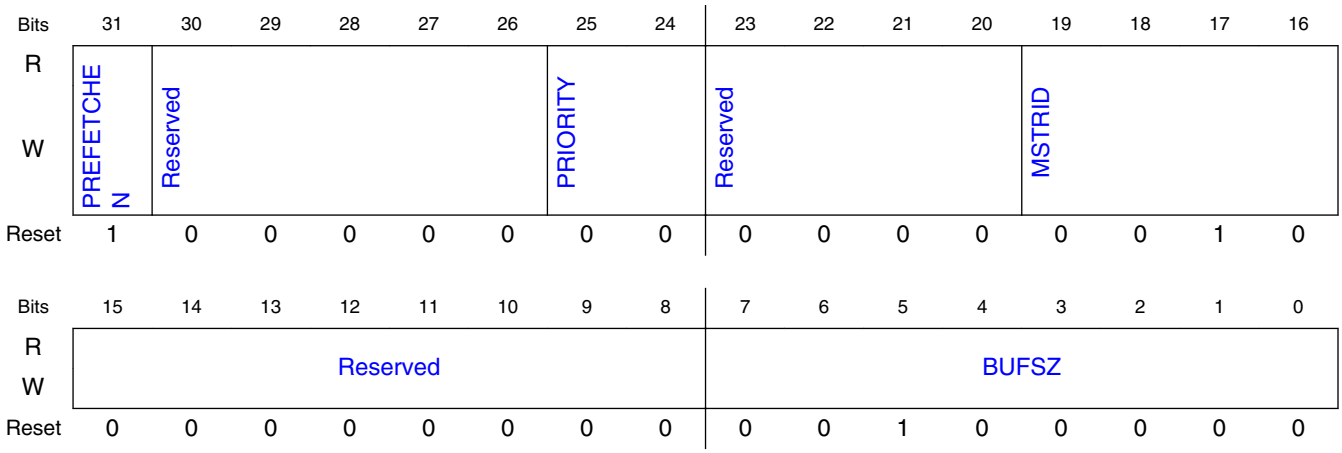
Field	Function
31 PREFETCHEN	AHB Read Prefetch Enable for current AHB RX Buffer corresponding Master. The prefetch feature is disabled when AHBCR[PREFETCHEN] is set 0. This field allows prefetch disable/enable separately for each master.
30-26 —	Reserved
25-24 PRIORITY	This priority for AHB Master Read 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.

25.7.2.12 AHB RX Buffer 2 Control Register 0 (AHBRXBUF2CR0)

25.7.2.12.1 Offset

Register	Offset
AHBRXBUF2CR0	28h

25.7.2.12.2 Diagram



25.7.2.12.3 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.
30-26 —	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.

25.7.2.13 AHB RX Buffer 3 Control Register 0 (AHBRXBUF3CR0)

25.7.2.13.1 Offset

Register	Offset
AHBRXBUF3CR0	2Ch

25.7.2.13.2 Diagram

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

25.7.2.13.3 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.
30-26 —	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.

25.7.2.14 Flash Control Register 0 (FLSHA1CR0 - FLSHB2CR0)

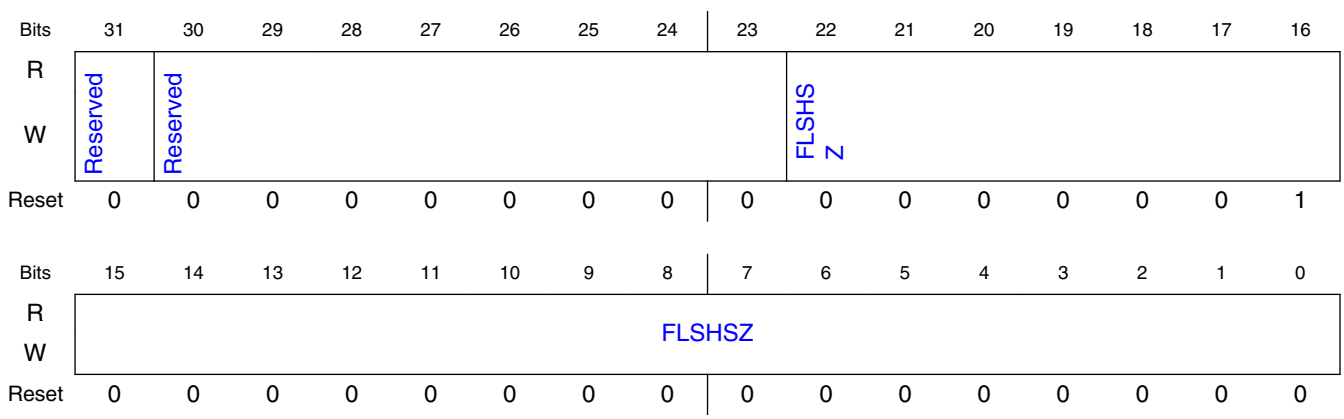
25.7.2.14.1 Offset

Register	Offset
FLSHA1CR0	60h
FLSHA2CR0	64h
FLSHB1CR0	68h
FLSHB2CR0	6Ch

25.7.2.14.2 Function

The Flash control register 0 contains Flash size setting. FlexSPI determines which device is accessed with this register setting (Chip Selection).

25.7.2.14.3 Diagram



25.7.2.14.4 Fields

Field	Function
31 —	Reserved
30-23 —	Reserved
22-0 FLSHSZ	Flash Size in KByte.

Field	Function
	The max flash size supported for each device is 0 GB. When FLSHSZ setting value is greater than 0x400000, the device flash size would be taken as 0 GB. The max total flash size supported (for all 4 devices) is also 0 GB. If the total flash size is larger than 0 GB, only 0 GB address space is accessible.

25.7.2.15 Flash Control Register 1 (FLSHA1CR1 - FLSHB2CR1)

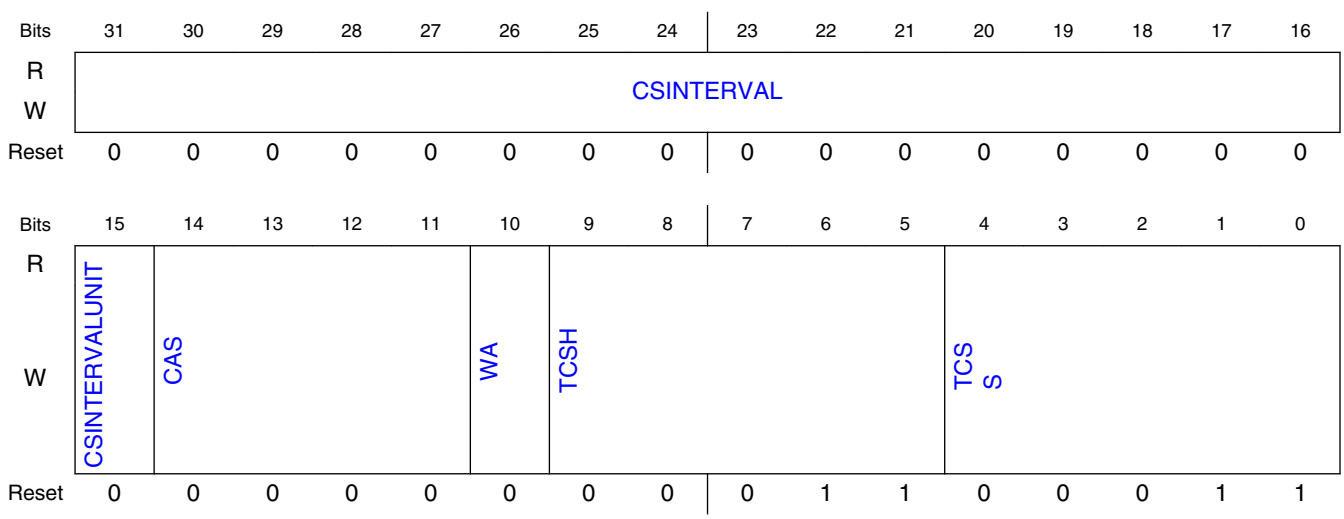
25.7.2.15.1 Offset

Register	Offset
FLSHA1CR1	70h
FLSHA2CR1	74h
FLSHB1CR1	78h
FLSHB2CR1	7Ch

25.7.2.15.2 Function

The Flash control register 1 contains the setting for FlexSPI to meet Flash device specific timings and Flash internal address space.

25.7.2.15.3 Diagram



25.7.2.15.4 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 TCSH	Serial Flash CS Hold time. 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.

25.7.2.16 Flash Control Register 2 (FLSHA1CR2 - FLSHB2CR2)

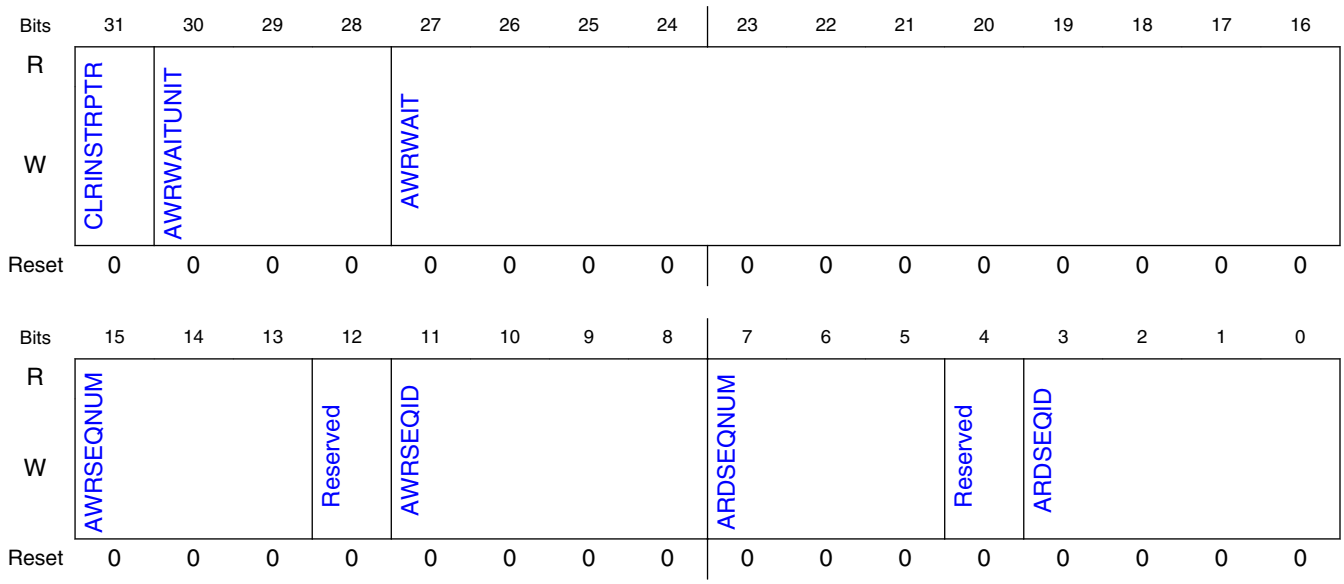
25.7.2.16.1 Offset

Register	Offset
FLSHA1CR2	80h
FLSHA2CR2	84h
FLSHB1CR2	88h
FLSHB2CR2	8Ch

25.7.2.16.2 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.

25.7.2.16.3 Diagram



25.7.2.16.4 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

Table continues on the next page...

Field	Function
15-13 AWRSEQNUM	Sequence Number for AHB Write triggered Command. For certain flash devices (E.g. HyperFlash/HyperRam/Serial NAND flash), 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. **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 (E.g. HyperFlash/HyperRam/Serial NAND flash), 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.

25.7.2.17 Flash Control Register 4 (FLSHCR4)

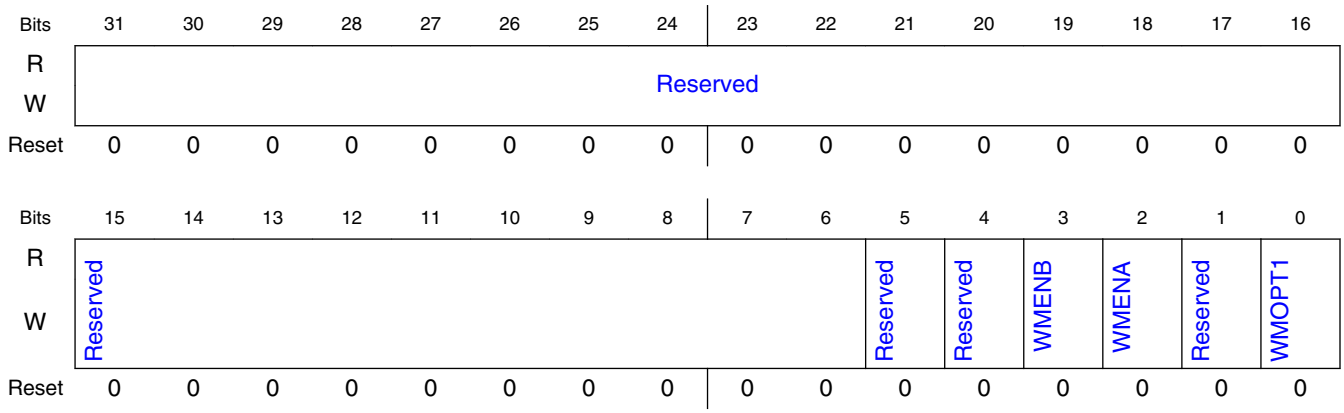
25.7.2.17.1 Offset

Register	Offset
FLSHCR4	94h

25.7.2.17.2 Function

The flash control register 4 provide the configuration for all external devices.

25.7.2.17.3 Diagram



25.7.2.17.4 Fields

Field	Function
31-6 —	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.

25.7.2.18 IP Control Register 0 (IPCR0)

25.7.2.18.1 Offset

Register	Offset
IPCR0	A0h

25.7.2.18.2 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.

25.7.2.18.3 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

25.7.2.18.4 Fields

Field	Function
31-0 SFAR	Serial Flash Address for IP command.

25.7.2.19 IP Control Register 1 (IPCR1)

25.7.2.19.1 Offset

Register	Offset
IPCR1	A4h

25.7.2.19.2 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.

25.7.2.19.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	IPAREN	Reserved				ISEQNUM			Reserved				ISEQID			
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	IDATSZ															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

25.7.2.19.4 Fields

Field	Function
31 IPAREN	Parallel mode Enabled for IP command. 0b - Flash will be accessed in Individual mode. 1b - Flash will be accessed in Parallel mode.
30-27	Reserved

Table continues on the next page...

Field	Function
—	
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.

25.7.2.20 IP Command Register (IPCMD)

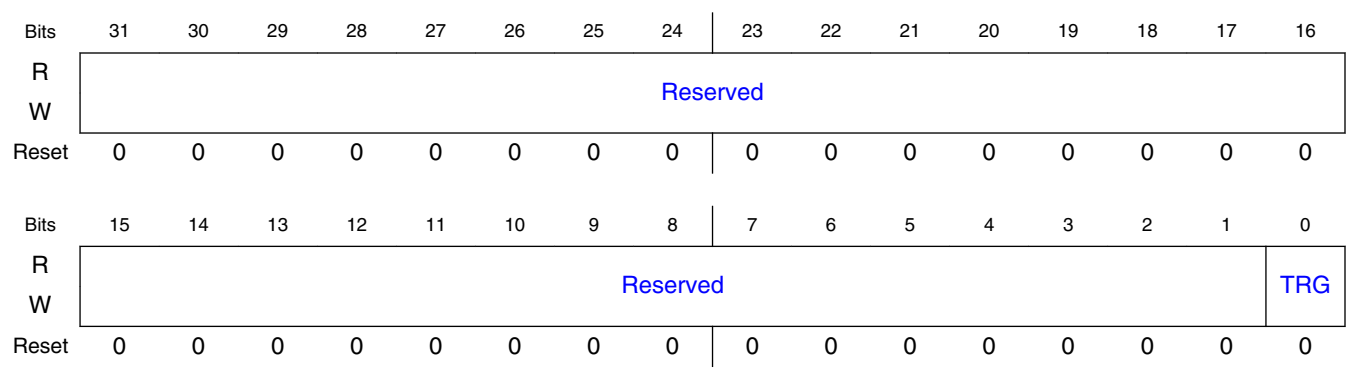
25.7.2.20.1 Offset

Register	Offset
IPCMD	B0h

25.7.2.20.2 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.

25.7.2.20.3 Diagram



25.7.2.20.4 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.

25.7.2.21 IP RX FIFO Control Register (IPRXFCR)

25.7.2.21.1 Offset

Register	Offset
IPRXFCR	B8h

25.7.2.21.2 Function

This register provides the configuration fields for IP RX FIFO management.

25.7.2.21.3 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

25.7.2.21.4 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.

25.7.2.22 IP TX FIFO Control Register (IPTXFCR)

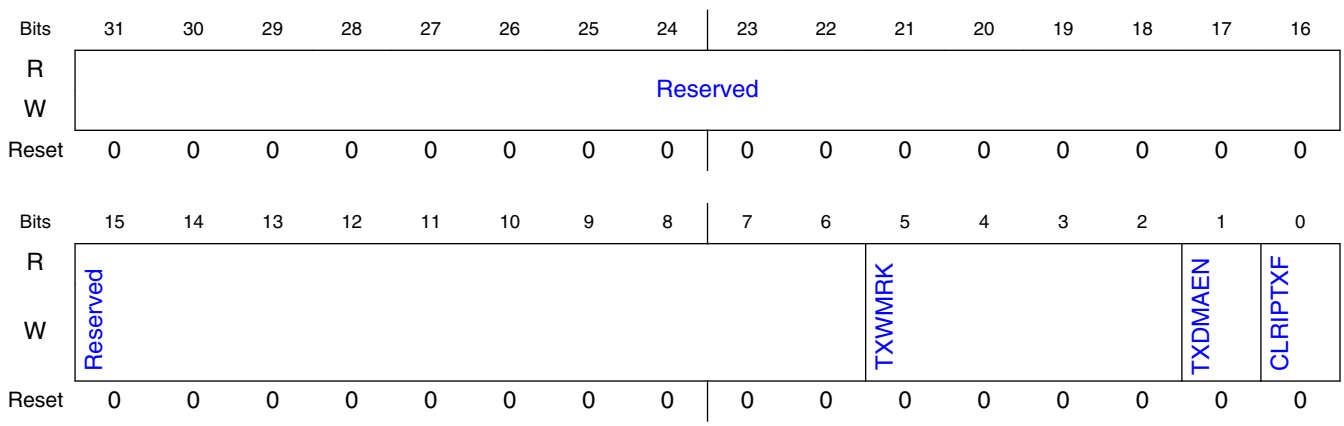
25.7.2.22.1 Offset

Register	Offset
IPTXFCR	BCh

25.7.2.22.2 Function

This register provides the configuration fields for IP TX FIFO management.

25.7.2.22.3 Diagram



25.7.2.22.4 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.

25.7.2.23 DLL Control Register 0 (DLLACR - DLLBCR)

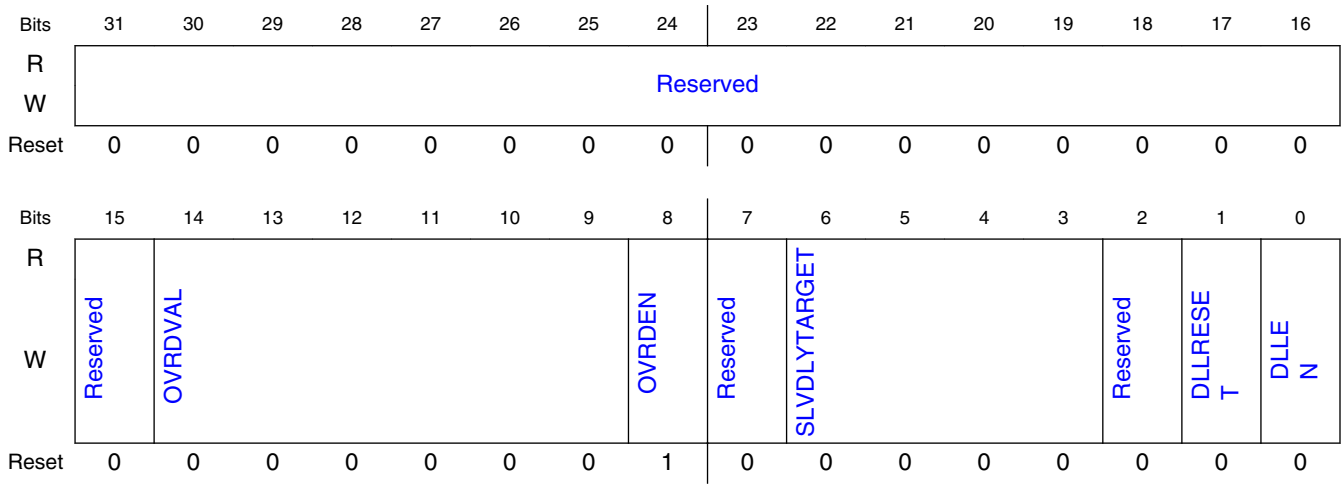
25.7.2.23.1 Offset

Register	Offset
DLLACR	C0h
DLLBCR	C4h

25.7.2.23.2 Function

This register provides the configuration fields for Flash A/B sample clock DLL.

25.7.2.23.3 Diagram



25.7.2.23.4 Fields

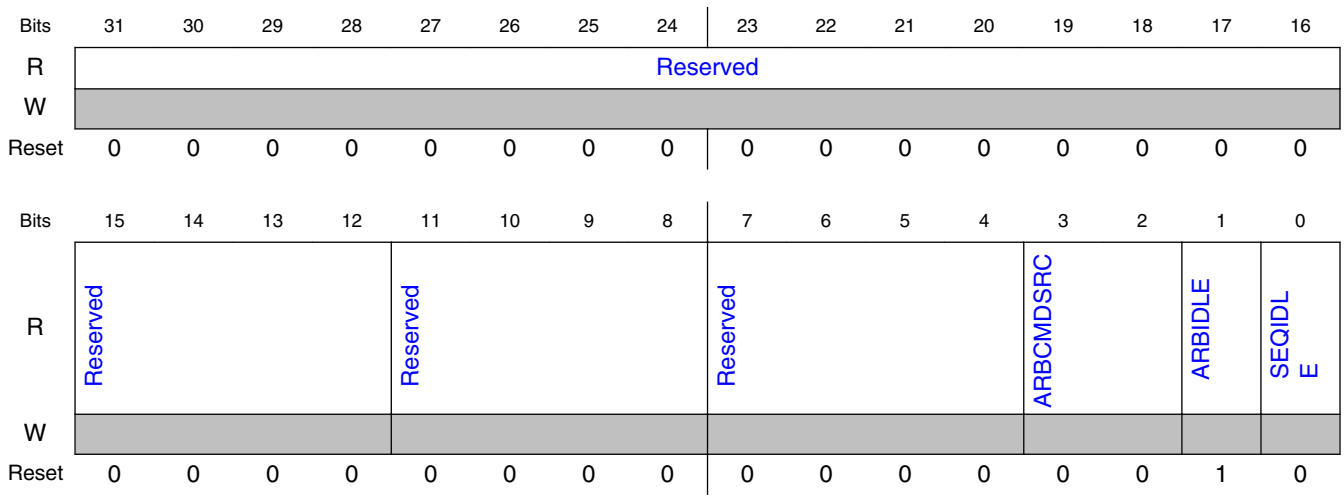
Field	Function
31-15 —	Reserved
14-9 OVRDVAL	Slave clock delay line delay cell number selection override value. When OVRDEN is set 0x1, the delay cell number in DLL is OVRDVAL+1.
8 OVRDEN	Slave clock delay line delay cell number selection override enable.
7 —	Reserved
6-3 SLVDLYTARGET	The delay target 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.

25.7.2.24 Status Register 0 (STS0)

25.7.2.24.1 Offset

Register	Offset
STS0	E0h

25.7.2.24.2 Diagram



25.7.2.24.3 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...

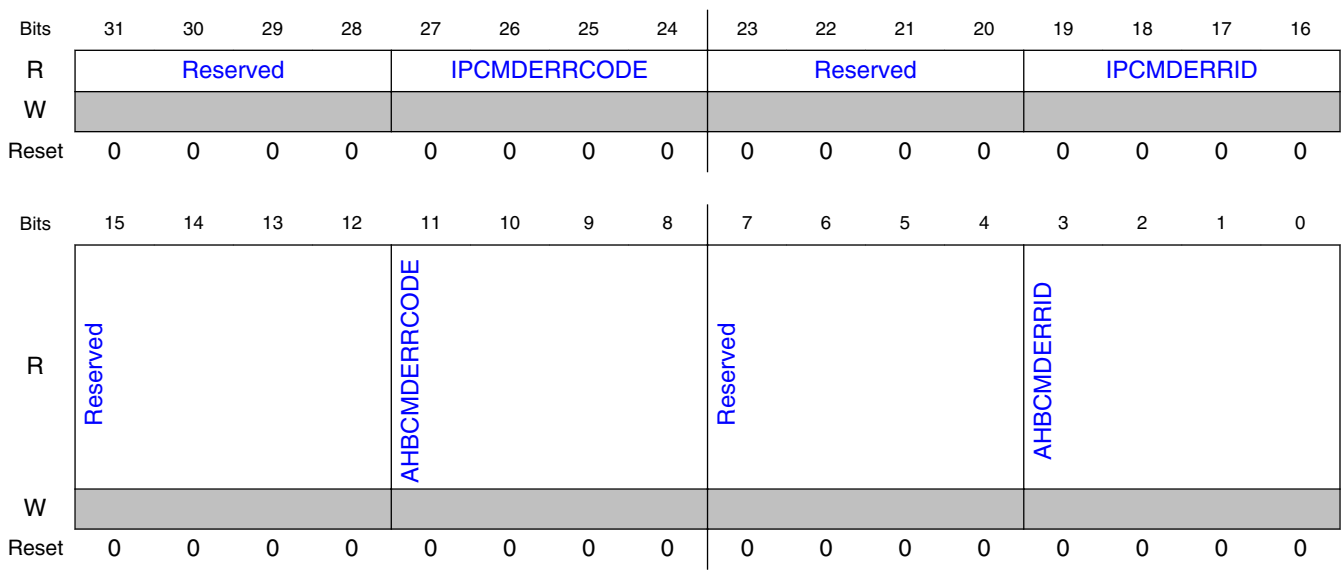
Field	Function
0 SEQIDLE	This status bit indicates the state machine in SEQ_CTL is idle and there is command sequence executing on FlexSPI interface.

25.7.2.25 Status Register 1 (STS1)

25.7.2.25.1 Offset

Register	Offset
STS1	E4h

25.7.2.25.2 Diagram



25.7.2.25.3 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).

25.7.2.26 Status Register 2 (STS2)

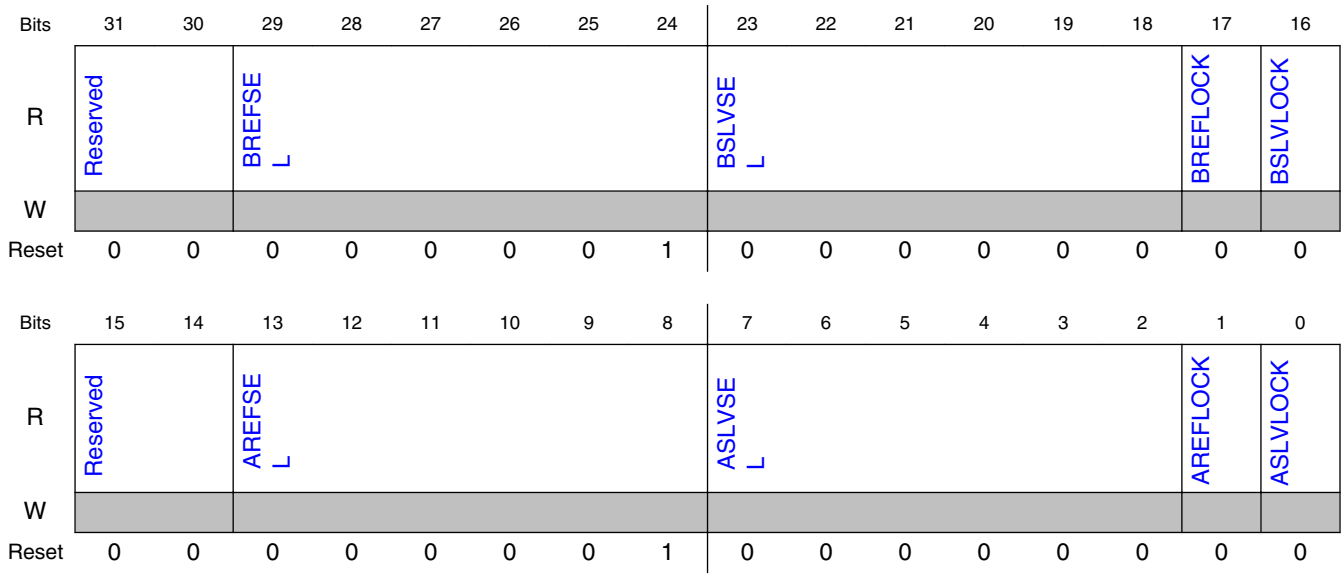
25.7.2.26.1 Offset

Register	Offset
STS2	E8h

25.7.2.26.2 Function

This register indicates the status of Flash A and B sample clock DLLs.

25.7.2.26.3 Diagram



25.7.2.26.4 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.

25.7.2.27 AHB Suspend Status Register (AHBSPNDSTS)

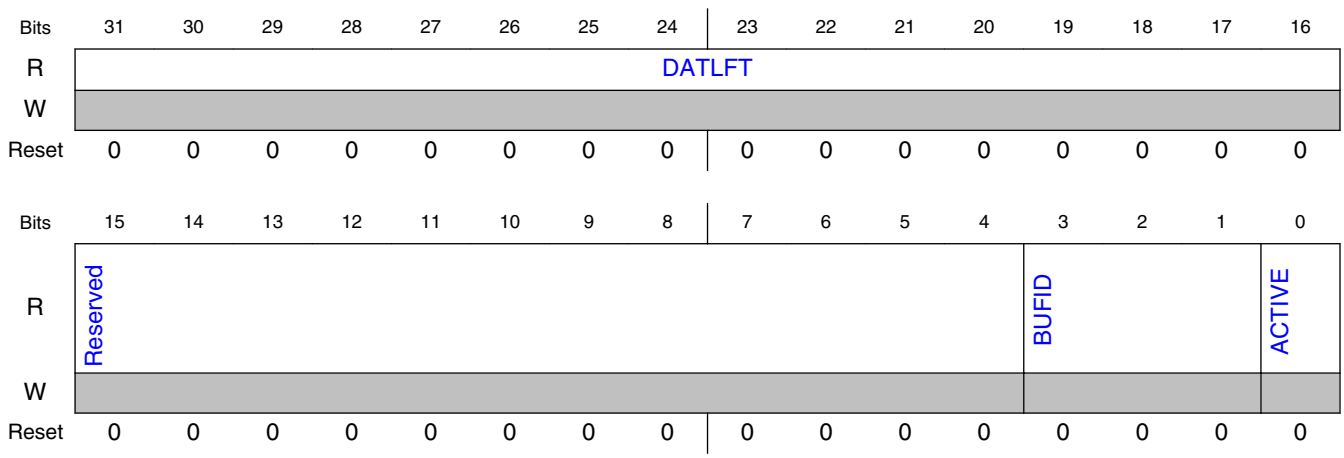
25.7.2.27.1 Offset

Register	Offset
AHBSPNDSTS	ECh

25.7.2.27.2 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.

25.7.2.27.3 Diagram



25.7.2.27.4 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.

25.7.2.28 IP RX FIFO Status Register (IPRXFSTS)

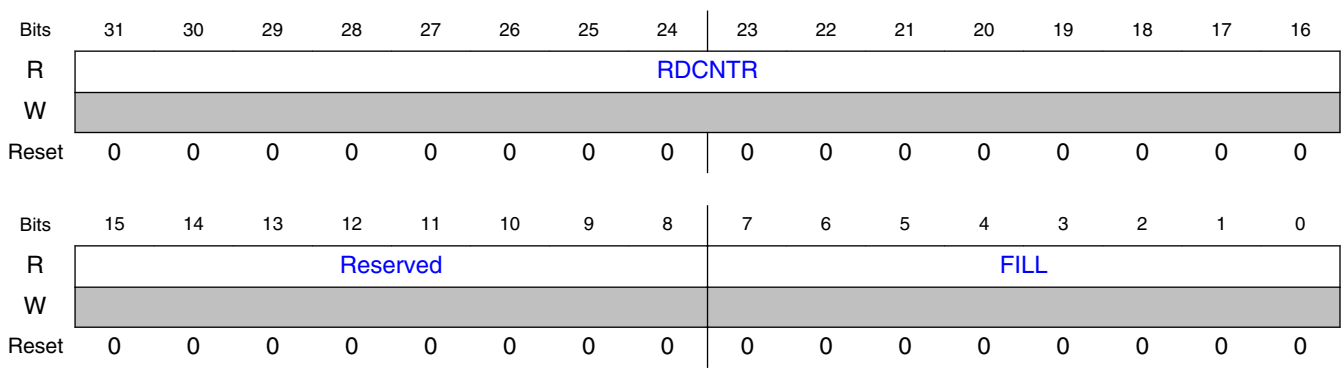
25.7.2.28.1 Offset

Register	Offset
IPRXFSTS	F0h

25.7.2.28.2 Function

This status register indicates the status of IP RX FIFO.

25.7.2.28.3 Diagram



25.7.2.28.4 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.

25.7.2.29 IP TX FIFO Status Register (IPTXFSTS)

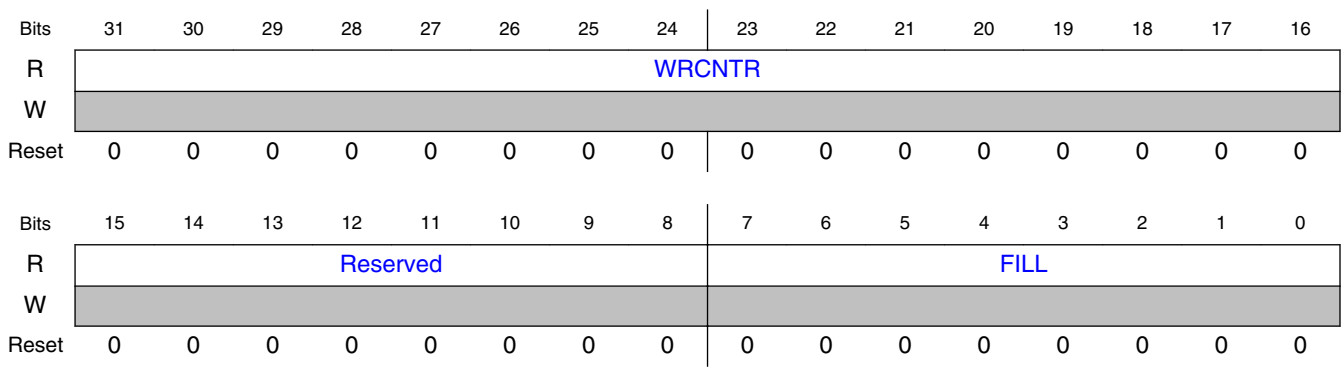
25.7.2.29.1 Offset

Register	Offset
IPTXFSTS	F4h

25.7.2.29.2 Function

This status register indicates the status of IP TX FIFO.

25.7.2.29.3 Diagram



25.7.2.29.4 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.

25.7.2.30 IP RX FIFO Data Register a (RFDR0 - RFDR31)

25.7.2.30.1 Offset

For a = 0 to 31:

Register	Offset
RFDRa	100h + (a × 4h)

25.7.2.30.2 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.

25.7.2.30.3 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

25.7.2.30.4 Fields

Field	Function
31-0 RXDATA	RX Data

25.7.2.31 IP TX FIFO Data Register a (TFDR0 - TFDR31)

25.7.2.31.1 Offset

For a = 0 to 31:

Register	Offset
TFDRa	180h + (a × 4h)

25.7.2.31.2 Function

These registers provide write access to IP TX FIFO by IPS bus.

25.7.2.31.3 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

25.7.2.31.4 Fields

Field	Function
31-0 TXDATA	TX Data

25.7.2.32 LUT a (LUT0 - LUT63)

25.7.2.32.1 Offset

For a = 0 to 63:

Register	Offset
LUTa	200h + (a × 4h)

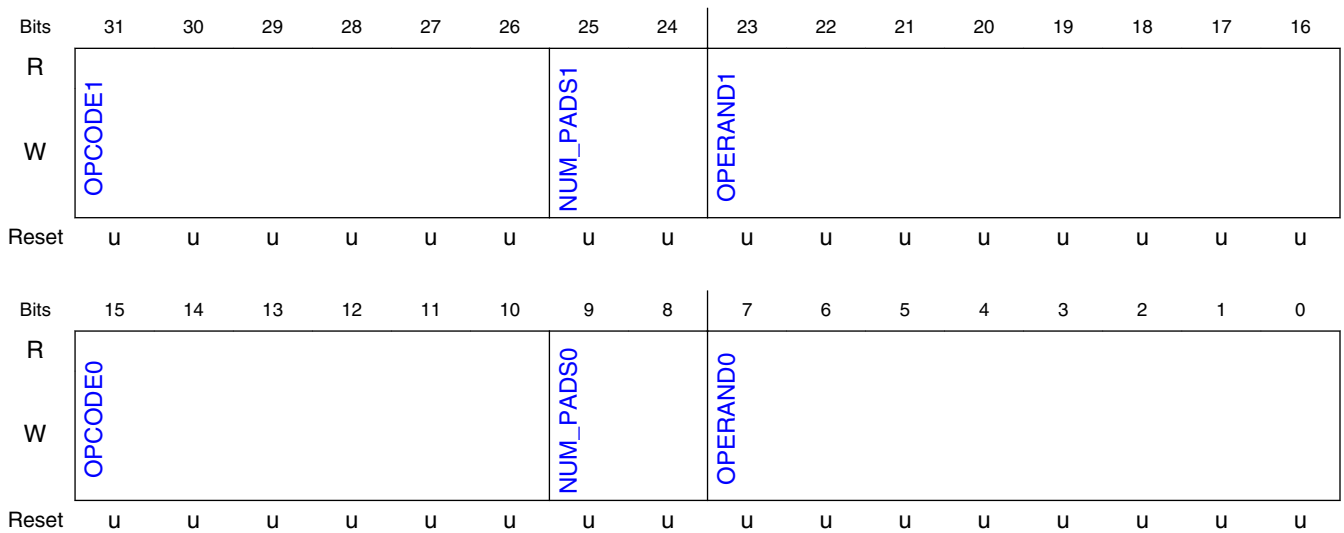
25.7.2.32.2 Function

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.

25.7.2.32.3 Diagram



25.7.2.32.4 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	

25.8 AHB Memory Map definition

This section describes FlexSPI module AHB memory map in detail.

25.8.1 AHB Memory Map for Serial Flash memory access

Following address range is mapped for AHB read/write access to serial flash memory: 0x60000000 - 0x80000000.

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.

25.8.2 AHB Memory Map for IP RX FIFO read access

Following address range is mapped for AHB read access to IP RX FIFO: 0x7FC00000 - 0x7FC00080.

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.

25.8.3 AHB Memory Map for IP TX FIFO write access

Following address range is mapped for AHB write access to IP TX FIFO: 0x7F800000 - 0x7F800080.

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 26
ARM Cortex M7 Platform

26.1 Chip-specific Arm Cortex M7 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
System Debug	-	System Debug

26.2 Arm Cortex M7 Platform

26.2.1 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:

- 16 KB L1 Instruction Cache
- 8 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 80. 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.

26.2.2 Block Diagram

A block diagram for the Cortex-M7 is given below:

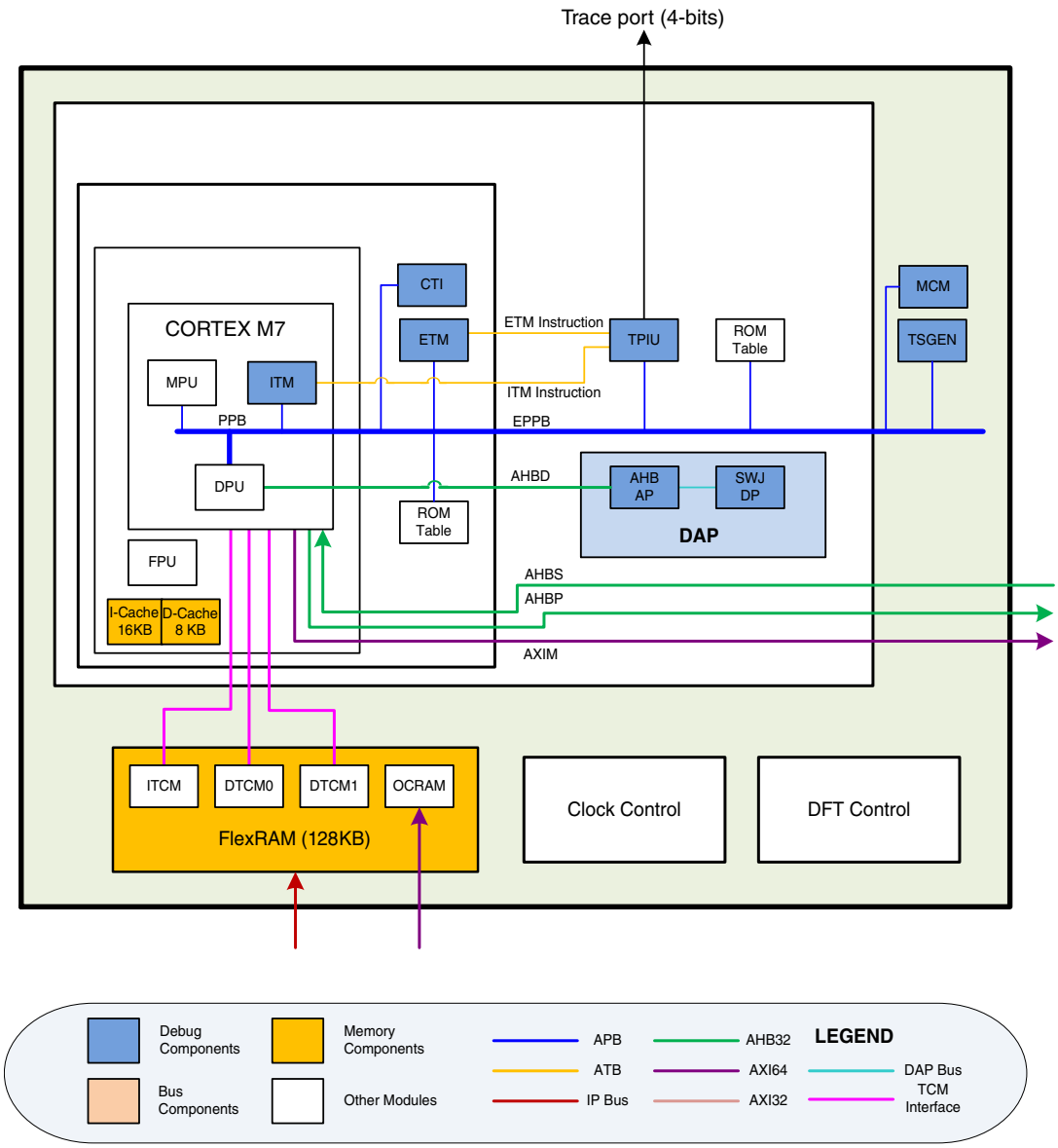


Figure 26-1. Cortex-M7 Platform Block Diagram

26.2.3 External Signals

26.2.4 Clocks

Table 26-2. Arm Clocks

Clock Name	Clock Root	Description
ipg_clk	ipg_clk_root	Peripheral clock
axi_clk	ipg_clk_root	Bus clock
main_clk	core_clk_root	Arm core clock

Table continues on the next page...

Table 26-2. Arm Clocks (continued)

trace_clk_in	trace_clk_root	Clock signal

Chapter 27
Network Interconnect Bus System (NIC-301)

27.1 Chip-specific NIC-301 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

Some modules (e.g. ENET, uSDHC) are not available in this device, so the related rows are NOT applicable in the tables of the "Memory Map and Register Definition" section.

27.2 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 are configured by NXP's board support package (BSP), and 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.

27.2.1 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.

27.2.2 Modes and Operations

The NIC-301 supports a normal functional mode only.

27.3 External Signals

There are no external I/O interfaces for NIC-301.

27.4 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*.

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	Setting this register to 1 can bypass the size merge function of NIC. **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

Table continues on the next page...

Register Name	Port Name	Module Name	Absolute Address	Reset Value	Descriptions
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. 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.
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
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

Table continues on the next page...

Register Name	Port Name	Module Name	Absolute Address	Reset Value	Descriptions
write_qos	m_c_5	TestPort	GPV1_BASE + 0x47000 + 0x104	0	Read-only register.
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 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

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 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:

Table continues on the next page...

Register Name	Port Name	Module Name	Absolute Address	Reset Value	Descriptions
					• 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 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

Table continues on the next page...

Register Name	Port Name	Module Name	Absolute Address	Reset Value	Descriptions
					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 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_M, and SIM_M7. For more details, please see the Arm document: *DDI0397I_corelink_network_interconnect_nic301_r2p3_trm.pdf*.

Chapter 28
On-Chip RAM Memory Controller (OCRAM)

28.1 Chip-specific OCRAM 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

28.2 Overview

There is 1 OCRAM controller implemented in the chip. One controller is for up to 128 KB on-chip RAM. The size of OCRAM is configured in the FlexRAM module via IOMUXC. For more details, refer to the FlexRAM chapter.

The on-chip RAM 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.

The internal block diagram is shown in the figure below.

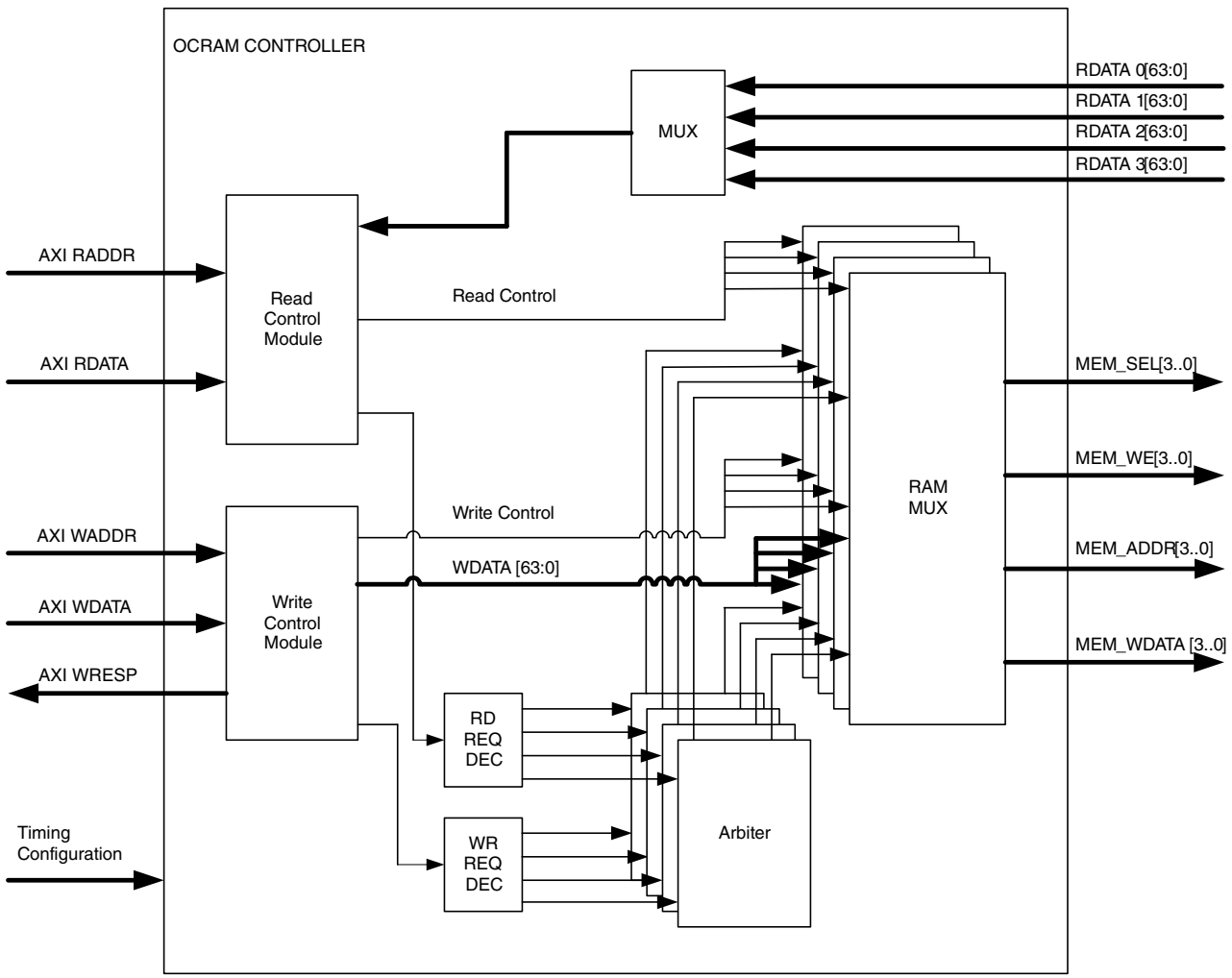


Figure 28-1. On-chip RAM Block Diagram

28.3 Basic Functions

28.3.1 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.

28.4 Advanced Features

This section describes some advanced features designed to avoid timing issues when the on-chip RAM 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.

28.4.1 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 normal OCRAM, the read data wait state is configurable via IOMUXC.GPR3[0].

28.4.2 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 on-chip RAM.

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 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).

For the normal OCRAM, the read address pipeline is configurable via IOMUXC.GPR3[1].

28.4.3 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 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, 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 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).

For the normal OCRAM, the write data pipeline is configurable via IOMUXC.GPR3[2].

28.4.4 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 on-chip RAM.

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 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).

For the normal OCRAM, the write address pipeline is configurable via IOMUXC.GPR3[3]

28.5 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 found here.

- TrustZone bits: IOMUXC_GPR10

Chapter 29
FlexRAM

29.1 Chip-specific FlexRAM 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

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

29.2 Overview

29.2.1 Introduction

This SoC has 128 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.

29.2.2 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 128 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 128 KB SRAM size
  - Synchronous to the system bus, runs at the same frequency as bus fabric
- Parameterized RAM Array
  - Support up to 128 KB total memory space
  - Support up to 8 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)/4
  - Flexible RAM bank organization.
- Flexible power mode:
  - Run mode: All RAM banks are powered on
  - Retention mode: All 4 banks are powered on

- Generates interrupt upon TCM, OCRAM access out of configured RAM range
- Generates interrupt when access to pre-configured OCRAM, ITCM and DTCM address occurs
  - Independent address control for OCRAM, ITCM and DTCM
  - Interrupt generation on read or write access based register settings

29.2.3 Block diagram

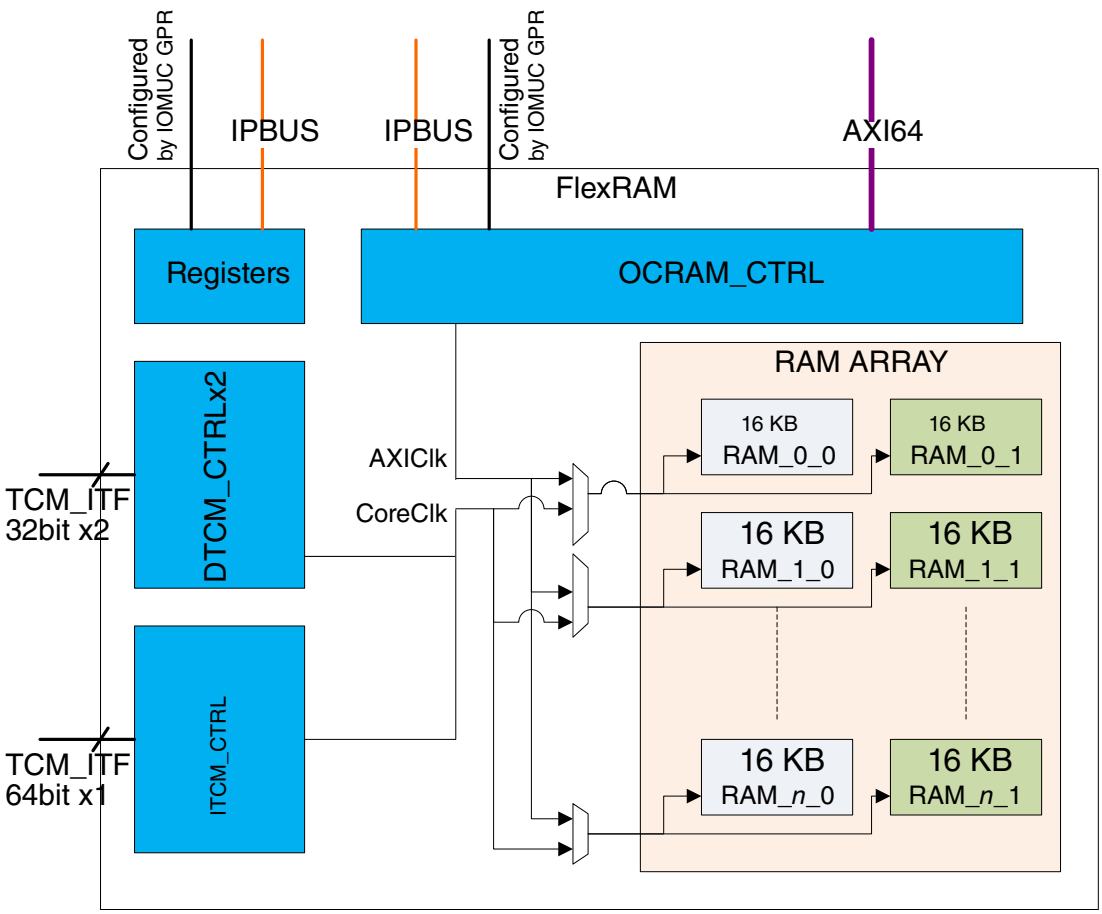


Figure 29-1. FLEXRAM block diagram

29.3 Memory Map and register definition

This section includes the FLEXRAM module memory map and detailed descriptions of all registers.

29.3.1 FLEXRAM register descriptions

29.3.1.1 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
4h	OCRAM Magic Address Register (OCRAM_MAGIC_ADDR)	32	RW	0000_0000h
8h	DTCM Magic Address Register (DTCM_MAGIC_ADDR)	32	RW	0000_0000h
Ch	ITCM Magic Address Register (ITCM_MAGIC_ADDR)	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

29.3.1.2 TCM CRTL Register (TCM_CTRL)

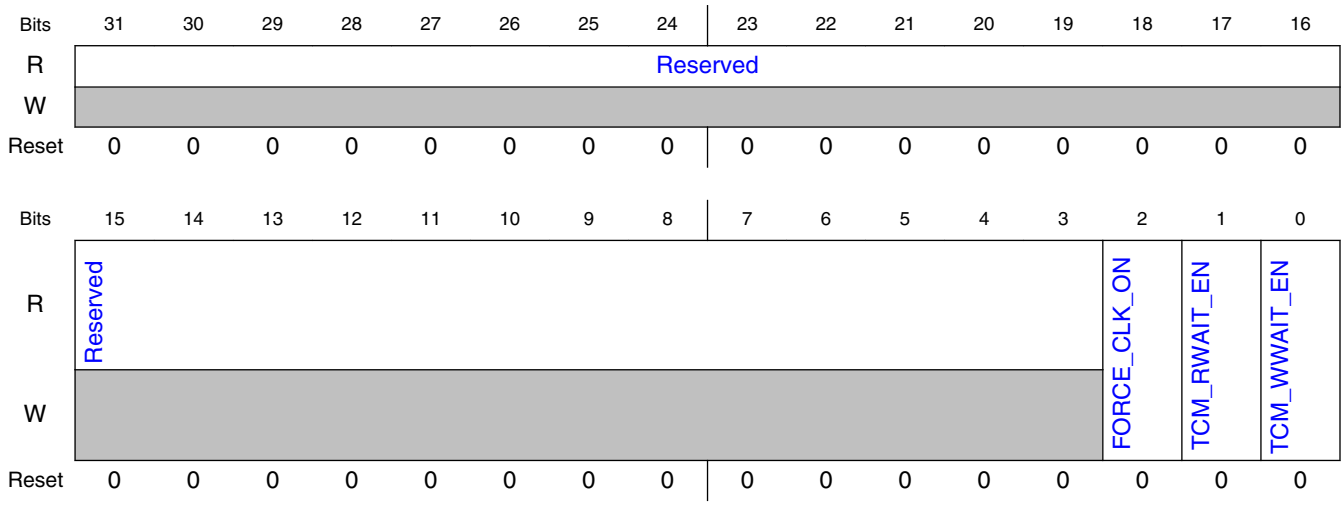
29.3.1.2.1 Offset

Register	Offset
TCM_CTRL	0h

29.3.1.2.2 Function

.

29.3.1.2.3 Diagram



29.3.1.2.4 Fields

Field	Function
31-3 —	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.

29.3.1.3 OCRAM Magic Address Register (OCRAM_MAGIC_ADDR)

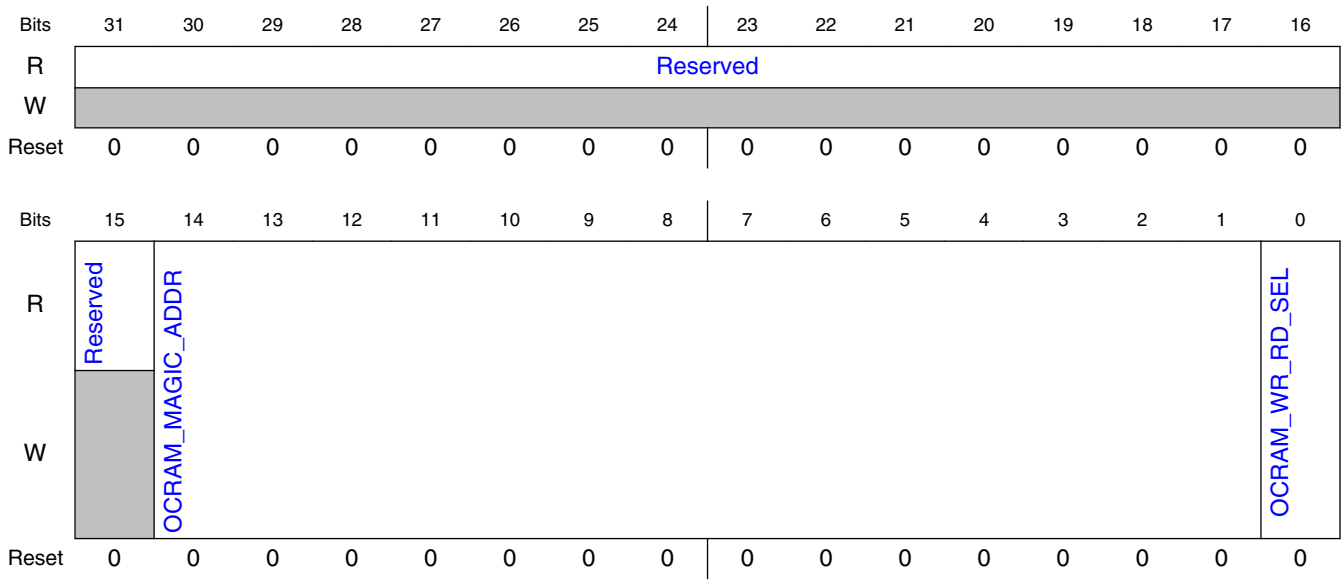
29.3.1.3.1 Offset

Register	Offset
OCRAM_MAGIC_ADDR	4h

29.3.1.3.2 Function

.

29.3.1.3.3 Diagram



29.3.1.3.4 Fields

Field	Function
31-15 —	Reserved
14-1 OCRAM_MAGIC_ADDR	OCRAM Magic Address When OCRAM access hits Magic Address, it will generate interrupt if access type is match with ocram_wr_rd_sel.
0 OCRAM_WR_RD_SEL	OCRAM Write Read Select 0b - When OCRAM read access hits magic address, it will generate interrupt. 1b - When OCRAM write access hits magic address, it will generate interrupt.

29.3.1.4 DTCM Magic Address Register (DTCM_MAGIC_ADDR)

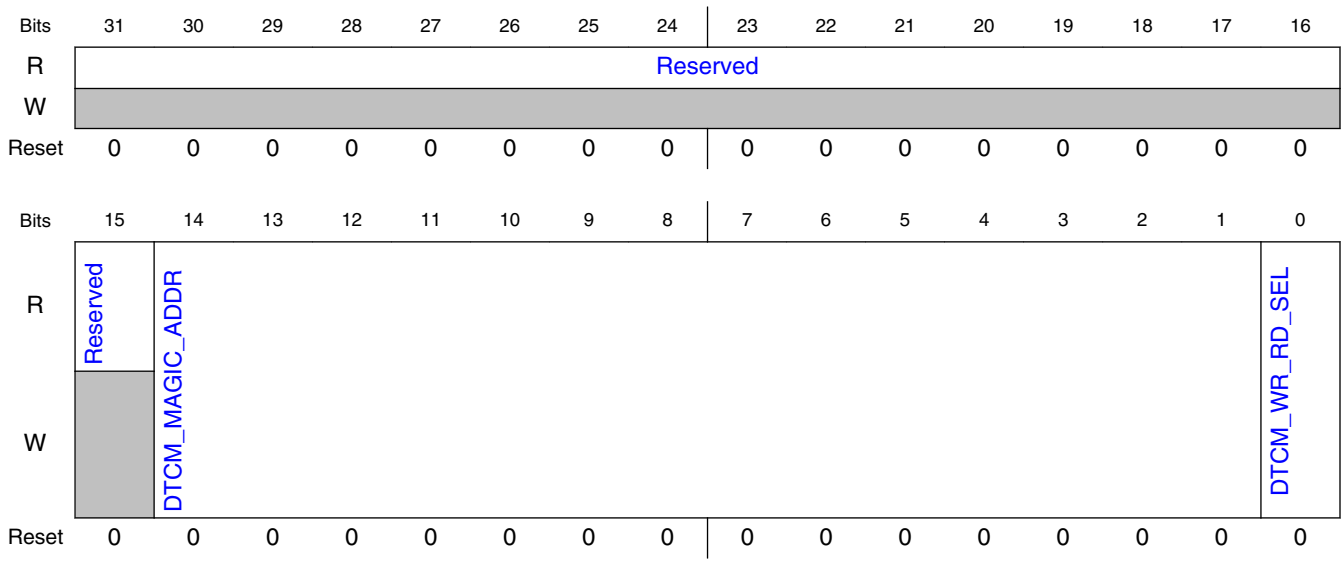
29.3.1.4.1 Offset

Register	Offset
DTCM_MAGIC_ADDR	8h

29.3.1.4.2 Function

.

29.3.1.4.3 Diagram



29.3.1.4.4 Fields

Field	Function
31-15 —	Reserved
14-1 DTCM_MAGIC_ADDR	DTCM Magic Address When DTCM access hits Magic Address, it will generate interrupt if access type is match with dtcm_wr_rd_sel.
0 DTCM_WR_RD_SEL	DTCM Write Read Select 0b - When DTCM read access hits magic address, it will generate interrupt. 1b - When DTCM write access hits magic address, it will generate interrupt.

29.3.1.5 ITCM Magic Address Register (ITCM_MAGIC_ADDR)

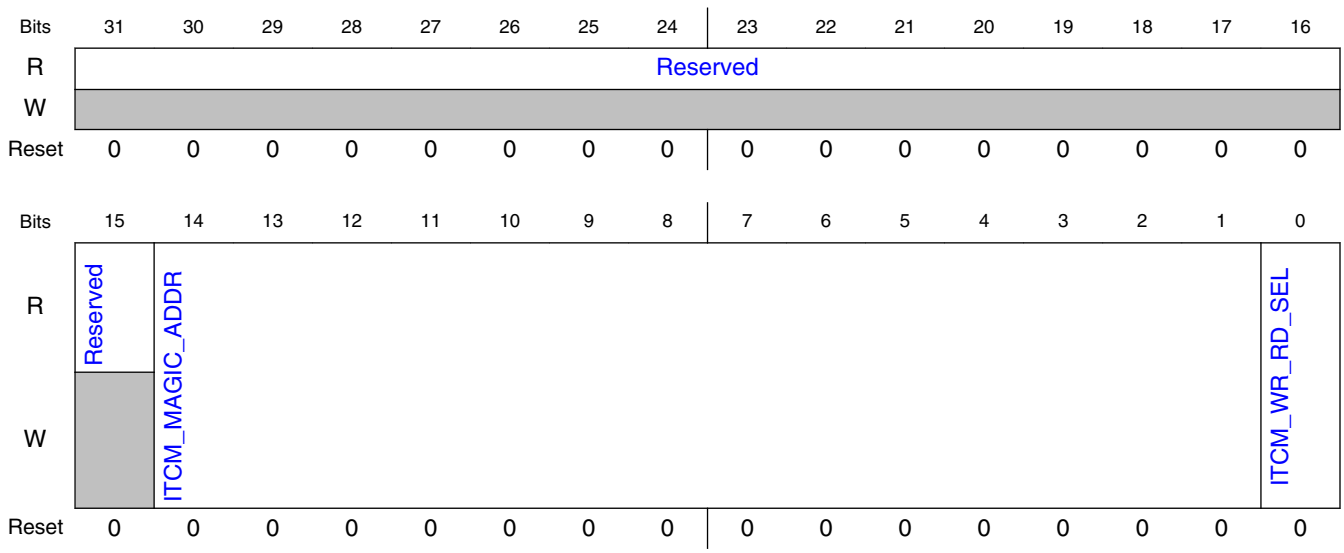
29.3.1.5.1 Offset

Register	Offset
ITCM_MAGIC_ADDR	Ch

29.3.1.5.2 Function

.

29.3.1.5.3 Diagram



29.3.1.5.4 Fields

Field	Function
31-15 —	Reserved
14-1 ITCM_MAGIC_ADDR	ITCM Magic Address When ITCM access hits Magic Address, it will generate interrupt if access type is match with itcm_wr_rd_sel.
0 ITCM_WR_RD_SEL	ITCM Write Read Select 0b - When ITCM read access hits magic address, it will generate interrupt. 1b - When ITCM write access hits magic address, it will generate interrupt.

29.3.1.6 Interrupt Status Register (INT_STATUS)

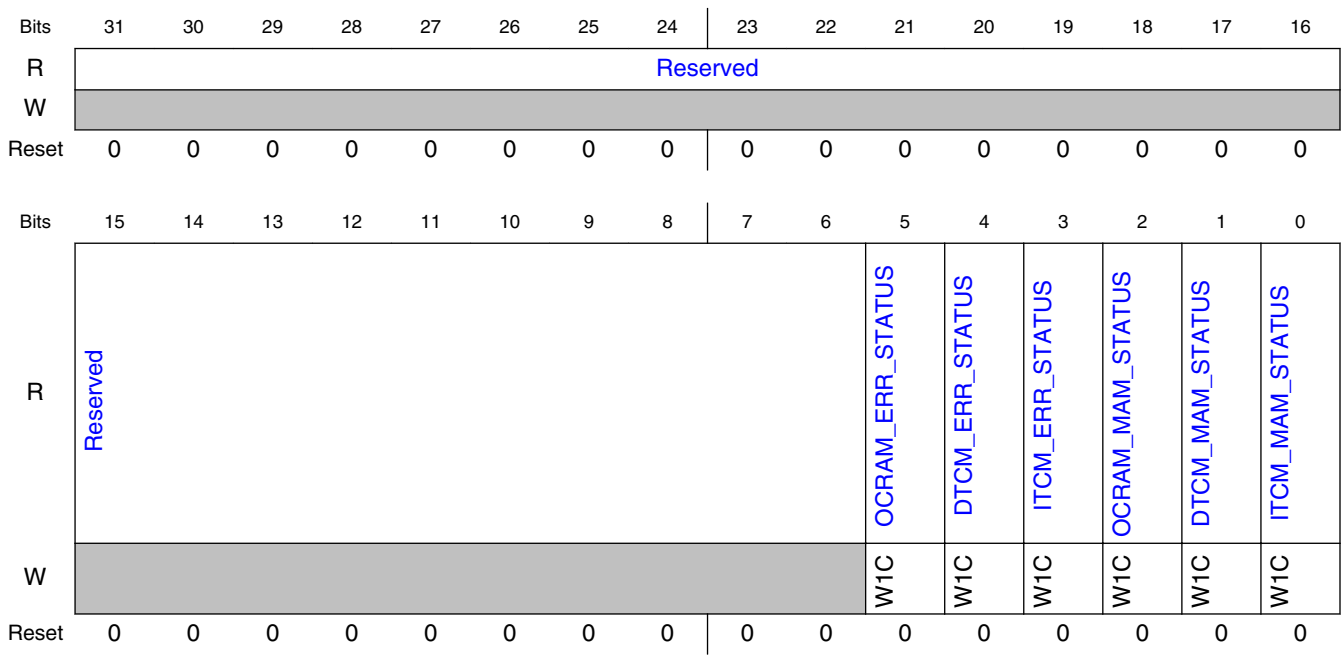
29.3.1.6.1 Offset

Register	Offset
INT_STATUS	10h

29.3.1.6.2 Function

.

29.3.1.6.3 Diagram



29.3.1.6.4 Fields

Field	Function
31-6 —	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.

Table continues on the next page...

Field	Function
	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 OCRAM_MAM_STATUS	OCRAM Magic Address Match Status When OCRAM access hits Magic Address and access type is also matched, this bit will be asserted if corresponding status enable bit in INT_STAT_EN register is set. 0b - OCRAM did not access magic address. 1b - OCRAM accessed magic address.
1 DTCM_MAM_STATUS	DTCM Magic Address Match Status When DTCM access hits Magic Address and access type is also matched, this bit will be asserted if corresponding status enable bit in INT_STAT_EN register is set. 0b - DTCM did not access magic address. 1b - DTCM accessed magic address.
0 ITCM_MAM_STATUS	ITCM Magic Address Match Status When ITCM access hits Magic Address and access type is also matched, this bit will be asserted if corresponding status enable bit in INT_STAT_EN register is set. 0b - ITCM did not access magic address. 1b - ITCM accessed magic address.

29.3.1.7 Interrupt Status Enable Register (INT_STAT_EN)

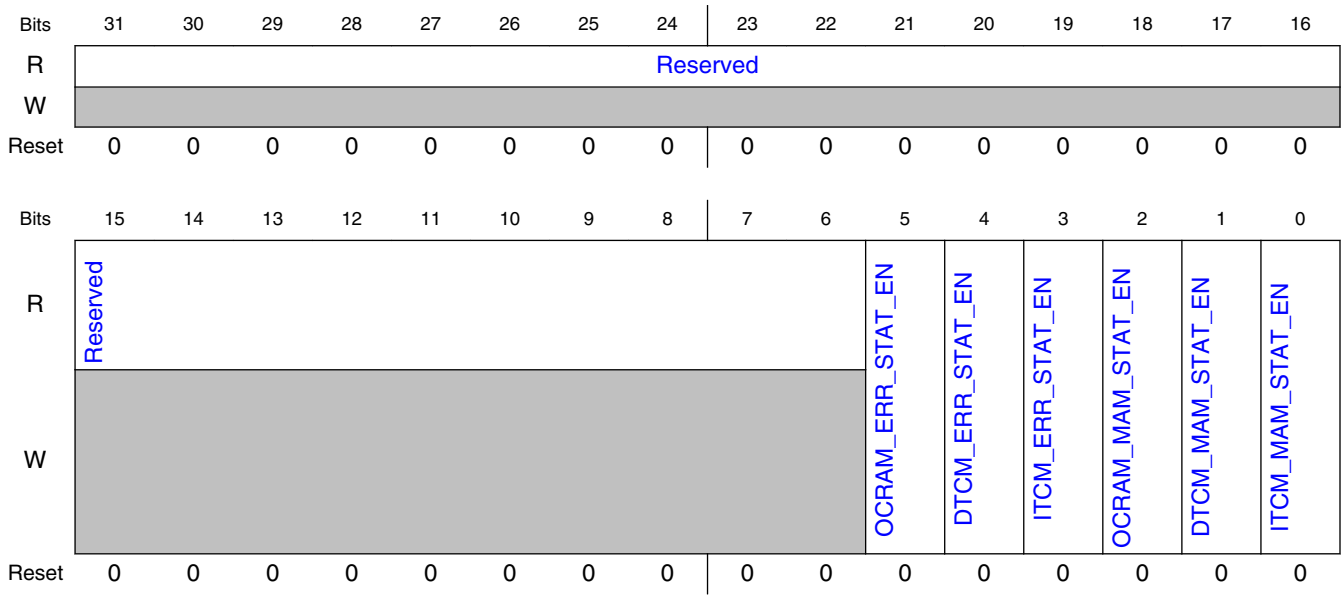
29.3.1.7.1 Offset

Register	Offset
INT_STAT_EN	14h

29.3.1.7.2 Function

Setting the bits in this register to 1 enables the corresponding Interrupt Status to be set by the specified event.

29.3.1.7.3 Diagram



29.3.1.7.4 Fields

Field	Function
31-6 —	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 OCRAM_MAM_STAT_EN	OCRAM Magic Address Match Status Enable 0b - Masked 1b - Enabled
1 DTCM_MAM_STAT_EN	DTCM Magic Address Match Status Enable 0b - Masked 1b - Enabled
0 ITCM_MAM_STAT_EN	ITCM Magic Address Match Status Enable 0b - Masked 1b - Enabled

29.3.1.8 Interrupt Enable Register (INT_SIG_EN)

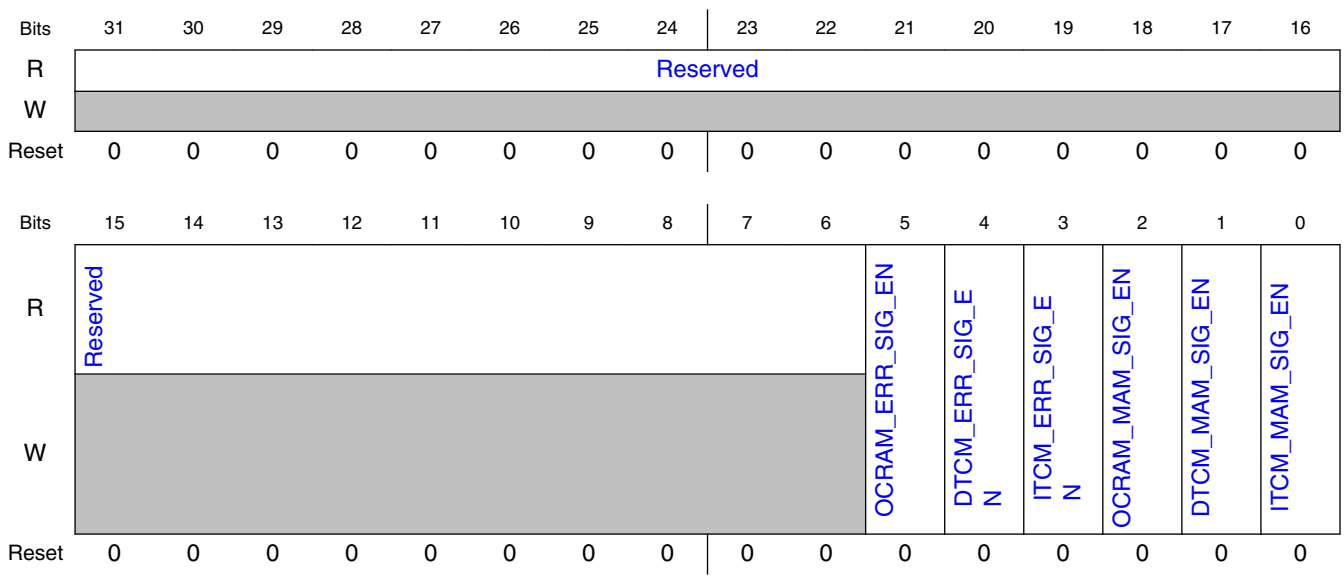
29.3.1.8.1 Offset

Register	Offset
INT_SIG_EN	18h

29.3.1.8.2 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.

29.3.1.8.3 Diagram



29.3.1.8.4 Fields

Field	Function
31-6 —	Reserved
5	OCRAM Access Error Interrupt Enable 0b - Masked

Table continues on the next page...

Field	Function
OCRAM_ERR_SIG_EN	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 OCRAM_MAM_SIG_EN	OCRAM Magic Address Match Interrupt Enable 0b - Masked 1b - Enabled
1 DTCM_MAM_SIG_EN	DTCM Magic Address Match Interrupt Enable 0b - Masked 1b - Enabled
0 ITCM_MAM_SIG_EN	ITCM Magic Address Match Interrupt Enable 0b - Masked 1b - Enabled

29.4 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.

29.4.1 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.

29.4.2 RAM Bank Allocation

OCRAM, DTCM and ITCM share 128 KB of on-chip RAM. Software can configure from 0 to 128 KB full RAM array size for OCRAM, DTCM and ITCM by bank configuration signal (e.g. IOMUX GPR16 and GPR17). The RAM size step is 32 KB. The allocated RAM bank can be not overlapped for OCRAM, DTCM and ITCM. Table below show an example of a 32KB ITCM, 32KB DTCM and 64KB OCRAM configuration.

Table 29-2. FLEXRAM Partitioning Example (4 Banks Case)

	ITCM	DTCM	OCRAM
Bank0	0	0	1
Bank1	0	0	1
Bank2	0	1	0
Bank3	1	0	0

Each RAM bank has 2 bits bank configuration signal (e.g. from IOMUX GPR17).

Table 29-3. RAM Bank configuration

	RAM Bank Configuration
00	Not used
01	OCRAM
10	DTCM
11	ITCM

29.4.3 Clocks

The following table describes the clock sources of the FlexRAM module.

Table 29-4. 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

29.4.4 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.

29.4.5 Interrupts

FLEXRAM can generate interrput in the following cases:

- OCRAM, DTCM or ITCM access the RAM address which is not allocated.
- OCRAM, DTCM or ITCM access the RAM address which is assigned in *_MAGIC_ADDR register.

Chapter 30
AHB to IP Bridge (AIPSTZ)

30.1 Chip-specific AIPSTZ 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

30.2 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.

30.2.1 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.

30.3 Clocks

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

Table 30-2. AIPSTZ Clocks

Clock name	Clock Root	Description
hclk	ahb_clk_root	Module clock

30.4 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.

30.5 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.

30.6 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.

30.7 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.

30.7.1 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 30-3. 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 30-3. 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 30-4. 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.

30.8 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	30.8.1/1016
4007_C040	Off-Platform Peripheral Access Control Registers (AIPSTZ1_OPACR)	32	R/W	4444_4444h	30.8.2/1019
4007_C044	Off-Platform Peripheral Access Control Registers (AIPSTZ1_OPACR1)	32	R/W	4444_4444h	30.8.3/1022
4007_C048	Off-Platform Peripheral Access Control Registers (AIPSTZ1_OPACR2)	32	R/W	4444_4444h	30.8.4/1025
4007_C04C	Off-Platform Peripheral Access Control Registers (AIPSTZ1_OPACR3)	32	R/W	4444_4444h	30.8.5/1028
4007_C050	Off-Platform Peripheral Access Control Registers (AIPSTZ1_OPACR4)	32	R/W	4444_4444h	30.8.6/1031
4017_C000	Master Priviledge Registers (AIPSTZ2_MPR)	32	R/W	7700_0000h	30.8.1/1016
4017_C040	Off-Platform Peripheral Access Control Registers (AIPSTZ2_OPACR)	32	R/W	4444_4444h	30.8.2/1019
4017_C044	Off-Platform Peripheral Access Control Registers (AIPSTZ2_OPACR1)	32	R/W	4444_4444h	30.8.3/1022
4017_C048	Off-Platform Peripheral Access Control Registers (AIPSTZ2_OPACR2)	32	R/W	4444_4444h	30.8.4/1025
4017_C04C	Off-Platform Peripheral Access Control Registers (AIPSTZ2_OPACR3)	32	R/W	4444_4444h	30.8.5/1028
4017_C050	Off-Platform Peripheral Access Control Registers (AIPSTZ2_OPACR4)	32	R/W	4444_4444h	30.8.6/1031

30.8.1 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 30-5

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 30-5. 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 30-6. 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				MPROT5				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.

Table continues on the next page...

AIPSTZx_MPR field descriptions (continued)

Field	Description
	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. 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 MPROT5	Master 5 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
-	This field is reserved. Reserved

30.8.2 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 30-7

Table 30-7. 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.

Table continues on the next page...

AIPSTZx_OPACR field descriptions (continued)

Field	Description
	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. 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.

Table continues on the next page...

AIPSTZx_OPACR field descriptions (continued)

Field	Description
	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 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.

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.
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.

30.8.3 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 30-7

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.

Table continues on the next page...

AIPSTZx_OPACR1 field descriptions (continued)

Field	Description
	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 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.

Table continues on the next page...

AIPSTZx_OPACR1 field descriptions (continued)

Field	Description
	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. 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.

Table continues on the next page...

AIPSTZx_OPACR1 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.
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.

30.8.4 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 30-7

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.

Table continues on the next page...

AIPSTZx_OPACR2 field descriptions (continued)

Field	Description
	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 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.

Table continues on the next page...

AIPSTZx_OPACR2 field descriptions (continued)

Field	Description
	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. 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.

Table continues on the next page...

AIPSTZx_OPACR2 field descriptions (continued)

Field	Description
	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.

30.8.5 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 30-7

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.

Table continues on the next page...

AIPSTZx_OPACR3 field descriptions (continued)

Field	Description
	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 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.

Table continues on the next page...

AIPSTZx_OPACR3 field descriptions (continued)

Field	Description
	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. 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.

Table continues on the next page...

AIPSTZx_OPACR3 field descriptions (continued)

Field	Description
	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.

30.8.6 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 30-7

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 31
Audio Overview

31.1 Audio Overview

The audio subsystem consists of the following modules: SAI-1, 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.

31.1.1 Audio Module Overview

The following figure shows a high level block diagram of the audio subsystem.

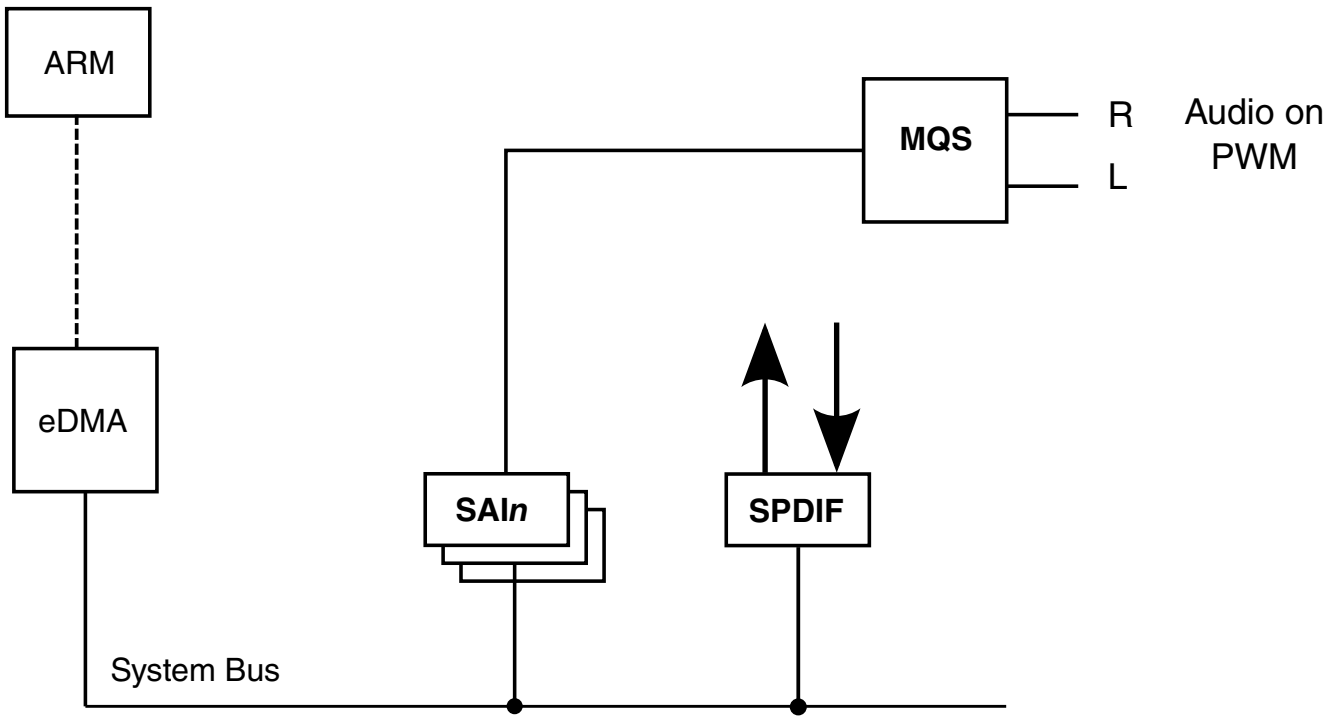


Figure 31-1. Audio peripherals block diagram

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 31-1. Audio Interface Summary

Interface	Function	RX Data Line	TX Data Line
SAI-1	External audio	2	2
SAI-3	External audio	1	1
MQS	External audio	0	2
SPDIF	External audio	1	1

SAI-1 is used for multi-channel audio interface, which supports up to 2-channels audio input or 2-channels audio output at 384 kHz/32-bit. 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.

Table 31-2. SAI-1 TX/RX Data Multiplexing

Options	RX Data Line	TX Data Line
0	1	2
1	2	1

SAI-1 to IOMUXC connection is implemented as shown in the diagram below.

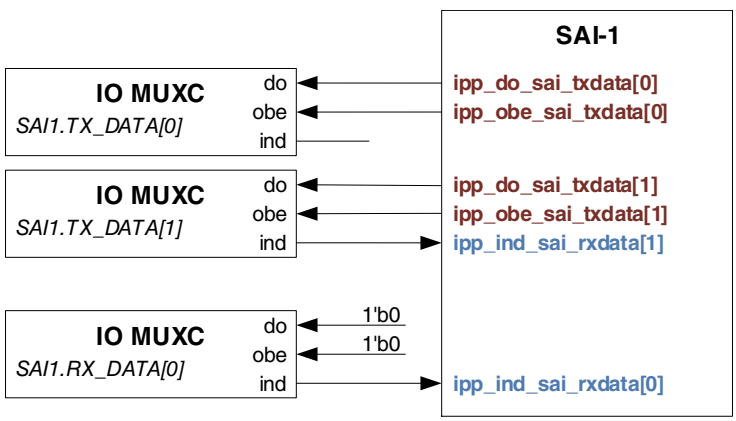


Figure 31-2. SAI-1 TX/RX Data Multiplexing

Detailed clock multiplexing scheme is shown in the following figure.

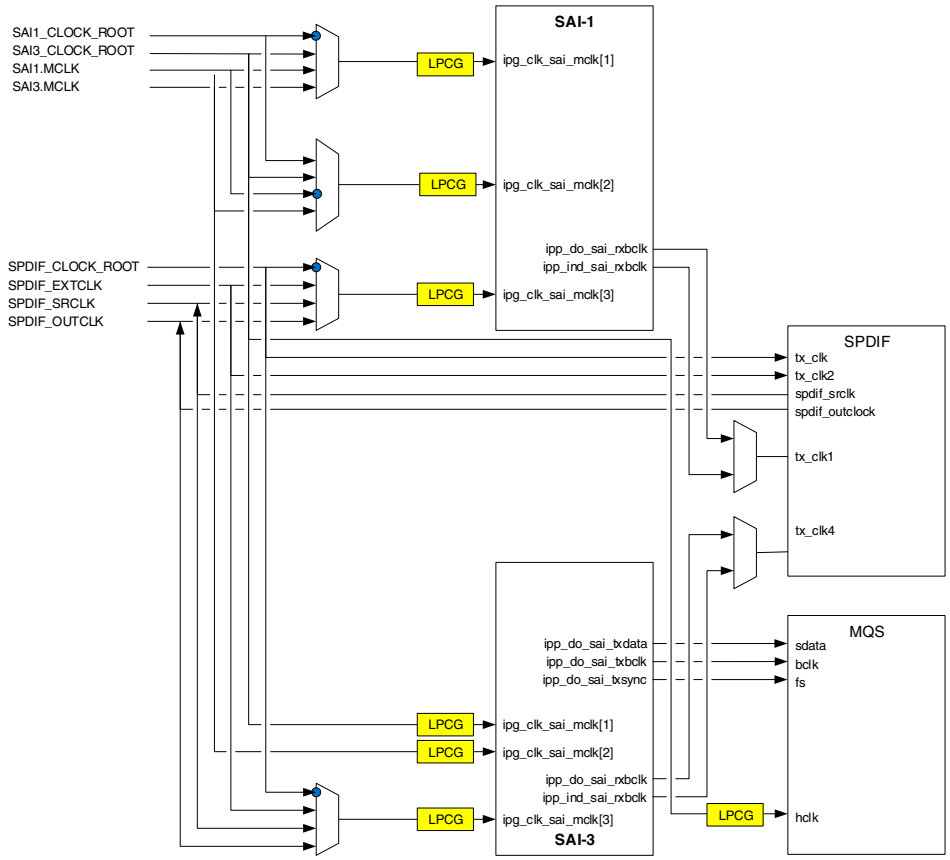


Figure 31-3. Audio subsystem clocking diagram

31.1.2 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 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

31.1.3 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

31.1.4 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, ESAI 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 32
Synchronous Audio Interface (SAI)

32.1 Chip-specific SAI information

Table 32-1. 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 SAI3 are 2, and 1 respectively.

NOTE

In this device, it does NOT support synchronous mode between different SAI peripherals.

For more details about the features and clocking of different SAI instances, see the Audio Subsystem Overview section of the Multimedia chapter.

32.2 Introduction

The I^2S (or I2S) module provides a synchronous audio interface (SAI) that supports full-duplex serial interfaces with frame synchronization such as I^2S, AC97, TDM, and codec/DSP interfaces.

32.2.1 Features

Note that some of the features are not supported across all SAI instances; see the chip-specific information in the first section of this chapter.

- Transmitter with independent bit clock and frame sync supporting 2 data lines
- Receiver with independent bit clock and frame sync supporting 2 data lines
- Each data line can support a maximum Frame size of 32 words
- Word size of between 8-bits and 32-bits
- Word size configured separately for first word and remaining words in frame
- Asynchronous 32 x 32-bit FIFO for each transmit and receive data line
- Supports graceful restart after FIFO error
- Supports automatic restart after FIFO error without software intervention
- Supports packing of 8-bit and 16-bit data into each 32-bit FIFO word
- Supports combining multiple data line FIFOs into single data line FIFO

32.2.2 Block diagram

The following block diagram also shows the module clocks.

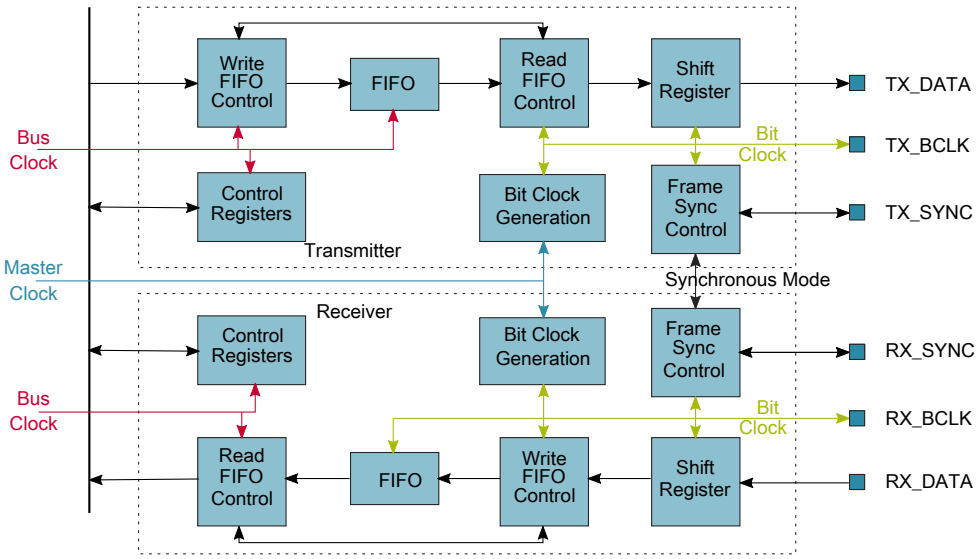


Figure 32-1. I²S/SAI block diagram

32.2.3 Modes of operation

Module power modes include Run mode, Stop modes, and Debug mode.

32.2.3.1 Run mode

In Run mode, the SAI transmitter and receiver operate normally.

32.2.3.2 Stop modes

In Stop mode, the SAI transmitter and/or receiver can continue operating provided the appropriate Stop Enable bit is set (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.

32.2.3.3 Debug mode

In Debug mode, the SAI transmitter and/or receiver can continue operating provided 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. The transmitter and receiver bit clocks are not affected by Debug mode.

32.3 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[1:0]	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[1:0]	Receive Data. The receive data is sampled synchronously by the bit clock.	I

32.4 Memory map and register definition

A read or write access to an address from offset 0x100 and above will result in a bus error.

32.4.1 I2S register descriptions

32.4.1.1 I2S memory map

SAI1 base address: 401E_0000h

SAI3 base address: 401E_8000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Version ID Register (VERID)	32	RO	0300_0000h
4h	Parameter Register (PARAM)	32	RO	Table 32-1
8h	SAI Transmit Control Register (TCSR)	32	RW	0000_0000h
Ch	SAI Transmit Configuration 1 Register (TCR1)	32	RW	0000_0000h
10h	SAI Transmit Configuration 2 Register (TCR2)	32	RW	0000_0000h
14h	SAI Transmit Configuration 3 Register (TCR3)	32	RW	0000_0000h
18h	SAI Transmit Configuration 4 Register (TCR4)	32	RW	0000_0000h
1Ch	SAI Transmit Configuration 5 Register (TCR5)	32	RW	0000_0000h
20h - 24h	SAI Transmit Data Register (TDR0 - TDR1)	32	WORZ	See description
40h - 44h	SAI Transmit FIFO Register (TFR0 - TFR1)	32	RO	See description
60h	SAI Transmit Mask Register (TMR)	32	RW	0000_0000h
88h	SAI Receive Control Register (RCSR)	32	RW	0000_0000h
8Ch	SAI Receive Configuration 1 Register (RCR1)	32	RW	0000_0000h
90h	SAI Receive Configuration 2 Register (RCR2)	32	RW	0000_0000h
94h	SAI Receive Configuration 3 Register (RCR3)	32	RW	0000_0000h
98h	SAI Receive Configuration 4 Register (RCR4)	32	RW	0000_0000h
9Ch	SAI Receive Configuration 5 Register (RCR5)	32	RW	0000_0000h
A0h - A4h	SAI Receive Data Register (RDR0 - RDR1)	32	RO	See description
C0h - C4h	SAI Receive FIFO Register (RFR0 - RFR1)	32	RO	See description
E0h	SAI Receive Mask Register (RMR)	32	RW	0000_0000h

32.4.1.2 Version ID Register (VERID)

32.4.1.2.1 Offset

Register	Offset
VERID	0h

32.4.1.2.2 Function

Contains version numbers for the module design and feature set.

32.4.1.2.3 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

32.4.1.2.4 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.

32.4.1.3 Parameter Register (PARAM)

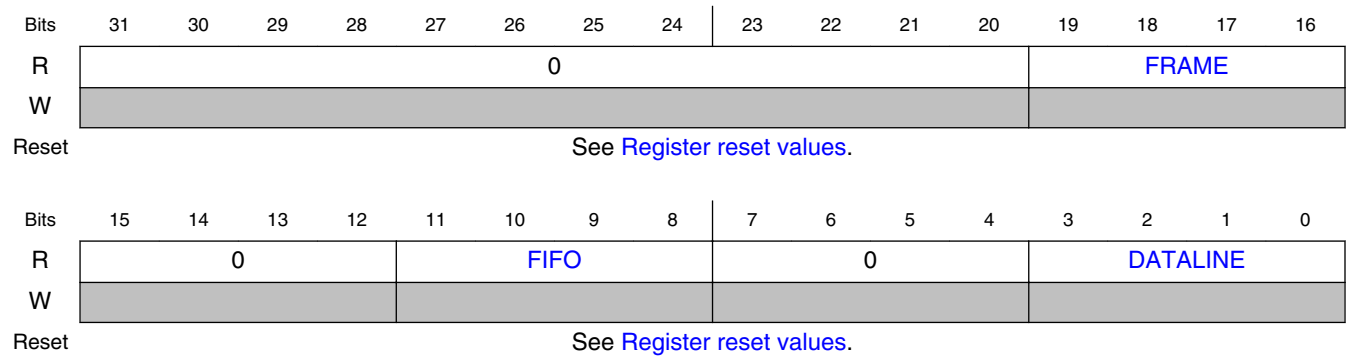
32.4.1.3.1 Offset

Register	Offset
PARAM	4h

32.4.1.3.2 Function

Contains parameter values that were implemented in the module.

32.4.1.3.3 Diagram



32.4.1.3.4 Register reset values

Register	Reset value
PARAM	SAI1: 0005_0502h SAI3: 0005_0501h

32.4.1.3.5 Fields

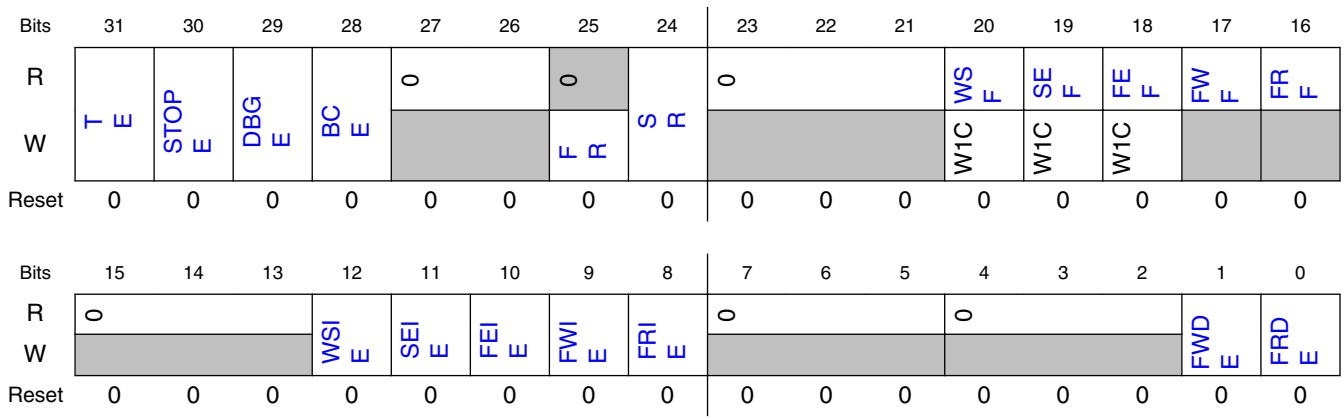
Field	Function
31-20 —	Reserved
19-16 FRAME	Frame Size The maximum number of slots per frame is 2^FRAME.
15-12 —	Reserved
11-8 FIFO	FIFO Size The number of words in each FIFO is 2^FIFO.
7-4 —	Reserved
3-0 DATALINE	Number of Datalines The number of datalines implemented.

32.4.1.4 SAI Transmit Control Register (TCSR)

32.4.1.4.1 Offset

Register	Offset
TCSR	8h

32.4.1.4.2 Diagram



32.4.1.4.3 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 DBGE	Debug Enable 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.

Table continues on the next page...

Field	Function
	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.
15-13 —	Reserved
12 WSIE	Word Start Interrupt Enable Enables/disables word start interrupts. 0b - Disables interrupt. 1b - Enables interrupt.
11	Sync Error Interrupt Enable

Table continues on the next page...

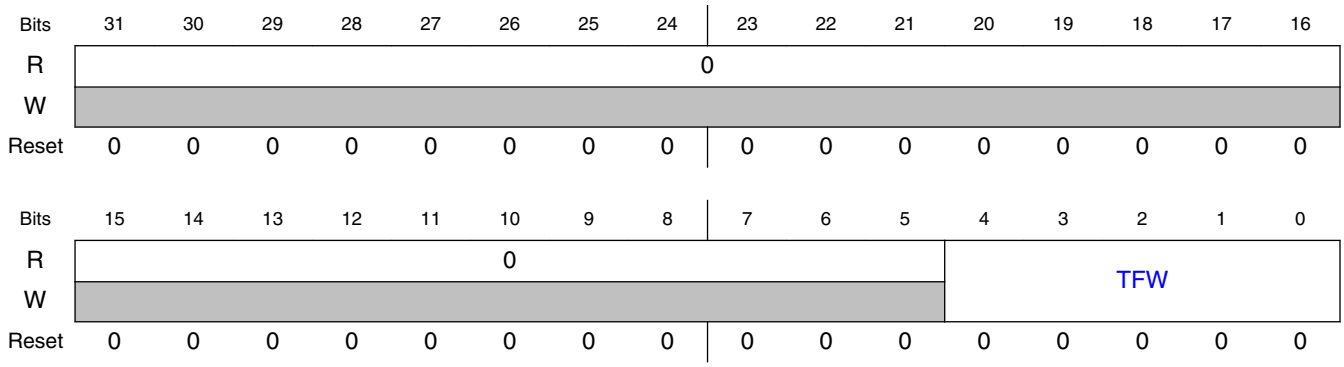
Field	Function
SEIE	Enables/disables sync error interrupts. 0b - Disables interrupt. 1b - Enables interrupt.
10 FEIE	FIFO Error Interrupt Enable Enables/disables FIFO error interrupts. 0b - Disables the interrupt. 1b - Enables the interrupt.
9 FWIE	FIFO Warning Interrupt Enable Enables/disables FIFO 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.

32.4.1.5 SAI Transmit Configuration 1 Register (TCR1)

32.4.1.5.1 Offset

Register	Offset
TCR1	Ch

32.4.1.5.2 Diagram



32.4.1.5.3 Fields

Field	Function
31-5 —	Reserved
4-0 TFW	Transmit FIFO Watermark Configures the watermark level for all enabled transmit channels.

32.4.1.6 SAI Transmit Configuration 2 Register (TCR2)

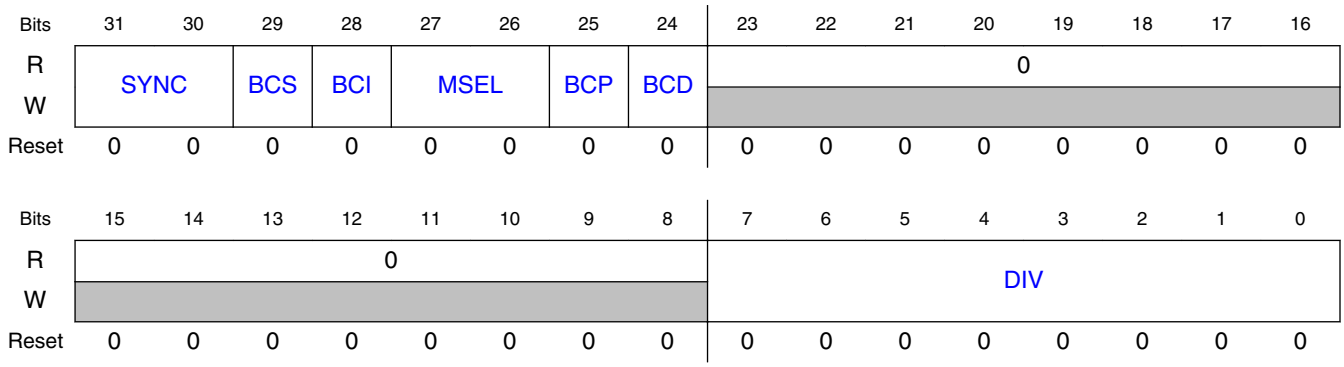
32.4.1.6.1 Offset

Register	Offset
TCR2	10h

32.4.1.6.2 Function

This register must not be altered when TCSR[TE] is set.

32.4.1.6.3 Diagram



32.4.1.6.4 Fields

Field	Function
31-30 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. 00b - Asynchronous mode. 01b - Synchronous with receiver. 10b - Reserved. 11b - Reserved.
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 MSEL	MCLK Select 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.

Table continues on the next page...

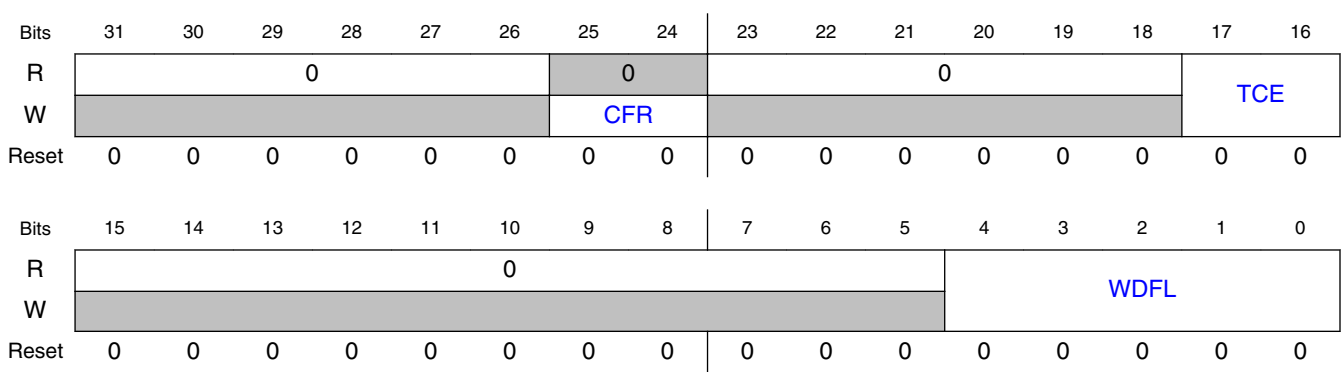
Field	Function
	NOTE: Depending on the device, some Master Clock options might not be available. See the chip-specific information for the meaning of each option. 00b - Bus Clock selected. 01b - Master Clock (MCLK) 1 option selected. 10b - Master Clock (MCLK) 2 option selected. 11b - Master Clock (MCLK) 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.

32.4.1.7 SAI Transmit Configuration 3 Register (TCR3)

32.4.1.7.1 Offset

Register	Offset
TCR3	14h

32.4.1.7.2 Diagram



32.4.1.7.3 Fields

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td>31-26
—</td><td>Reserved</td></tr>
<tr><td>25-24
CFR</td><td>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 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.
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.
<table><tr><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>SAI1_TCR3</td><td>—</td></tr><tr><td>—</td><td>SAI3_TCR3</td></tr></table></td></tr>
<tr><td>23-18
—</td><td>Reserved</td></tr>
<tr><td>17-16
TCE</td><td>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.
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>Field supported in</th><th>Field not supported in</th></tr><tr><td>SAI1_TCR3</td><td>—</td></tr><tr><td>SAI3_TCR3[16]</td><td>SAI3_TCR3[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>

32.4.1.8 SAI Transmit Configuration 4 Register (TCR4)

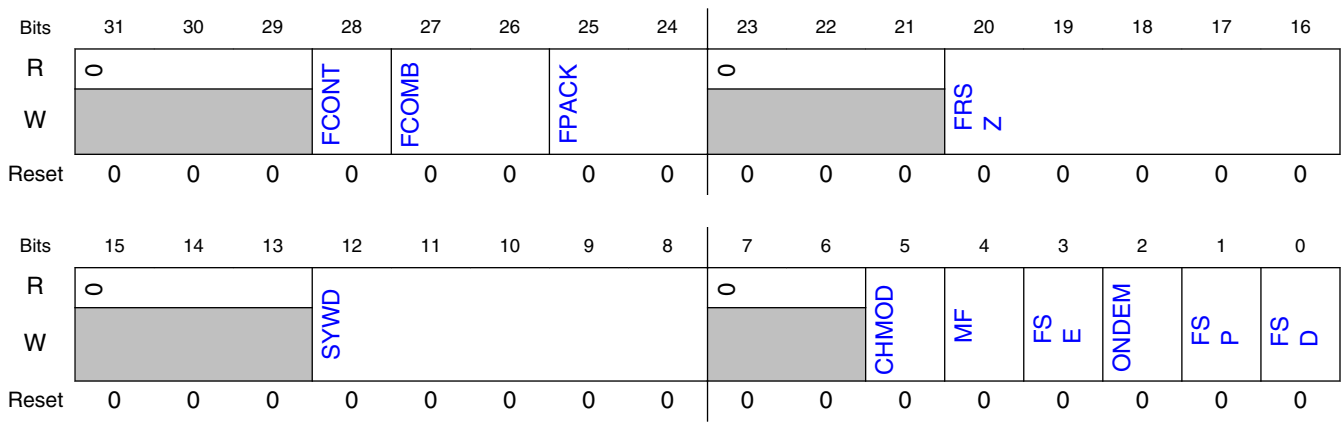
32.4.1.8.1 Offset

Register	Offset
TCR4	18h

32.4.1.8.2 Function

This register must not be altered when TCSR[TE] is set.

32.4.1.8.3 Diagram



32.4.1.8.4 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

Table continues on the next page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>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.

<table><tr><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>SAI1_TCR4</td><td>—</td></tr><tr><td>—</td><td>SAI3_TCR4</td></tr></table>
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).</td></tr>
<tr><td>25-24
FPACK</td><td>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</td></tr>
<tr><td>23-21
—</td><td>Reserved</td></tr>
<tr><td>20-16
FRSZ</td><td>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.</td></tr>
<tr><td>15-13
—</td><td>Reserved</td></tr>
<tr><td>12-8
SYWD</td><td>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.</td></tr>
<tr><td>7-6
—</td><td>Reserved</td></tr>
</table>

Table continues on the next page...

Field	Function
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 FSP	Frame Sync Polarity 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.

32.4.1.9 SAI Transmit Configuration 5 Register (TCR5)

32.4.1.9.1 Offset

Register	Offset
TCR5	1Ch

32.4.1.9.2 Function

This register must not be altered when TCSR[TE] is set.

32.4.1.9.3 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

32.4.1.9.4 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

32.4.1.10 SAI Transmit Data Register (TDR0 - TDR1)

32.4.1.10.1 Offset

Register	Offset
TDR0	20h
TDR1	24h

NOTE

Each module instance supports a different number of registers.

Register supported	Register not supported
SAI1_T DR0– TDR1	—
SAI3_T DR0	SAI3_T DR1

32.4.1.10.2 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.															

32.4.1.10.3 Register reset values

Register	Reset value
TDR0	SAI1,SAI3: 0000_0000h
TDR1	0000_0000h

32.4.1.10.4 Fields

Field	Function
31-0 TDR	Transmit Data Register 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.

32.4.1.11 SAI Transmit FIFO Register (TFR0 - TFR1)

32.4.1.11.1 Offset

Register	Offset
TFR0	40h
TFR1	44h

32.4.1.11.2 Function

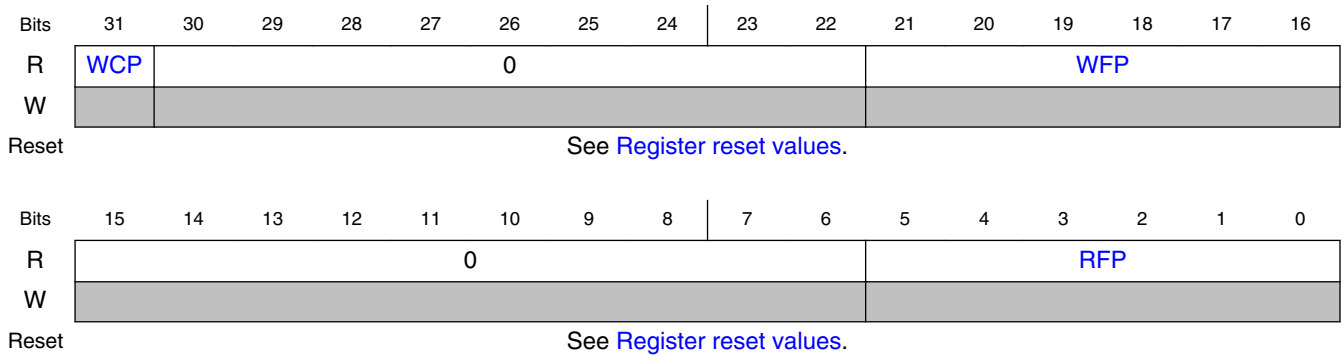
The MSB of the read and write pointers is used to distinguish between FIFO full and empty conditions. If the read and write pointers are identical, then the FIFO is empty. If the read and write pointers are identical except for the MSB, then the FIFO is full.

NOTE

Each module instance supports a different number of registers.

Register supported	Register not supported
SAI1_T FR0– TFR1	—
SAI3_T FR0	SAI3_T FR1

32.4.1.11.3 Diagram



32.4.1.11.4 Register reset values

Register	Reset value
TFR0	SAI1,SAI3: 0000_0000h
TFR1	0000_0000h

32.4.1.11.5 Fields

Field	Function
31 WCP	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. **Field supported in** / **Field not supported in**: SAI1_TFR0–TFR1 / —; — / SAI3_TFR0 0b - No effect. 1b - FIFO combine is enabled for FIFO writes and this FIFO will be written on the next FIFO write.
30-22 —	Reserved
21-16 WFP	Write FIFO Pointer FIFO write pointer for transmit data channel.
15-6 —	Reserved

Table continues on the next page...

Field	Function
5-0 RFP	Read FIFO Pointer FIFO read pointer for transmit data channel.

32.4.1.12 SAI Transmit Mask Register (TMR)

32.4.1.12.1 Offset

Register	Offset
TMR	60h

32.4.1.12.2 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.

32.4.1.12.3 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

32.4.1.12.4 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.

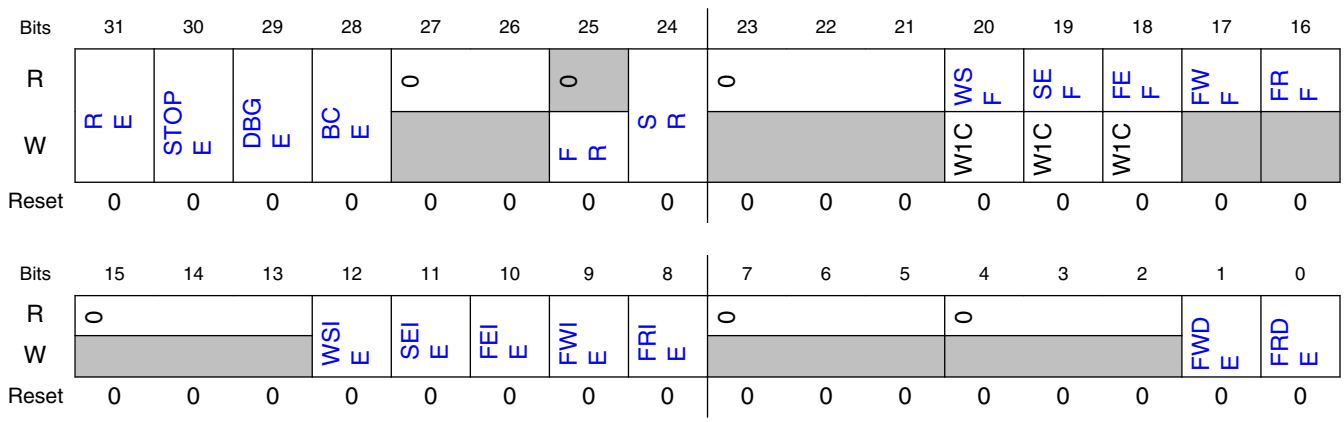
Field	Function
	00000000000000000000000000000000b - Word N is enabled. 00000000000000000000000000000001b - Word N is masked. The transmit data pins are tri-stated or drive zero when masked.

32.4.1.13 SAI Receive Control Register (RCSR)

32.4.1.13.1 Offset

Register	Offset
RCSR	88h

32.4.1.13.2 Diagram



32.4.1.13.3 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	Debug Enable

Table continues on the next page...

Field	Function
DBGE	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 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 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

Table continues on the next page...

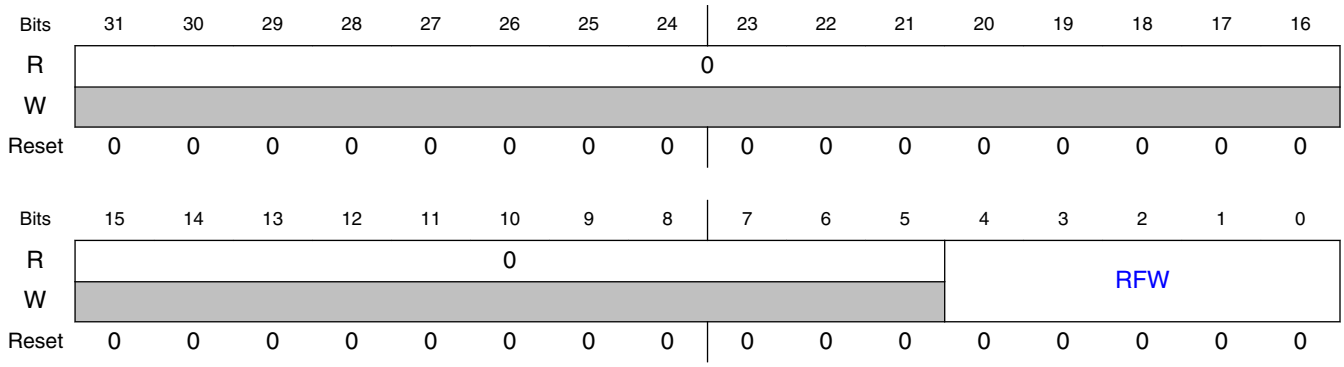
Field	Function
—	
12 WSIE	Word Start Interrupt Enable Enables/disables word start interrupts. 0b - Disables interrupt. 1b - Enables interrupt.
11 SEIE	Sync Error Interrupt Enable Enables/disables sync error interrupts. 0b - Disables interrupt. 1b - Enables interrupt.
10 FEIE	FIFO Error Interrupt Enable 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.

32.4.1.14 SAI Receive Configuration 1 Register (RCR1)

32.4.1.14.1 Offset

Register	Offset
RCR1	8Ch

32.4.1.14.2 Diagram



32.4.1.14.3 Fields

Field	Function
31-5 —	Reserved
4-0 RFW	Receive FIFO Watermark Configures the watermark level for all enabled receiver channels.

32.4.1.15 SAI Receive Configuration 2 Register (RCR2)

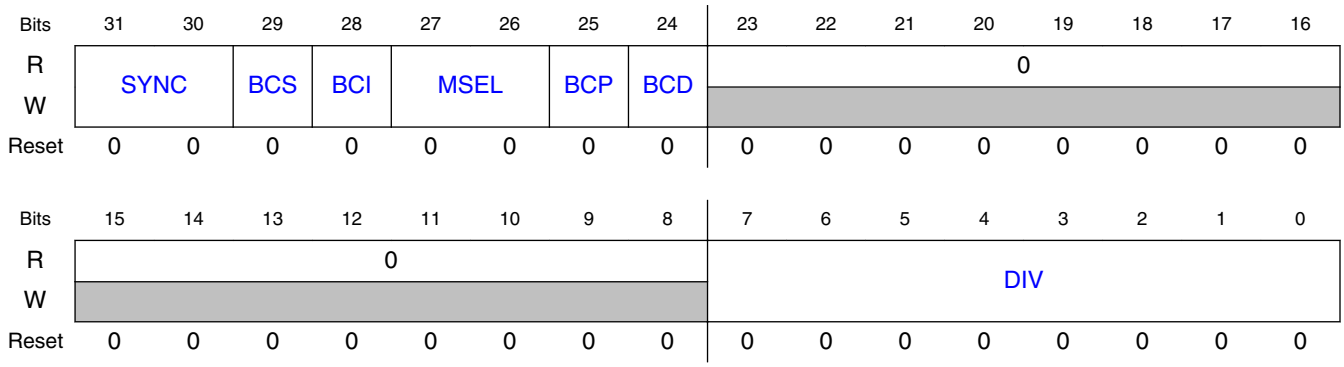
32.4.1.15.1 Offset

Register	Offset
RCR2	90h

32.4.1.15.2 Function

This register must not be altered when RCSR[RE] is set.

32.4.1.15.3 Diagram



32.4.1.15.4 Fields

Field	Function
31-30 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. 00b - Asynchronous mode. 01b - Synchronous with transmitter. 10b - Reserved. 11b - Reserved.
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 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.

Table continues on the next page...

Field	Function
	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.

32.4.1.16 SAI Receive Configuration 3 Register (RCR3)

32.4.1.16.1 Offset

Register	Offset
RCR3	94h

32.4.1.16.2 Diagram

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

32.4.1.16.3 Fields

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td>31-26
—</td><td>Reserved</td></tr>
<tr><td>25-24
CFR</td><td>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.
<table>
<tr><th>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>SAI1_RCR3</td><td>—</td></tr>
<tr><td>—</td><td>SAI3_RCR3</td></tr>
</table>
</td></tr>
<tr><td>23-18
—</td><td>Reserved</td></tr>
<tr><td>17-16
RCE</td><td>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.
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.
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>Field supported in</th><th>Field not supported in</th></tr>
<tr><td>SAI1_RCR3</td><td>—</td></tr>
<tr><td>SAI3_RCR3[16]</td><td>SAI3_RCR3[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>

32.4.1.17 SAI Receive Configuration 4 Register (RCR4)

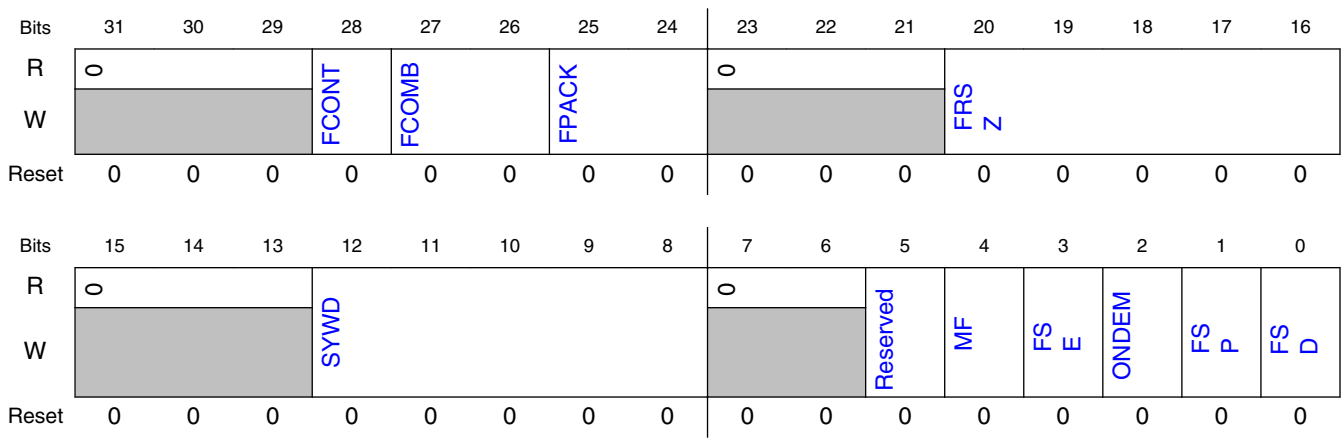
32.4.1.17.1 Offset

Register	Offset
RCR4	98h

32.4.1.17.2 Function

This register must not be altered when RCSR[RE] is set.

32.4.1.17.3 Diagram



32.4.1.17.4 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

Table continues on the next page...

<table>
<tr><th>Field</th><th>Function</th></tr>
<tr><td></td><td>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.

<table><tr><th>Field supported in</th><th>Field not supported in</th></tr><tr><td>SAI1_RCR4</td><td>—</td></tr><tr><td>—</td><td>SAI3_RCR4</td></tr></table>
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).</td></tr>
<tr><td>25-24
FPACK</td><td>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</td></tr>
<tr><td>23-21
—</td><td>Reserved</td></tr>
<tr><td>20-16
FRSZ</td><td>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.</td></tr>
<tr><td>15-13
—</td><td>Reserved</td></tr>
<tr><td>12-8
SYWD</td><td>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.</td></tr>
<tr><td>7-6
—</td><td>Reserved</td></tr>
</table>

Table continues on the next page...

Field	Function
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. 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.

32.4.1.18 SAI Receive Configuration 5 Register (RCR5)

32.4.1.18.1 Offset

Register	Offset
RCR5	9Ch

32.4.1.18.2 Function

This register must not be altered when RCSR[RE] is set.

32.4.1.18.3 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

32.4.1.18.4 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 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

32.4.1.19 SAI Receive Data Register (RDR0 - RDR1)

32.4.1.19.1 Offset

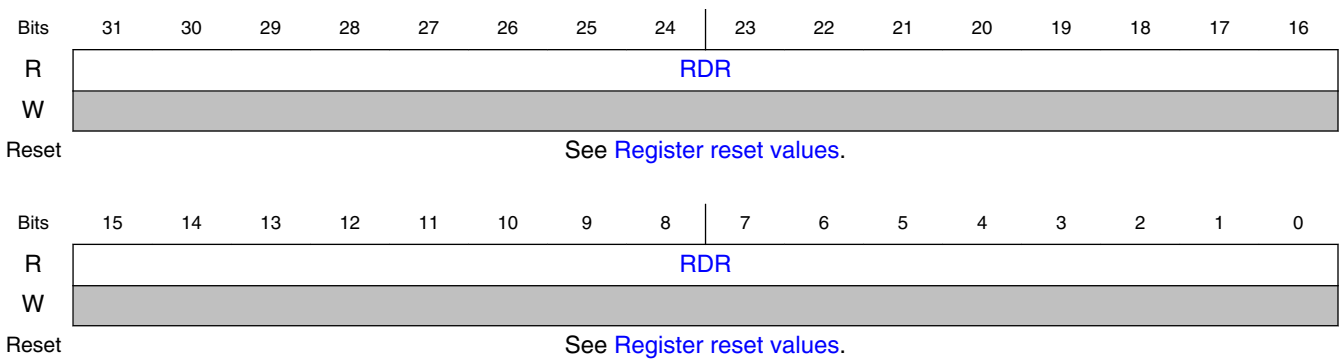
Register	Offset
RDR0	A0h
RDR1	A4h

NOTE

Each module instance supports a different number of registers.

Register supported	Register not supported
SAI1_R DR0– RDR1	—
SAI3_R DR0	SAI3_R DR1

32.4.1.19.2 Diagram



32.4.1.19.3 Register reset values

Register	Reset value
RDR0	SAI1,SAI3: 0000_0000h
RDR1	0000_0000h

32.4.1.19.4 Fields

Field	Function
31-0 RDR	Receive Data Register Reads from this register when the receive FIFO is not empty will return the data from the top of the receive FIFO. Reads from this register when the receive FIFO is empty are ignored.

32.4.1.20 SAI Receive FIFO Register (RFR0 - RFR1)

32.4.1.20.1 Offset

Register	Offset
RFR0	C0h
RFR1	C4h

32.4.1.20.2 Function

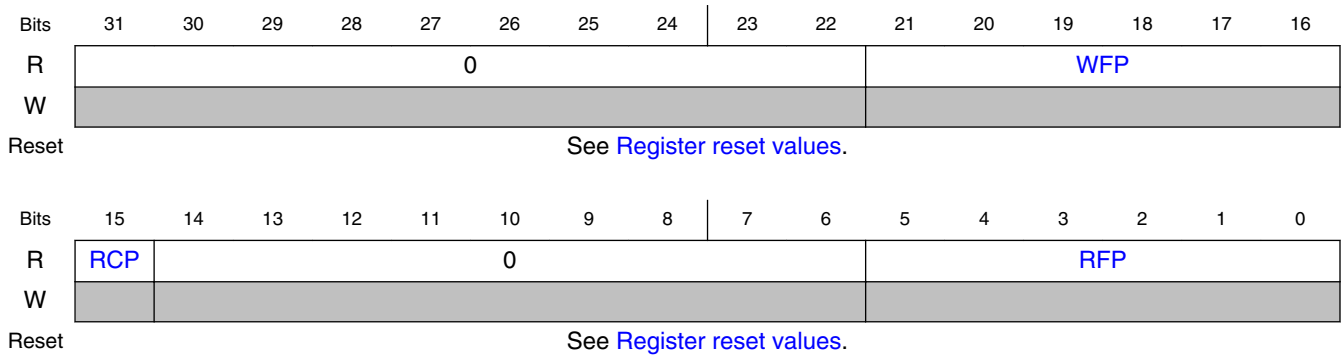
The MSB of the read and write pointers is used to distinguish between FIFO full and empty conditions. If the read and write pointers are identical, then the FIFO is empty. If the read and write pointers are identical except for the MSB, then the FIFO is full.

NOTE

Each module instance supports a different number of registers.

Register supported	Register not supported
SAI1_R FR0– RFR1	—
SAI3_R FR0	SAI3_R FR1

32.4.1.20.3 Diagram



32.4.1.20.4 Register reset values

Register	Reset value
RFR0	SAI1,SAI3: 0000_0000h
RFR1	0000_0000h

32.4.1.20.5 Fields

Field	Function
31-22 —	Reserved
21-16 WFP	Write FIFO Pointer FIFO write pointer for receive data channel.
15 RCP	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. **Field supported in** / **Field not supported in** SAI1_RFR0–RFR1 / — — / SAI3_RFR0 0b - No effect. 1b - FIFO combine is enabled for FIFO reads and this FIFO will be read on the next FIFO read.
14-6 —	Reserved

Table continues on the next page...

Field	Function
5-0 RFP	Read FIFO Pointer FIFO read pointer for receive data channel.

32.4.1.21 SAI Receive Mask Register (RMR)

32.4.1.21.1 Offset

Register	Offset
RMR	E0h

32.4.1.21.2 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.

32.4.1.21.3 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

32.4.1.21.4 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.

Field	Function
	00000000000000000000000000000000b - Word N is enabled. 00000000000000000000000000000001b - Word N is masked.

32.5 Functional description

This section provides a complete functional description of the block.

32.5.1 SAI clocking

The SAI clocks include:

- The audio master clock
- The bit clock
- The bus clock

32.5.1.1 Audio master clock

The audio master clock is used to generate 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.

32.5.1.2 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 is using 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.

32.5.1.3 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 specific 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.

32.5.2 SAI resets

The SAI is asynchronously reset on system reset. The SAI has a software reset and a FIFO reset.

32.5.2.1 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. It 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. It does not reset the configuration registers. The software reset remains asserted until cleared by software.

32.5.2.2 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 empties the FIFO contents and is to be 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.

32.5.3 Synchronous modes

The SAI transmitter and receiver can operate synchronously to each other.

32.5.3.1 Synchronous mode

The SAI transmitter and receiver can be configured to operate with synchronous bit clock and frame sync.

If the transmitter bit clock and frame sync are to be used by both the transmitter and receiver:

- The transmitter must be configured for asynchronous operation and the receiver for synchronous operation.
- In synchronous mode, the receiver is enabled only when both the transmitter and receiver are enabled.
- It is recommended that the transmitter is the last enabled and the first disabled.

If the receiver bit clock and frame sync are to be used by both the transmitter and receiver:

- The receiver must be configured for asynchronous operation and the transmitter for synchronous operation.
- In synchronous mode, the transmitter is enabled only when both the receiver and transmitter are both enabled.
- It is recommended that the receiver is the last enabled and the first disabled.

When operating in synchronous mode, only the bit clock, frame sync, and transmitter/receiver enable are shared. The transmitter and receiver otherwise operate independently, although configuration registers must be configured consistently across both the transmitter and receiver.

32.5.4 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 is used to indicate 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:

- Externally generated or internally generated
- Active high or active low
- Assert with the first bit in frame or asserts one bit early
- Assert for a duration between 1 bit clock and the first word length
- Frame length from 1 to 32 words per frame
- Word length to support 8 to 32 bits per word

  - First word length and remaining word lengths can be configured separately
- Words can be configured to transmit/receive MSB first or LSB first

These configuration options cannot be changed after the SAI transmitter or receiver is enabled.

32.5.5 Data FIFO

Each transmit and receive channel includes a FIFO of size 32 x 32-bit. The FIFO data is accessed using the SAI Transmit/Receive Data Registers.

32.5.5.1 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 32-2 for LSB First configurations and Figure 32-3 for MSB First configurations.

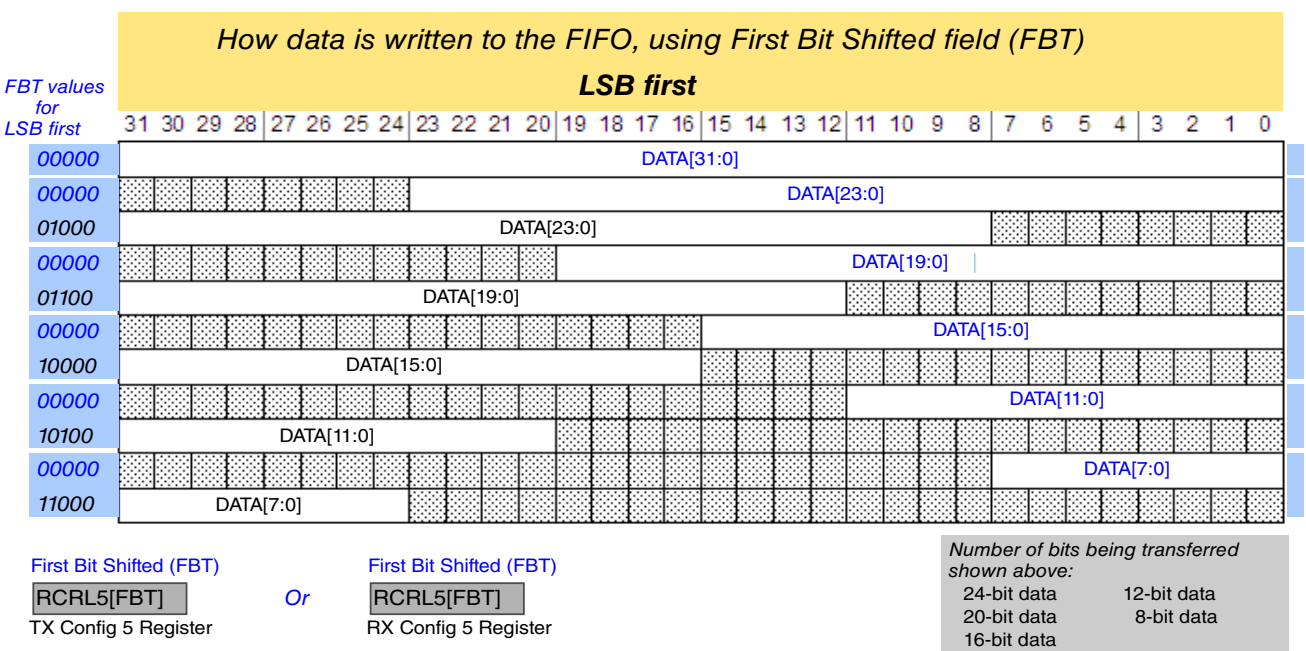


Figure 32-2. SAI first bit shifted, LSB first

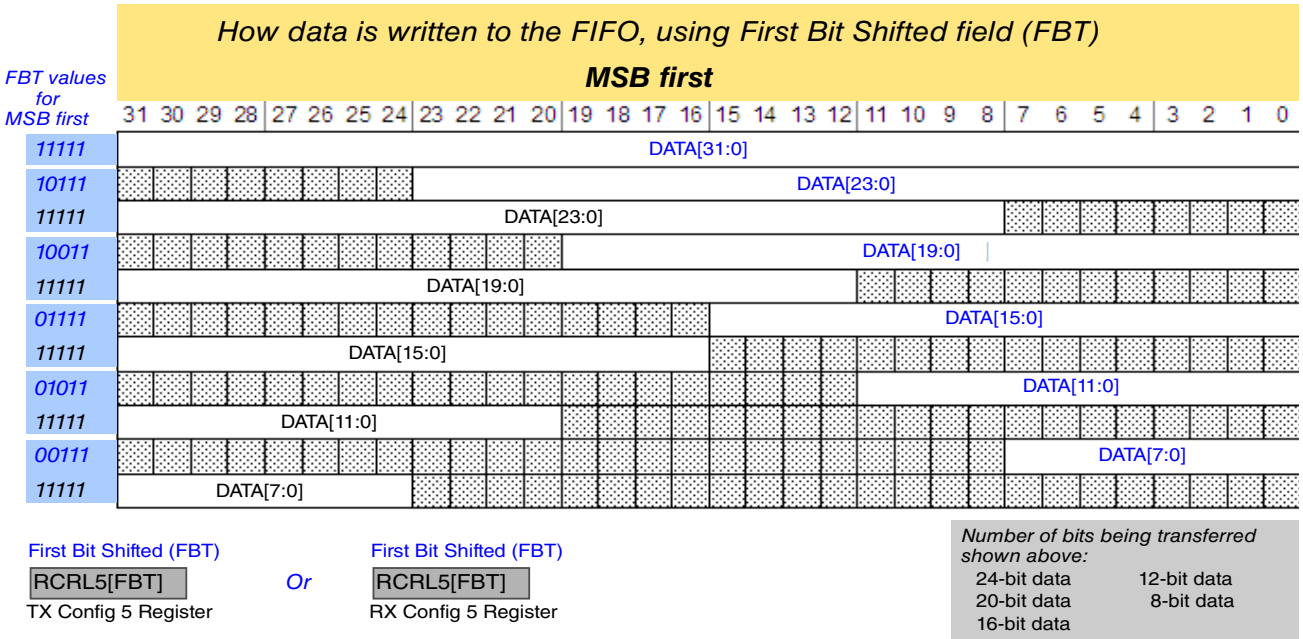


Figure 32-3. SAI first bit shifted, MSB first

32.5.5.2 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 will increment 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, then writes to a Transmit Data Register are ignored.
- If the Transmit FIFO is empty, then to avoid a FIFO underrun, the Transmit Data Register must be written at least 3 bit clocks before the start of the next unmasked word. Before enabling the transmitter, the Transmit FIFO should be initialized with data (since after the transmitter is enabled, the transmitter will start a new frame, and if no data is in the FIFO, then the transmitter will 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, then reads from a Receive Data Register are ignored.
- If the Receive FIFO is full, then to avoid a FIFO overrun, the Receive Data Register must be read at least 3 bit clocks before the end of an unmasked word.

32.5.5.3 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.

When 16-bit FIFO packing is enabled for transmit, the transmit shift register is loaded 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 $0 (first bit is selected by TCFG5[FBT] must be configured within this 16-bit word) and the second word transmitted is taken from the 16-bit word at byte offset $2 (first bit is selected by TCSR5[FBT][3:0]). The transmitter will transmit logic zero until the start of the next word once the 16-bit word has been transmitted.

When 16-bit FIFO packing is enabled 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 $0 (first bit is selected by RCFG5[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 $2 (first bit is selected by RCSR5[FBT][3:0]). The receiver will ignore 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 $0, second word in byte offset $1, third word in byte offset $2 and fourth word in byte offset $3. The TCFG5[FBT] and/or RCFG5[FBT] must be configured within byte offset $0.

32.5.5.4 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 FIFO Combine mode is enabled.

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 will access the first enabled channel FIFO, while the second access to a FIFO register will access 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 the 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 are loaded at the start of each frame and every N unmasked words (where N is the number of enabled data channels). The first word transmitted is loaded from the first enabled channel FIFO, while the second word transmitted is loaded from the second enabled channel FIFO, and so on until the end of the frame. Since the first word in each frame is always loaded 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 is stored to the first enabled channel FIFO, while the second word received is stored to the second enabled channel FIFO, and so on until the end of the frame. Since the first word in each frame is always stored 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 will load or store 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 will assert if any of the enabled data channel meets the warning flag or request flag conditions.

32.5.6 Word mask register

The SAI transmitter and receiver each contain a word mask register, namely 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.

32.5.7 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 are generated to wake up the CPU from stop mode.

32.5.7.1 FIFO request flag

The FIFO request flag is set based on the number of entries in the FIFO and the FIFO watermark configuration.

The transmit FIFO request flag is set when the number of entries in any of the enabled transmit FIFOs is less than or equal to the transmit FIFO watermark configuration and is cleared when the number of entries in each enabled transmit FIFO is greater than the transmit FIFO watermark configuration.

The receive FIFO request flag is set when the number of entries in any of the enabled receive FIFOs is greater than the receive FIFO watermark configuration and is cleared 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.

32.5.7.2 FIFO warning flag

The FIFO warning flag is set based on the number of entries in the FIFO.

The transmit warning flag is set when the number of entries in any of the enabled transmit FIFOs is empty and is cleared when the number of entries in each enabled transmit FIFO is not empty.

The receive warning flag is set when the number of entries in any of the enabled receive FIFOs is full and is cleared 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.

32.5.7.3 FIFO error flag

The transmit FIFO error flag is set when the any of the enabled transmit FIFOs underflow. After it is set, all enabled transmit channels will transmit zero data before TCSR[FEF] is cleared.

When TCR4[FCONT] is set, the FIFO will continue transmitting data following an underflow without software intervention. To ensure that data is transmitted in the correct order, the transmitter will continue from the same word number in the frame that caused the FIFO to underflow, but only after new data has been written to the transmit FIFO. Software should still clear the TCSR[FEF] flag, but without reinitializing the transmit FIFOs.

RCSR[FEF] is set when the any of the enabled receive FIFOs overflow. After it is set, all enabled receive channels discard received data until RCSR[FEF] is cleared and the next next receive frame starts. All enabled receive FIFOs should be emptied before RCSR[FEF] is cleared.

When RCR4[FCONT] is set, the FIFO will continue receiving data following an overflow without software intervention. To ensure that data is received in the correct order, the receiver will continue 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.

32.5.7.4 Sync error flag

The sync error flag, TCSR[SEF] or RCSR[SEF], is set 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 is set. When the sync error flag is set, the transmitter or receiver continues checking for frame sync assertion when idle or at the end of each frame.

The sync error flag can generate an interrupt only.

32.5.7.5 Word start flag

The word start flag is set at the start of the second bit clock for the selected word, as configured by the Word Flag register field.

The word start flag can generate an interrupt only.

Chapter 33
Medium Quality Sound (MQS)

33.1 Chip-specific MQS information

Table 33-1. 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

33.2 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.

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 44 kHz or 48 kHz input signals from SAI3 are in left_justified format. MQS provides the SNR target as no more than 20 dB 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.

33.2.1 Block Diagram

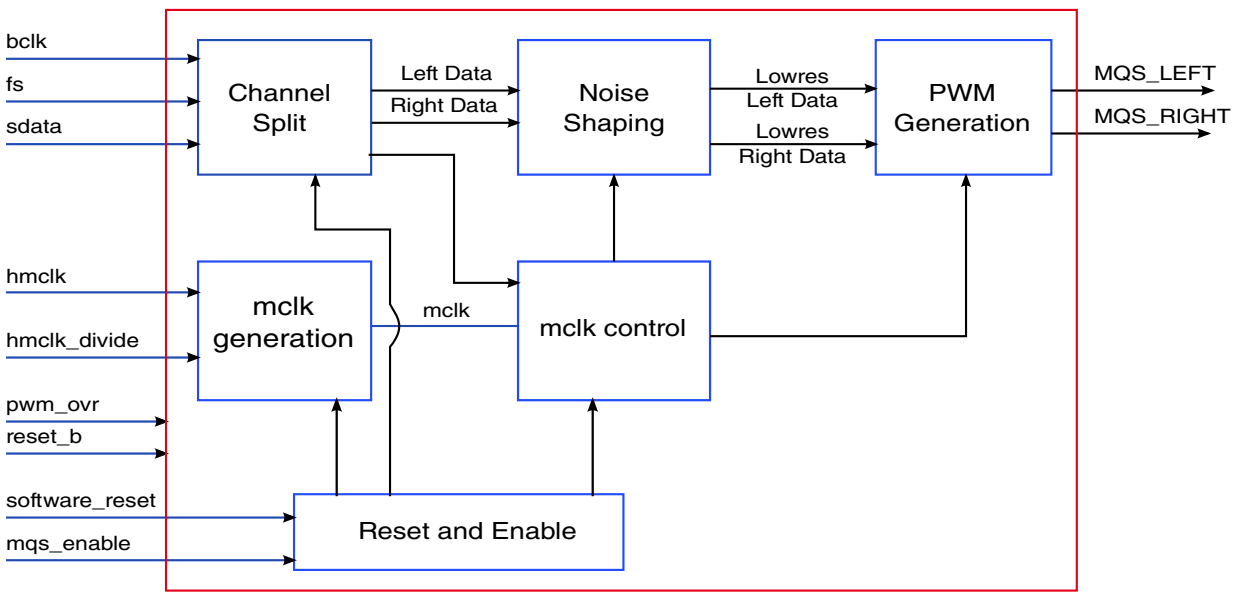


Figure 33-1. Block Diagram

MQS has the following sub-modules:

1. Channel Split: Splits the I2S signals into separate left channel and right channel audio data.
2. Noise Shaping: Uses the sigma-delta algorithm to generate low-resolution, very high sampling audio, while the audio sampling rate is increased.
3. 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.
4. mclk generation: Used to generate the master clock (mclk). The frequency of mclk is determined by the final bit duration of PWM generation module.
5. mclk control:Used as a metronome to co-ordinate the different functional blocks working synchronously.
6. Reset and Enable: Used to generate the reset and enable logic to different clock domains.

33.3 External Signals

The following table describes the external signals of MQS:

Table 33-2. MQS External Signals

Signal	Description	Pad	Mode	Direction
MQS_LEFT	Left signal output	GPIO_AD_B0_07	ALT1	O
		GPIO_EMC_17	ALT2	
MQS_RIGHT	Right signal output	GPIO_AD_B0_06	ALT1	O
		GPIO_EMC_16	ALT2	

Table 33-3. MQS External Signals

Signal	Description	Pad	Mode	Direction
MQS_LEFT	Left signal output	GPIO_AD_01	ALT4	O
MQS_RIGHT	Right signal output	GPIO_AD_02	ALT4	O

33.4 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 GPR
mqs_enable	In	1	module enable	From GPR
pwm_ovr	In	1	PWM oversampling ratio1—64, 0--32	From GPR
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 GPR
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	

33.5 Programming Considerations

MQS has no internal programmable registers. But it does have some programmability from IOMUXC_GPR2.

Register Bits	Name	Description
IOMUXC_GPR2[26]	MQS_OVERSAMPLE	Used to control the PWM oversampling rate compared with mclk. 1—64, 0—32.
IOMUXC_GPR2[25]	MQS_EN	MQS enable. 1—Enable MQS, 0—Disable MQS
IOMUXC_GPR2[24]	MQS_SW_RST	MQS software reset. 1—Enable software reset for MQS 0—Exit software reset for MQS
IOMUXC_GPR2[23:16]	MQS_CLK_DIV[7:0]	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

33.5.1 Usage Model

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 34
Sony/Philips Digital Interface (SPDIF)

34.1 Chip-specific SPDIF information

Table 34-1. 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

34.2 Overview

The Sony/Philips Digital Interface (SPDIF) audio block is a stereo transceiver that allows the processor to receive and transmit digital audio.

The SPDIF transceiver allows the handling of both SPDIF channel status (CS) and User (U) data and includes a frequency measurement block that allows the precise measurement of an incoming sampling frequency.

A recovered clock is provided 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.

As the SPDIF internal data width is 24-bit, the eight most-significant bits of all registers return zeros.

The figure below shows a block diagram of the SPDIF transceiver data paths (receiver and transmitter) and its interface.

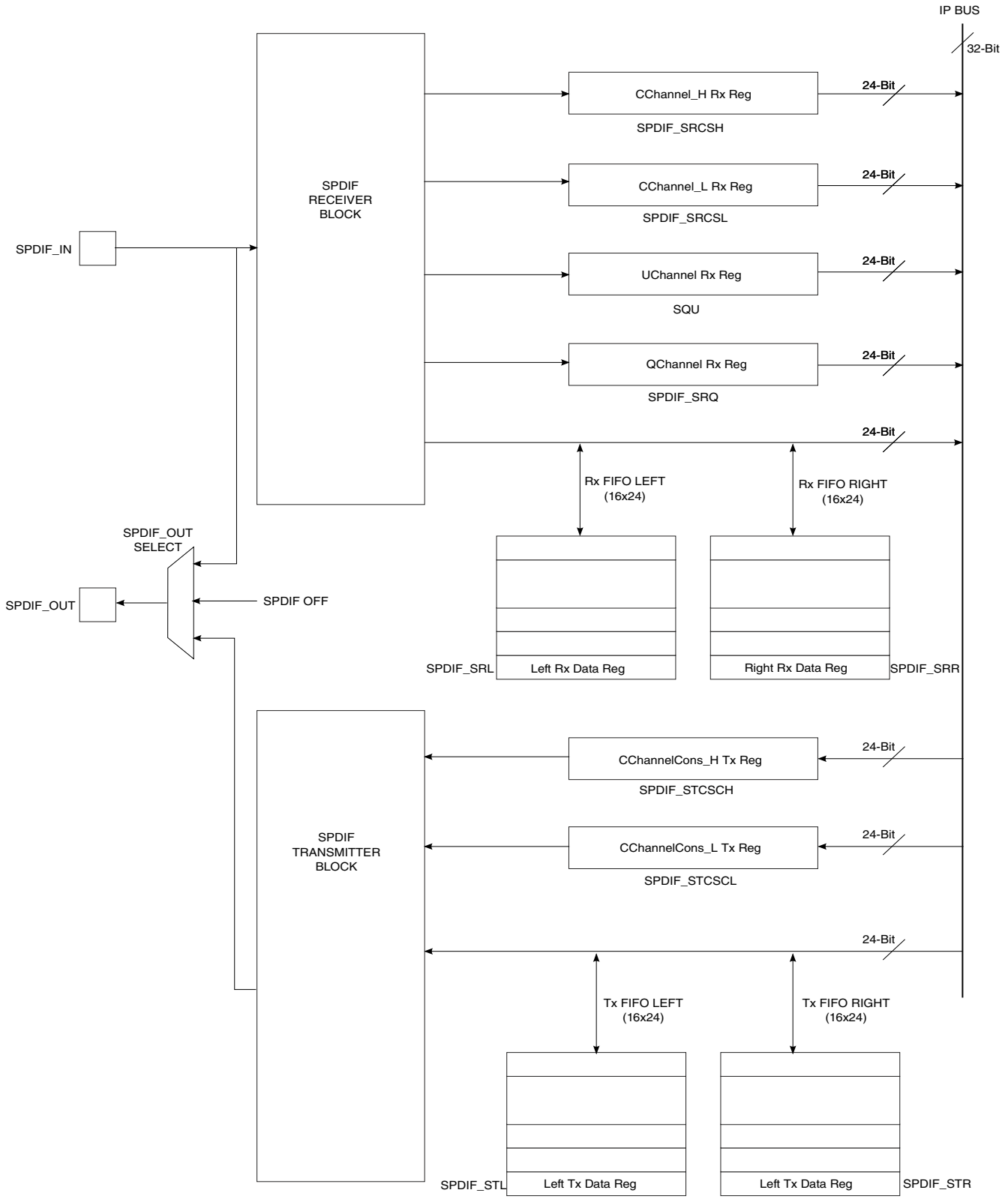


Figure 34-1. SPDIF Transceiver Data Interface Block Diagram

34.3 External Signals

The following table describes the external signals of SPDIF:

Table 34-2. SPDIF External Signals

Signal	Description	Pad	Mode	Direction
SPDIF_EXT_CLK	External clock signal	GPIO_06	ALT4	I
		GPIO_12	ALT6	
SPDIF_IN	Input line	GPIO_04	ALT4	I
		GPIO_10	ALT6	
SPDIF_LOCK	Lock signal	GPIO_07	ALT4	O
		GPIO_13	ALT6	
SPDIF_OUT	Output line signal	GPIO_05	ALT4	O
		GPIO_11	ALT6	
SPDIF_SR_CLK	SR clock signal	GPIO_03	ALT4	O
		GPIO_09	ALT6	

34.4 Clocks

The table found here describes the clock sources for SPDIF.

Please see clock control block for clock setting, configuration and gating information.

Table 34-3. SPDIF Clocks

Clock name	Clock Root	Description
gclkw_t0	ipg_clk_root	Global clock
ipg_clk_s	ipg_clk_root	Peripheral access clock
tx_clk	spdif0_clk_root	Module Tx clock

34.5 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 1 shows the clock structure of the SPDIF transceiver.

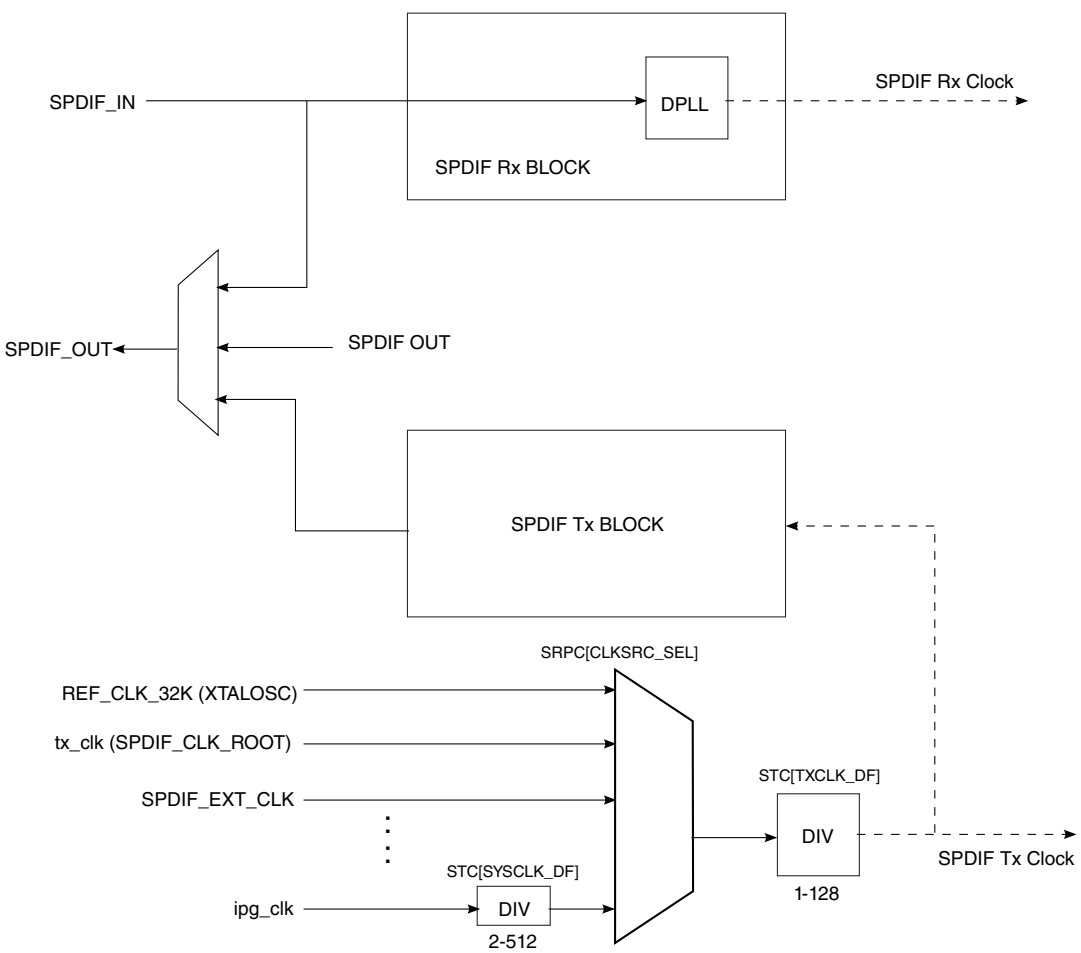


Figure 34-2. SPDIF Transceiver Clock Diagram

34.5.1 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.

The following functions are performed by the SPDIF receiver:

- Audio Data Reception see Audio Data Reception
- Channel Status bits Reception see Channel Status Reception
- U Channel bits Reception see User Bit Reception
- Validity Flag Reception see Validity Flag Reception
- SPDIF Receiver Exception support see SPDIF Receiver
- SPDIF Lock Detection

34.5.1.1 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.

- **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.

- **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.

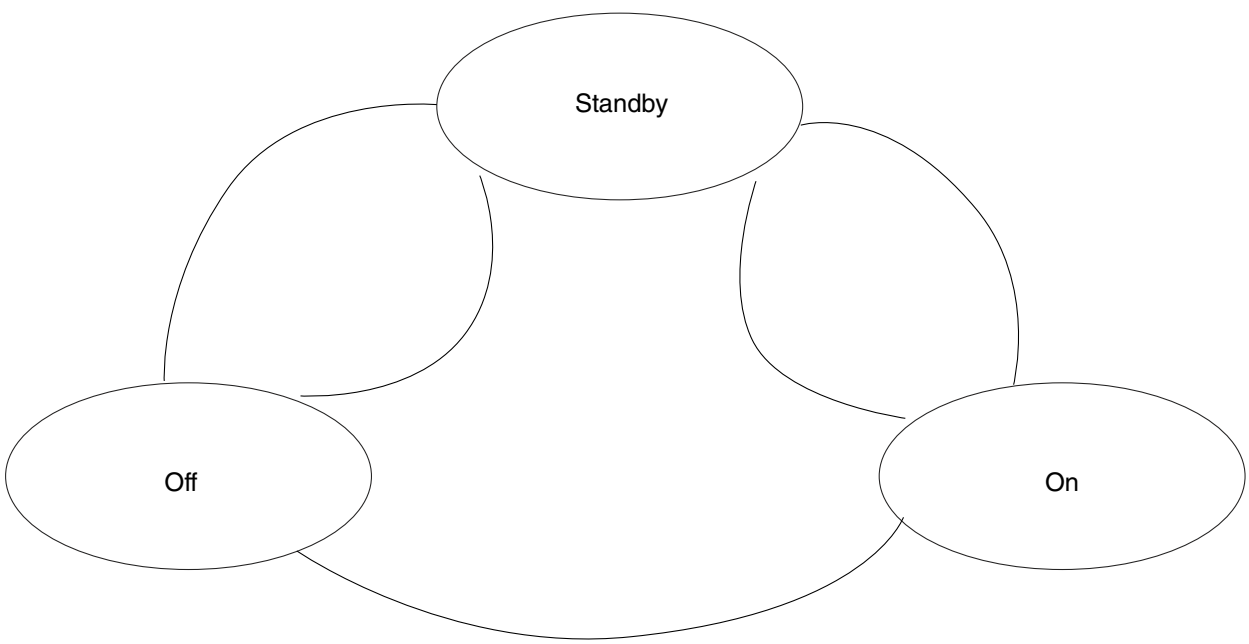


Figure 34-3. FIFO Auto-resync Controller State Machine

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 SPDIFConfig register to enable/disable the feature for the SPDIF Rx FIFO and SPDIF Tx FIFO.

34.5.1.1.1 Application Note

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 FIFO s 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.

- **SPDIF receiver - Additional features**

There are three exceptions associated with the SPDIF Receivers 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.

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.

34.5.1.2 Channel Status Reception

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.

34.5.1.2.1 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.

34.5.1.3 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).

- **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 34-4. 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.

- **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.

34.5.1.4 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".

34.5.1.5 SPDIF Receiver Interrupt Exception Definition

Several SPDIF exceptions can trigger an interrupt.

They are:

- Control Status channel change. Set when SPDIFRxCChannel_l 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.

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.

34.5.1.6 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 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.

34.5.1.7 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.

34.5.1.8 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}.

34.5.2 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 TxClk_Source for 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.

34.5.2.1 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).

- **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.

- **SPDIF transmitter data registers - Automatic resynchronization of FIFOs**

See Audio Data Reception.

- **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.

34.5.2.2 Channel Status Transmission

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.

34.5.2.3 Validity Flag Transmission

The validity bit setting is performed via bit 5 of the SPDIF_SCR register.

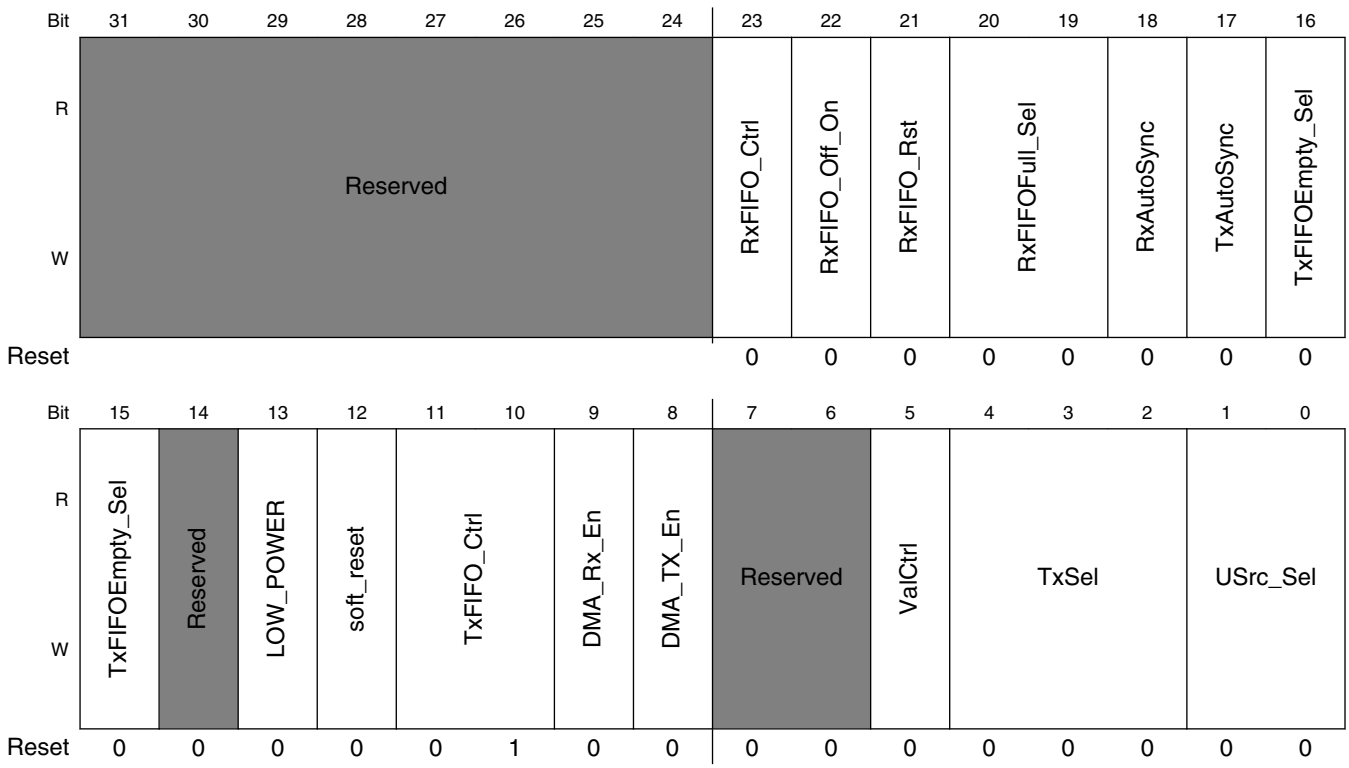
34.6 SPDIF Memory Map/Register Definition

SPDIF memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
401D_C000	SPDIF Configuration Register (SPDIF_SCR)	32	R/W	0000_0400h	34.6.1/1107
401D_C004	CDText Control Register (SPDIF_SRCD)	32	R/W	0000_0000h	34.6.2/1109
401D_C008	PhaseConfig Register (SPDIF_SRPC)	32	R/W	0000_0000h	34.6.3/1110
401D_C00C	InterruptEn Register (SPDIF_SIE)	32	R/W	0000_0000h	34.6.4/1111
401D_C010	InterruptStat Register (SPDIF_SIS)	32	R	0000_0002h	34.6.5/1113
401D_C010	InterruptClear Register (SPDIF_SIC)	32	W	0000_0000h	34.6.6/1115
401D_C014	SPDIFRxLeft Register (SPDIF_SRL)	32	R	0000_0000h	34.6.7/1116
401D_C018	SPDIFRxRight Register (SPDIF_SRR)	32	R	0000_0000h	34.6.8/1117
401D_C01C	SPDIFRxCChannel_h Register (SPDIF_SRCSH)	32	R	0000_0000h	34.6.9/1117
401D_C020	SPDIFRxCChannel_l Register (SPDIF_SRCSL)	32	R	0000_0000h	34.6.10/ 1118
401D_C024	UchannelRx Register (SPDIF_SRU)	32	R	0000_0000h	34.6.11/ 1118
401D_C028	QchannelRx Register (SPDIF_SRQ)	32	R	0000_0000h	34.6.12/ 1119
401D_C02C	SPDIFTxLeft Register (SPDIF_STL)	32	W	0000_0000h	34.6.13/ 1119
401D_C030	SPDIFTxRight Register (SPDIF_STR)	32	W	0000_0000h	34.6.14/ 1120
401D_C034	SPDIFTxCChannelCons_h Register (SPDIF_STCSCH)	32	R/W	0000_0000h	34.6.15/ 1120
401D_C038	SPDIFTxCChannelCons_l Register (SPDIF_STCSCL)	32	R/W	0000_0000h	34.6.16/ 1121
401D_C044	FreqMeas Register (SPDIF_SRFM)	32	R	0000_0000h	34.6.17/ 1121
401D_C050	SPDIFTxClk Register (SPDIF_STC)	32	R/W	0002_0F00h	34.6.18/ 1122

34.6.1 SPDIF Configuration Register (SPDIF_SCR)

Address: 401D_C000h base + 0h offset = 401D_C000h



SPDIF_SCR field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
23 RxFIFO_Ctrl	0 Normal operation 1 Always read zero from Rx data register
22 RxFIFO_Off_On	0 SPDIF Rx FIFO is on 1 SPDIF Rx FIFO is off. Does not accept data from interface
21 RxFIFO_Rst	0 Normal operation 1 Reset register to 1 sample remaining
20–19 RxFIFOFull_Sel	00 Full interrupt if at least 1 sample in Rx left and right FIFOs 01 Full interrupt if at least 4 sample in Rx left and right FIFOs 10 Full interrupt if at least 8 sample in Rx left and right FIFOs 11 Full interrupt if at least 16 sample in Rx left and right FIFO
18 RxAutoSync	0 Rx FIFO auto sync off 1 RxFIFO auto sync on
17 TxAutoSync	0 Tx FIFO auto sync off 1 Tx FIFO auto sync on

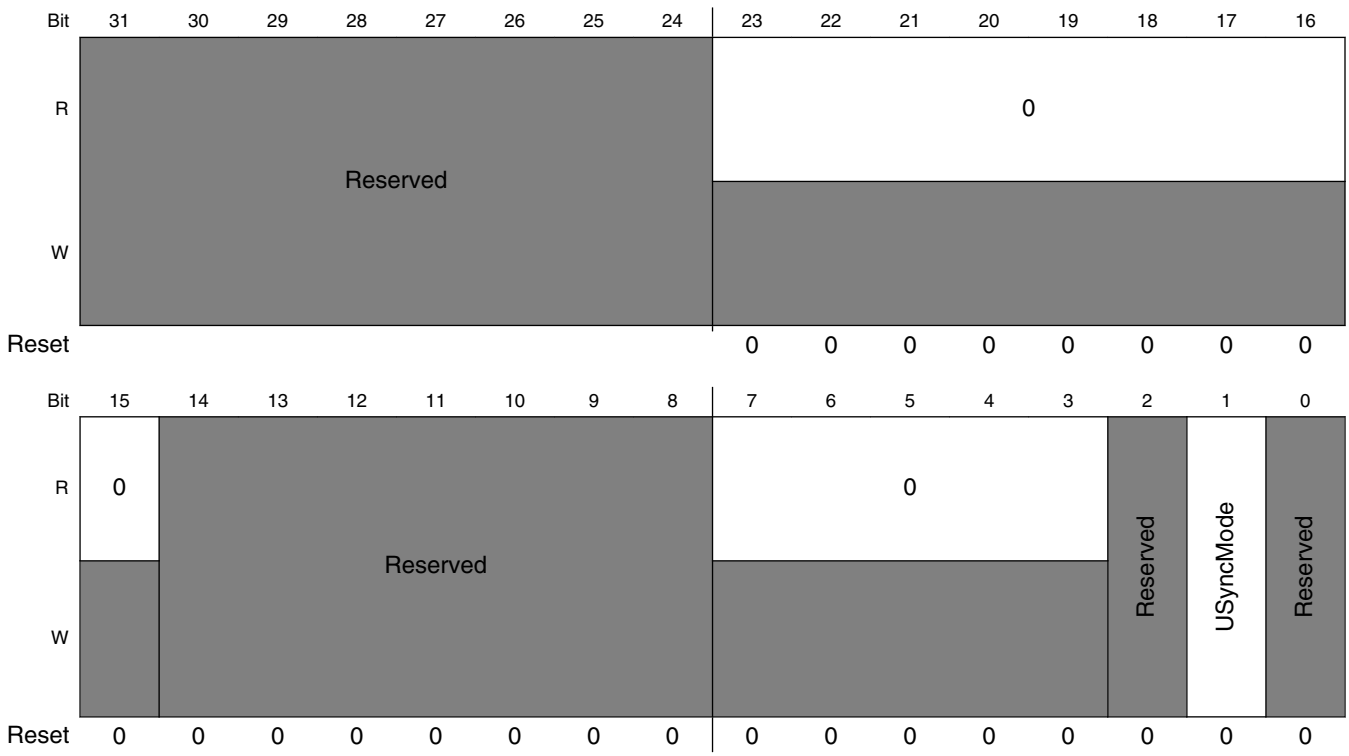
Table continues on the next page...

SPDIF_SCR field descriptions (continued)

Field	Description
16–15 TxFIFOEmpty_Sel	00 Empty interrupt if 0 sample in Tx left and right FIFOs 01 Empty interrupt if at most 4 sample in Tx left and right FIFOs 10 Empty interrupt if at most 8 sample in Tx left and right FIFOs 11 Empty interrupt if at most 12 sample in Tx left and right FIFOs
14 -	This field is reserved. Reserved
13 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	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	00 Send out digital zero on SPDIF Tx 01 Tx Normal operation 10 Reset to 1 sample remaining 11 Reserved
9 DMA_Rx_En	DMA Receive Request Enable (RX FIFO full)
8 DMA_TX_En	DMA Transmit Request Enable (Tx FIFO empty)
7–6 -	This field is reserved. Reserved
5 ValCtrl	0 Outgoing Validity always set 1 Outgoing Validity always clear
4–2 TxSel	000 Off and output 0 001 Feed-through SPDIFIN 101 Tx Normal operation Others Reserved
USrc_Sel	00 No embedded U channel 01 U channel from SPDIF receive block (CD mode) 10 Reserved 11 U channel from on chip transmitter

34.6.2 CDText Control Register (SPDIF_SRCD)

Address: 401D_C000h base + 4h offset = 401D_C004h

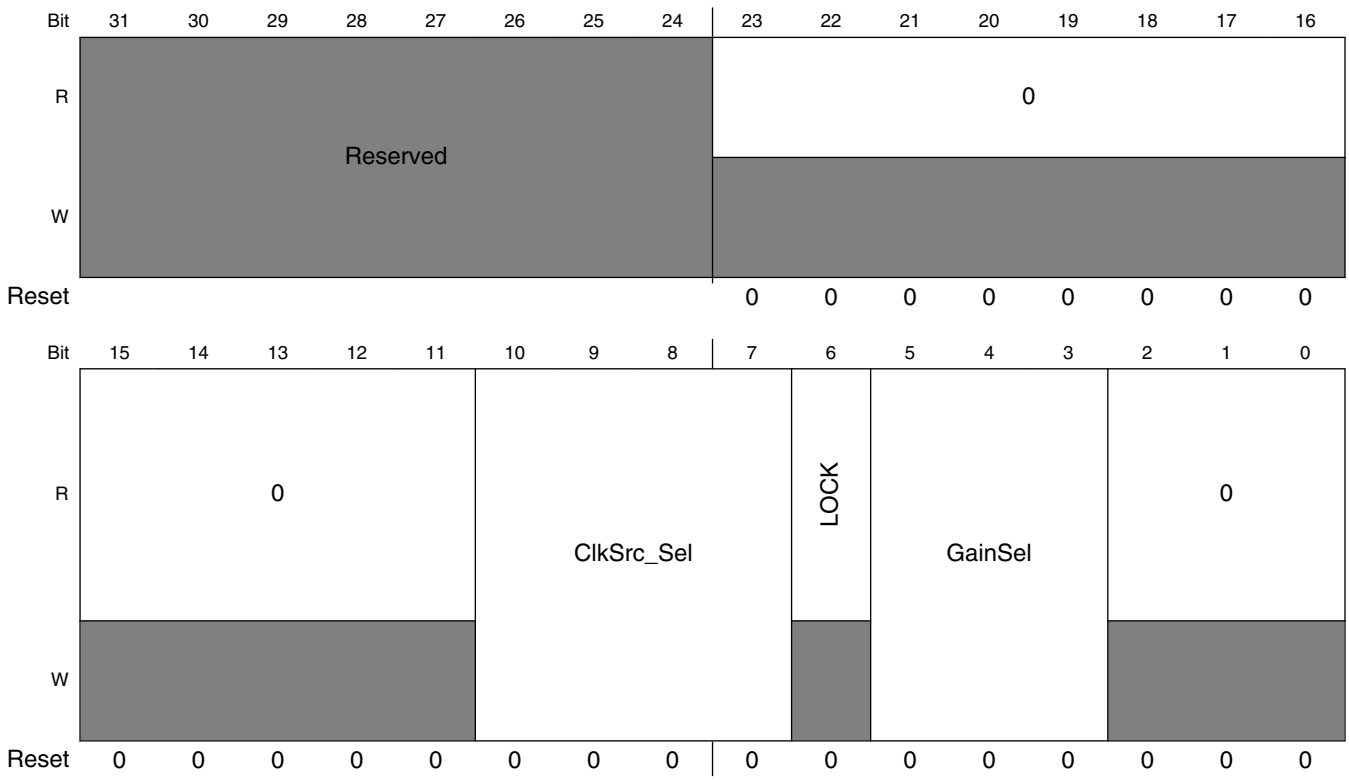


SPDIF_SRCD field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
23–15 Reserved	This read-only field is reserved and always has the value 0.
14–8 -	This field is reserved. Reserved. set to zero.
7–3 Reserved	This read-only field is reserved and always has the value 0.
2 -	This field is reserved. Reserved.
1 USyncMode	0 Non-CD data 1 CD user channel subcode
0 -	This field is reserved. Reserved.

34.6.3 PhaseConfig Register (SPDIF_SRPC)

Address: 401D_C000h base + 8h offset = 401D_C008h



SPDIF_SRPC field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
23–11 Reserved	This read-only field is reserved and always has the value 0.
10–7 ClkSrc_Sel	Clock source selection, all other settings not shown are reserved: 0000 if (DPLL Locked) SPDIF_RxClk else REF_CLK_32K (XTALOSC) 0001 if (DPLL Locked) SPDIF_RxClk else tx_clk (SPDIF0_CLK_ROOT) 0011 if (DPLL Locked) SPDIF_RxClk else SPDIF_EXT_CLK 0101 REF_CLK_32K (XTALOSC) 0110 tx_clk (SPDIF0_CLK_ROOT) 1000 SPDIF_EXT_CLK
6 LOCK	LOCK bit to show that the internal DPLL is locked, read only
5–3 GainSel	Gain selection: 000 24*(2**10) 001 16*(2**10)

Table continues on the next page...

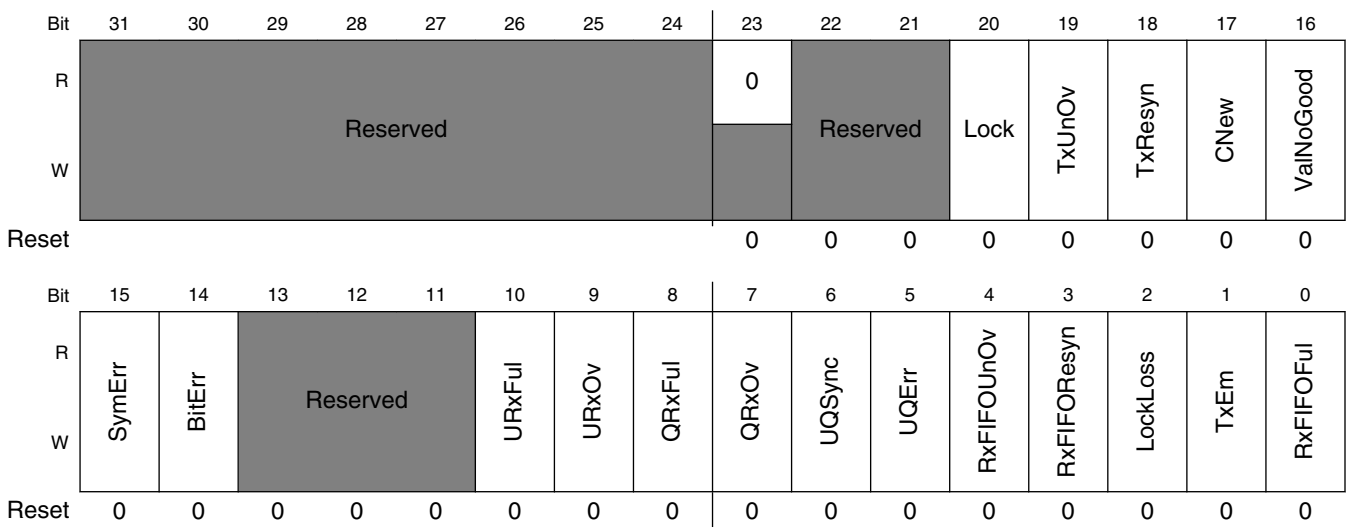
SPDIF_SRPC field descriptions (continued)

Field	Description
	010 12*(2**10) 011 8*(2**10) 100 6*(2**10) 101 4*(2**10) 110 3*(2**10)
Reserved	This read-only field is reserved and always has the value 0.

34.6.4 InterruptEn Register (SPDIF_SIE)

The InterruptEn register (SPDIF_SIE) provides control over the enabling of interrupts.

Address: 401D_C000h base + Ch offset = 401D_C00Ch



SPDIF_SIE field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
23 Reserved	This read-only field is reserved and always has the value 0.
22–21 -	This field is reserved. Reserved. set to zero.
20 Lock	SPDIF receiver's DPLL is locked
19 TxUnOv	SPDIF Tx FIFO under/overrun

Table continues on the next page...

SPDIF_SIE field descriptions (continued)

Field	Description
18 TxResyn	SPDIF Tx FIFO resync
17 CNew	SPDIF receive change in value of control channel
16 ValNoGood	SPDIF validity flag no good
15 SymErr	SPDIF receiver found illegal symbol
14 BitErr	SPDIF receiver found parity bit error
13–11 -	This field is reserved. Reserved. set to zero.
10 URxFul	U Channel receive register full, can't be cleared with reg. IntClear. To clear it, read from U Rx reg.
9 URxOv	U Channel receive register overrun
8 QRxFul	Q Channel receive register full, can't be cleared with reg. IntClear. To clear it, read from Q Rx reg.
7 QRxOv	Q Channel receive register overrun
6 UQSync	U/Q Channel sync found
5 UQErr	U/Q Channel framing error
4 RxFIFOUnOv	Rx FIFO underrun/overrun
3 RxFIFOResyn	Rx FIFO resync
2 LockLoss	SPDIF receiver loss of lock
1 TxEm	SPDIF Tx FIFO empty, can't be cleared with reg. IntClear. To clear it, write toTx FIFO.
0 RxFIFOFul	SPDIF Rx FIFO full, can't be cleared with reg. IntClear. To clear it, read from Rx FIFO.

34.6.5 InterruptStat Register (SPDIF_SIS)

The InterruptStat (SPDIF_SIS) register is a read only register that provides the status on interrupt operations.

Address: 401D_C000h base + 10h offset = 401D_C010h

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved								0			Lock	TxUnOv	TxResyn	CNew	ValNoGood
W																
Reset									0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	SymErr	BitErr	0			URxFul	URxOv	QRxFul	QRxOv	UQSync	UQErr	RxFIFOUnOv	RxFIFOResyn	LockLoss	TxEm	RxFIFOFul
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

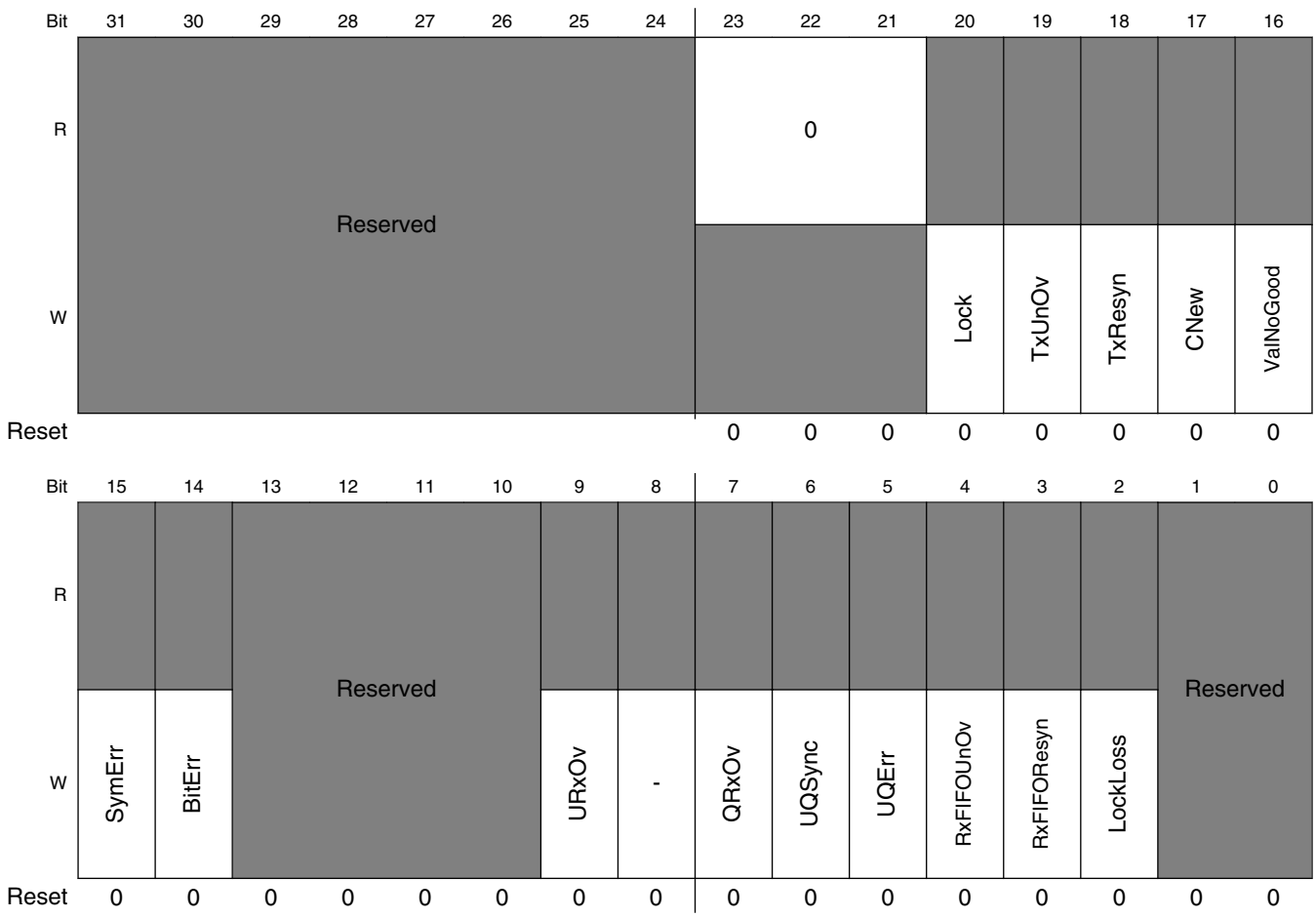
SPDIF_SIS field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
23–21 Reserved	This read-only field is reserved and always has the value 0.
20 Lock	SPDIF receiver's DPLL is locked
19 TxUnOv	SPDIF Tx FIFO under/overrun
18 TxResyn	SPDIF Tx FIFO resync
17 CNew	SPDIF receive change in value of control channel
16 ValNoGood	SPDIF validity flag no good
15 SymErr	SPDIF receiver found illegal symbol
14 BitErr	SPDIF receiver found parity bit error
13–11 Reserved	This read-only field is reserved and always has the value 0.
10 URxFul	U Channel receive register full, can't be cleared with reg. IntClear. To clear it, read from U Rx reg.
9 URxOv	U Channel receive register overrun
8 QRxFul	Q Channel receive register full, can't be cleared with reg. IntClear. To clear it, read from Q Rx reg.
7 QRxOv	Q Channel receive register overrun
6 UQSync	U/Q Channel sync found
5 UQErr	U/Q Channel framing error
4 RxFIFOUnOv	Rx FIFO underrun/overrun
3 RxFIFOResyn	Rx FIFO resync
2 LockLoss	SPDIF receiver loss of lock
1 TxEm	SPDIF Tx FIFO empty, can't be cleared with reg. IntClear. To clear it, write toTx FIFO.
0 RxFIFOFul	SPDIF Rx FIFO full, can't be cleared with reg. IntClear. To clear it, read from Rx FIFO.

34.6.6 InterruptClear Register (SPDIF_SIC)

The InterruptClear (SPDIF_SIC) register is a write only register and is used to clear interrupts.

Address: 401D_C000h base + 10h offset = 401D_C010h



SPDIF_SIC field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
23–21 Reserved	This read-only field is reserved and always has the value 0.
20 Lock	SPDIF receiver's DPLL is locked
19 TxUnOv	SPDIF Tx FIFO under/overrun

Table continues on the next page...

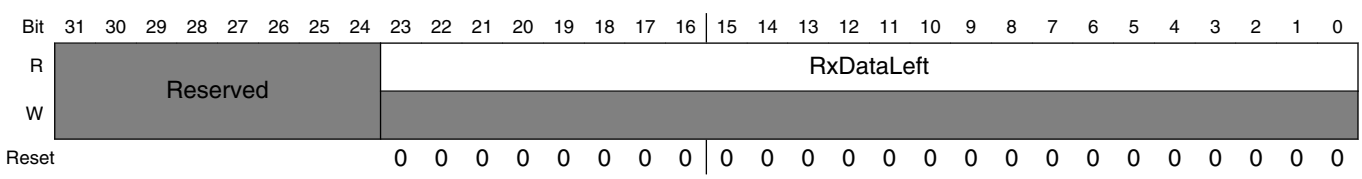
SPDIF_SIC field descriptions (continued)

Field	Description
18 TxResyn	SPDIF Tx FIFO resync
17 CNew	SPDIF receive change in value of control channel
16 ValNoGood	SPDIF validity flag no good
15 SymErr	SPDIF receiver found illegal symbol
14 BitErr	SPDIF receiver found parity bit error
13–10 -	This field is reserved. Reserved.
9 URxOv	U Channel receive register overrun
8 -	Reserved
7 QRxOv	Q Channel receive register overrun
6 UQSync	U/Q Channel sync found
5 UQErr	U/Q Channel framing error
4 RxFIFOUnOv	Rx FIFO underrun/overrun
3 RxFIFOResyn	Rx FIFO resync
2 LockLoss	SPDIF receiver loss of lock
-	This field is reserved. Reserved.

34.6.7 SPDIFRxLeft Register (SPDIF_SRL)

SPDIFRxLeft register is an audio data reception register.

Address: 401D_C000h base + 14h offset = 401D_C014h



SPDIF_SRL field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
RxDataLeft	Processor receive SPDIF data left

34.6.8 SPDIFRxRight Register (SPDIF_SRR)

SPDIFRxRight register is an audio data reception register.

Address: 401D_C000h base + 18h offset = 401D_C018h

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								RxDataRight																							
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

SPDIF_SRR field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
RxDataRight	Processor receive SPDIF data right

34.6.9 SPDIFRxCChannel_h Register (SPDIF_SRCSH)

SPDIFRxCChannel_h register is a channel status reception register.

Address: 401D_C000h base + 1Ch offset = 401D_C01Ch

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								RxCChannel_h																							
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

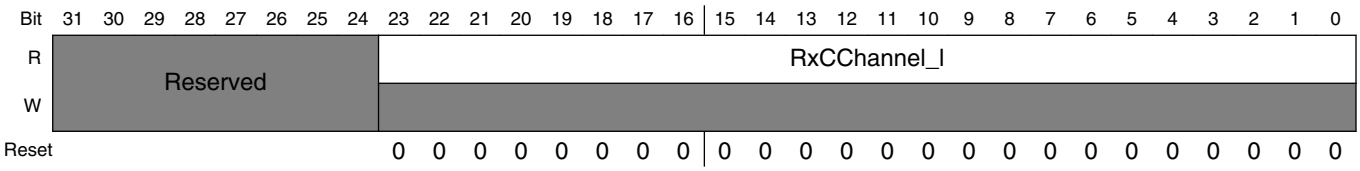
SPDIF_SRCSH field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
RxCChannel_h	SPDIF receive C channel register, contains first 24 bits of C channel without interpretation

34.6.10 SPDIFRxCChannel_l Register (SPDIF_SRCSL)

SPDIFRxCChannel_l register is a channel status reception register.

Address: 401D_C000h base + 20h offset = 401D_C020h



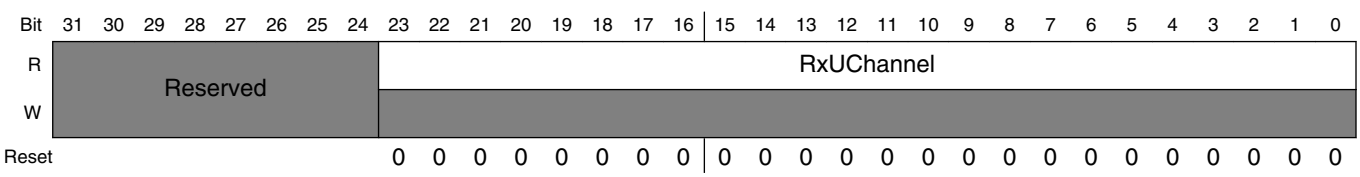
SPDIF_SRCSL field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
RxCChannel_l	SPDIF receive C channel register, contains next 24 bits of C channel without interpretation

34.6.11 UchannelRx Register (SPDIF_SRU)

UChannelRx register is a user bits reception register.

Address: 401D_C000h base + 24h offset = 401D_C024h



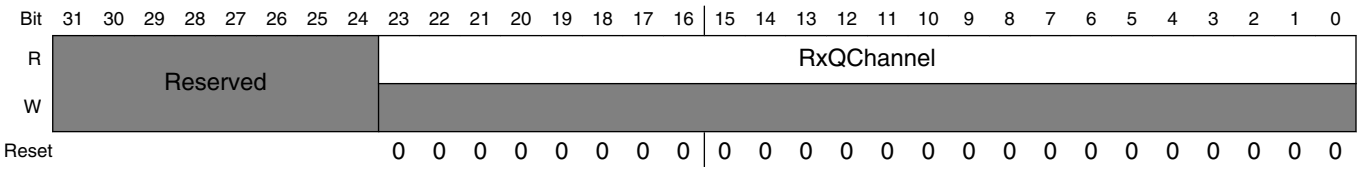
SPDIF_SRU field descriptions

Field	Description
31–24 [unimplemented]	This field is reserved. This is a 24-bit register the upper byte is unimplemented.
RxUChannel	SPDIF receive U channel register, contains next 3 U channel bytes

34.6.12 QchannelRx Register (SPDIF_SRQ)

QChannelRx register is a user bits reception register.

Address: 401D_C000h base + 28h offset = 401D_C028h



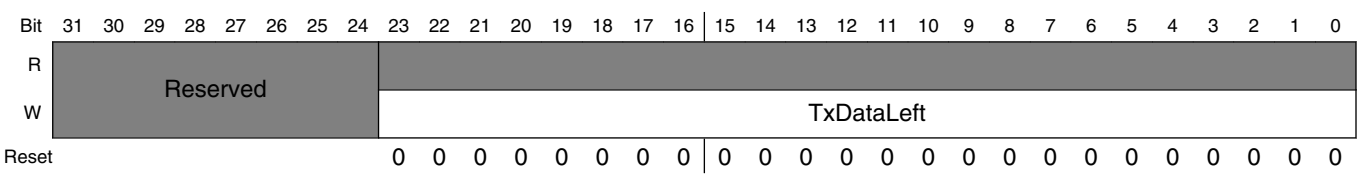
SPDIF_SRQ field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
RxQChannel	SPDIF receive Q channel register, contains next 3 Q channel bytes

34.6.13 SPDIFTxLeft Register (SPDIF_STL)

SPDIFTxLeft register is an audio data transmission register.

Address: 401D_C000h base + 2Ch offset = 401D_C02Ch



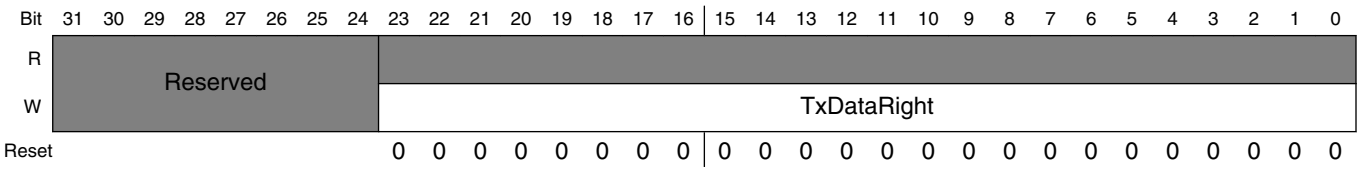
SPDIF_STL field descriptions

Field	Description
31–24 [unimplemented]	This field is reserved. This is a 24-bit register the upper byte is unimplemented.
TxDataLeft	SPDIF transmit left channel data. It is write-only, and always returns zeros when read

34.6.14 SPDIFTxRight Register (SPDIF_STR)

SPDIFTxRight register is an audio data transmission register.

Address: 401D_C000h base + 30h offset = 401D_C030h



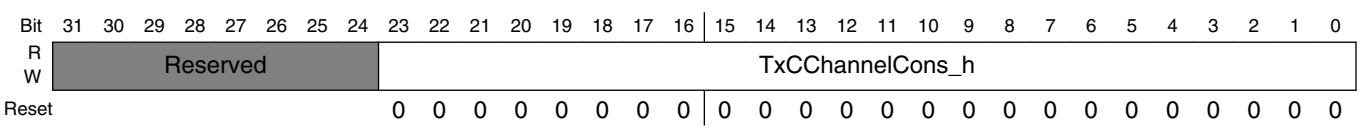
SPDIF_STR field descriptions

Field	Description
31–24 [unimplemented]	This field is reserved. This is a 24-bit register the upper byte is unimplemented.
TxDataRight	SPDIF transmit right channel data. It is write-only, and always returns zeros when read

34.6.15 SPDIFTxCChannelCons_h Register (SPDIF_STCSCH)

SPDIFTxCChannelCons_h register is a channel status transmission register.

Address: 401D_C000h base + 34h offset = 401D_C034h



SPDIF_STCSCH field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.

Table continues on the next page...

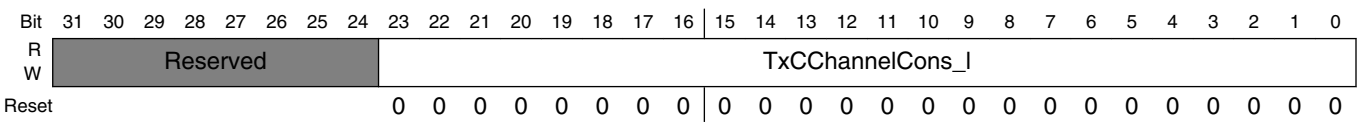
SPDIF_STCSCH field descriptions (continued)

Field	Description
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

34.6.16 SPDIFTxCChannelCons_l Register (SPDIF_STCSCL)

SPDIFTxCChannelCons_l register is a channel status transmission register.

Address: 401D_C000h base + 38h offset = 401D_C038h

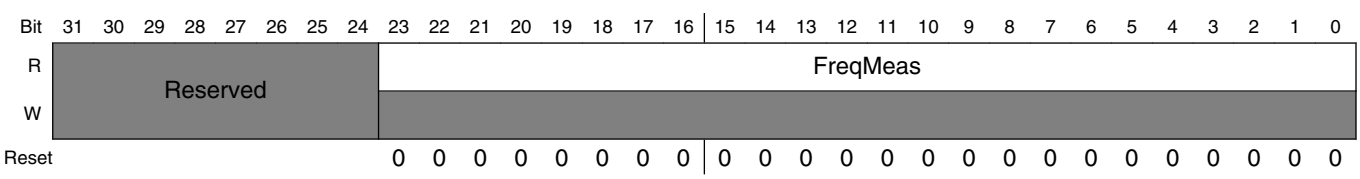


SPDIF_STCSCL field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
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

34.6.17 FreqMeas Register (SPDIF_SRFM)

Address: 401D_C000h base + 44h offset = 401D_C044h



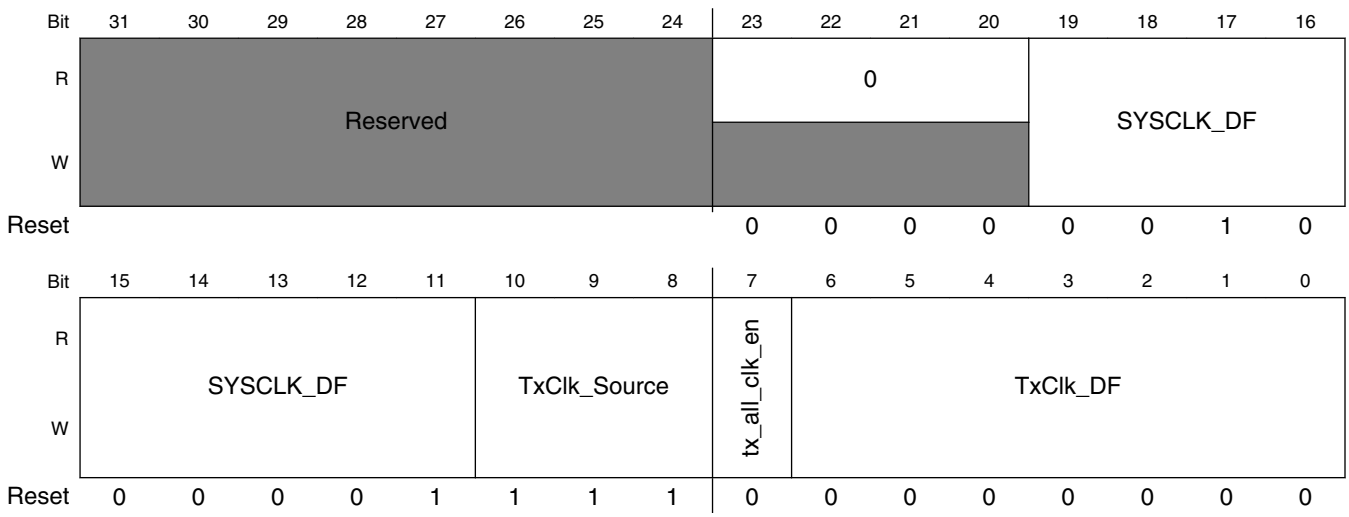
SPDIF_SRFM field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
FreqMeas	Frequency measurement data

34.6.18 SPDIFTxClk Register (SPDIF_STC)

The SPDIFTxClk Control register includes the means to select the transmit clock and frequency division.

Address: 401D_C000h base + 50h offset = 401D_C050h



SPDIF_STC field descriptions

Field	Description
31–24 [unimplemented]	This is a 24-bit register the upper byte is unimplemented. This field is reserved.
23–20 Reserved	This read-only field is reserved and always has the value 0.
19–11 SYSCLK_DF	system clock divider factor, 2~512. 0 no clock signal 1 divider factor is 2 ... ... 511 divider factor is 512
10–8 TxClk_Source	000 XTALOSC input (XTALOSC clock) 001 tx_clk input (from SPDIF0_CLK_ROOT. See CCM.) 010 tx_clk1 (from SAI1) 011 tx_clk2 SPDIF_EXT_CLK, from pads 100 tx_clk3 (from SAI2) 101 ipg_clk input (frequency divided) 110 tx_clk4 (from SAI3)
7 tx_all_clk_en	Spdif transfer clock enable. When data is going to be transfered, this bit should be set to1.

Table continues on the next page...

SPDIF_STC field descriptions (continued)

Field	Description
	0 disable transfer clock. 1 enable transfer clock.
TxClk_DF	Divider factor (1-128) 0 divider factor is 1 1 divider factor is 2 ... ... 127 divider factor is 128

Chapter 35
Universal Serial Bus Controller (USB)

35.1 Chip-specific USB 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 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 one independent USB controller core: On-The-Go (OTG) controller core. Each controller core supports UTMI interface. See Features for more details. The controller core is single-port core. For the OTG core, 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.

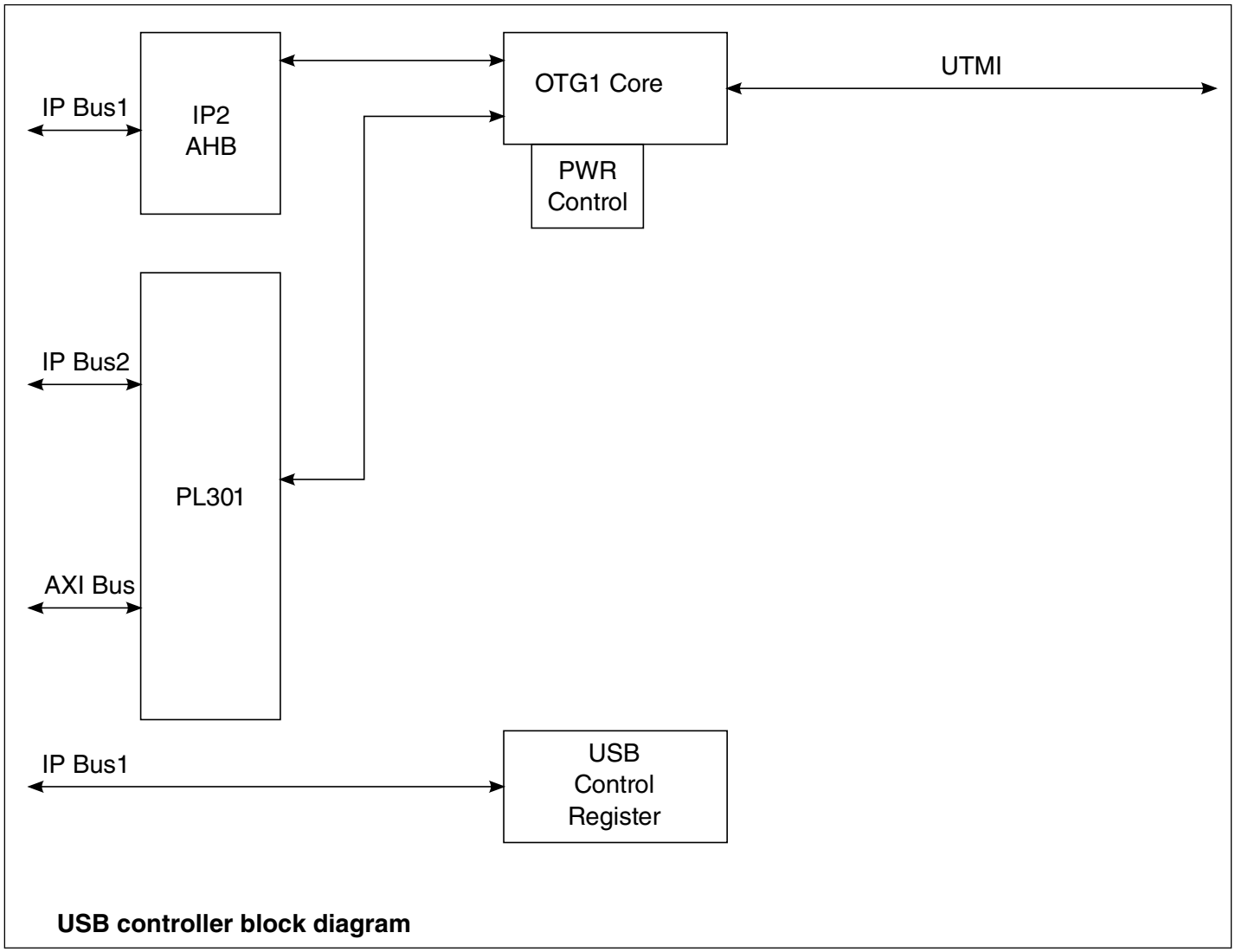


Figure 35-1. USB block diagram

35.2.1 Features

There is one USB 2.0 controller core in this chip:

- Controller Core 0 is also named 'OTG1 Core'; its connected port is named 'OTG1 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
- 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

35.2.2 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.

35.2.2.1 Normal Mode

The OTG controller core can operate in Host mode and Device (Peripheral) mode. The Host-only controller core can operate in Host mode only.

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.

35.2.2.2 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.

35.3 External Signals

The table found here describes the external signals of USB.

Table 35-2. USB External Signals

Signal	Description	Pad	Mode	Direction
USB_OTG1_ID	OTG1 ID Signal	GPIO_AD_10	ALT6	I
		GPIO_13	ALT3	
USB_OTG1_OC	External Input for VBUS over current detection	GPIO_AD_01	ALT6	I
		GPIO_12	ALT3	
USB_OTG1_PWR	To control PMIC to supply VBUS voltage	GPIO_AD_12	ALT6	O
		GPIO_AD_00	ALT3	

35.4 Functional Description

These sections describe the functionality of the various building blocks of the USB.

35.4.1 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.

35.4.1.1 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.

35.4.1.2 Peripheral (Device) Mode

- Up to eight bidirectional endpoints
- High/full-speed operation
- Support of HNP, and SRP
- Remote wake-up capability

35.4.2 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.

35.4.2.1 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.

35.4.2.2 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 (asynchronous) 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.

35.4.2.2.1 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).

35.4.3 Interrupts

35.4.3.1 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.

35.4.3.2 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.

35.5 USB Operation Model

This section describes the detailed application knowledge for OTG1 port.

35.5.1 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 35-3. 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.

35.5.1.1 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.

Table 35-4. 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
		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

Table continues on the next page...

Table 35-4. Device/Host Capability Registers (continued)

Offset	Size (Bytes)	Mnemonic	Register Name	Device Mode	Host Mode
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

35.5.1.2 Identification Registers

Identification registers are used to declare the slave interface presence and include a table of the hardware configuration parameters.

35.5.1.3 OTG Operations

35.5.2 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.

35.5.2.1 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.

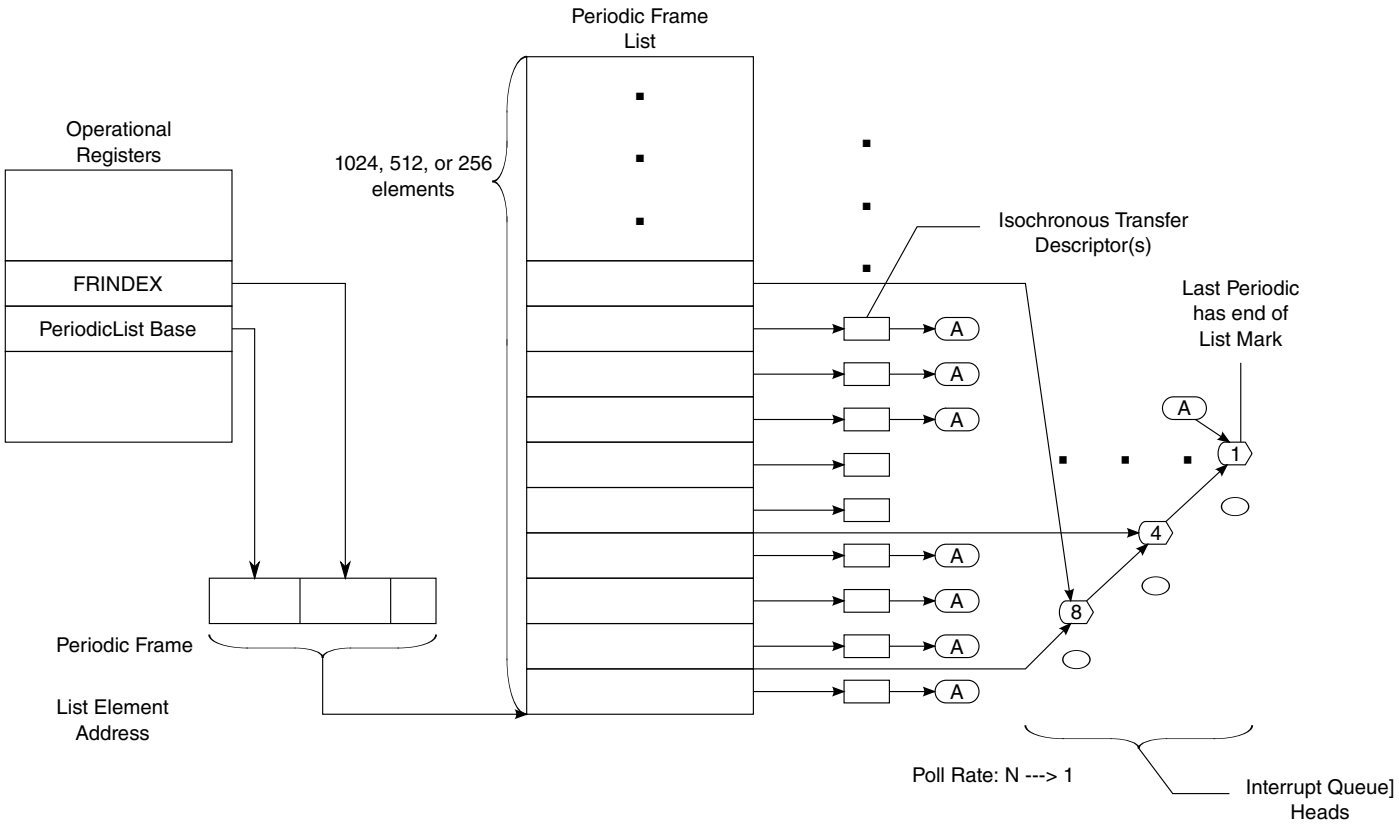


Figure 35-2. Periodic Schedule Organization

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 35-5. 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	

Table continues on the next page...

Table 35-5. Format of Frame List Element Pointer (continued)

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 35-6. Typ Field Value Definitions

Value	Meaning
00b	Isochronous Transfer Descriptor
01b	Queue Head
10b	Split Transaction Isochronous Transfer Descriptor.
11b	Frame Span Traversal Node.

35.5.2.2 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.

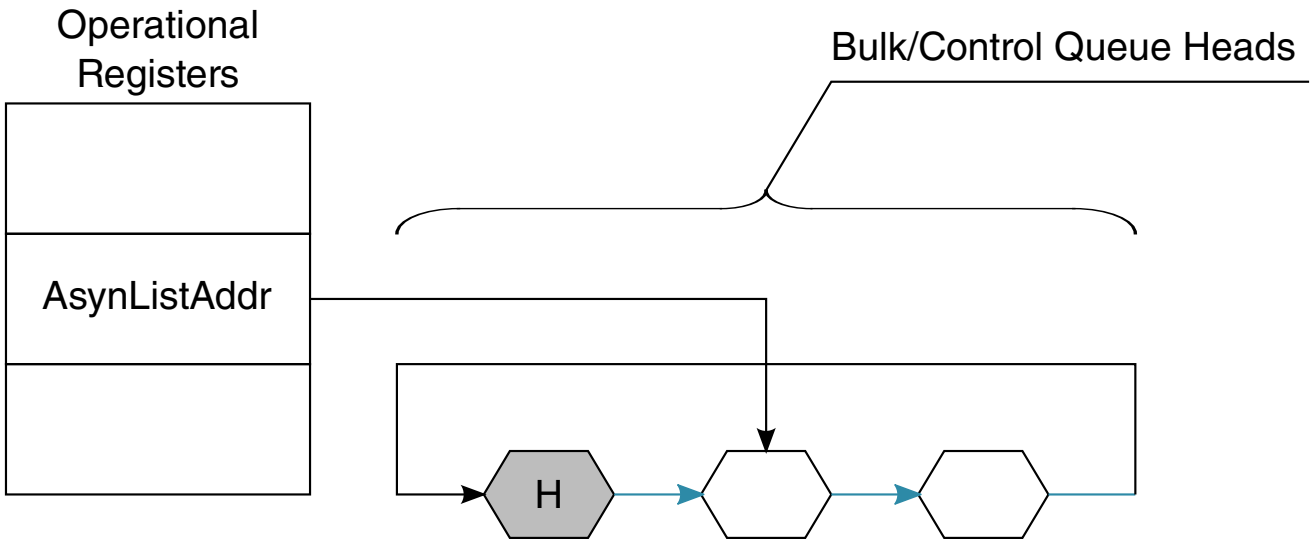


Figure 35-3. Asynchronous Schedule Organization

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.

35.5.2.3 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 35-7. 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 35-7. 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.

35.5.2.3.1 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 35-8. 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 35-8. 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.

35.5.2.3.2 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 35-9. 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 35-9. 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.	

35.5.2.3.3 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 35-10. 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 35-10. 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 35-11. 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 35-12. 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 35-13. 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.

35.5.2.4 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 35-14. 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

35.5.2.4.1 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 35-15. 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 35-15. 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.

35.5.2.4.2 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 35-16. 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 35-17. 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 35-17. 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.

35.5.2.4.3 siTD Transfer State

DWords 3-6 are used to manage the state of the transfer.

The following table describes siTD transfer state fields.

Table 35-18. 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 35-18. 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.

35.5.2.4.4 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 35-19. 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 35-19. 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.

35.5.2.4.5 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 35-20. 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.

35.5.2.5 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.

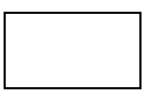
The following table shows the queue element transfer descriptor data structure.

Table 35-21. 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.

35.5.2.5.1 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 35-22. 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 35-22. 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.

35.5.2.5.2 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 35-23. 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.

35.5.2.5.3 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 35-24. 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 35-24. 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 35-24. 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 35-24. 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.

35.5.2.5.4 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 35-25. 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.

35.5.2.6 Queue Head

The table located in this section shows the Queue Head structure layout.

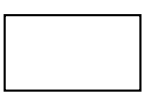
The following table shows the queue head structure layout.

Table 35-26. 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

35.5.2.6.1 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 35-27. 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.

35.5.2.6.2 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 35-28. 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 35-29. 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 35-29. 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.

35.5.2.6.3 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 35-30. 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 35-31. 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.

35.5.2.7 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.

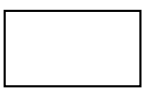
See Host Controller Operational Model for FSTNs for full operational details. Software must not use an FSTN in the Asynchronous Schedule. An FSTN in the Asynchronous schedule results in undefined behavior. Software must not use the FSTN feature with a host controller whose USB_HCIVERSION register indicates a revision implementation below 0096h. FSTNs are not defined for implementations before 0.96 and their use yields undefined results.

Table 35-32. 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

35.5.2.7.1 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 35-33. 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.

35.5.2.7.2 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 35-34. 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.

35.5.3 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.

35.5.3.1 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 35-35. 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.

35.5.3.2 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.

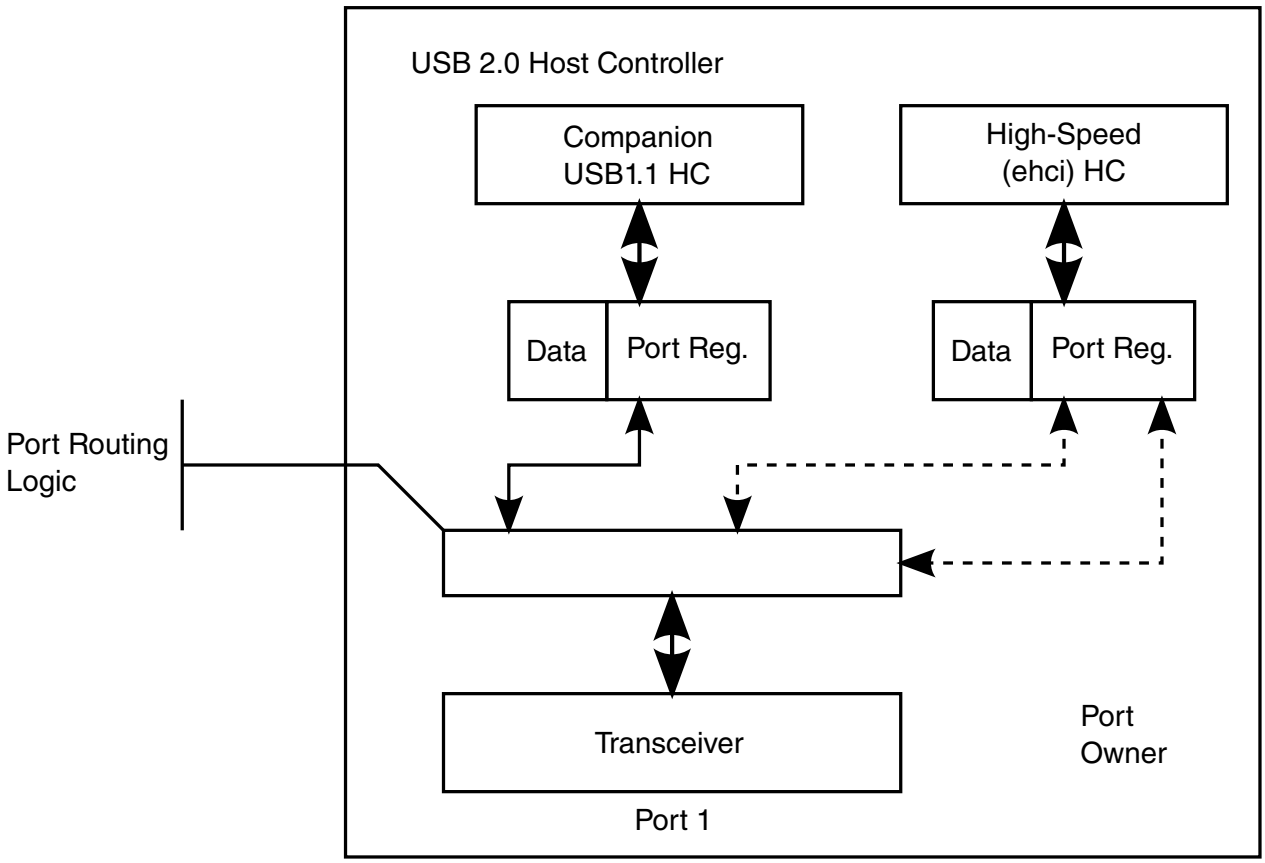


Figure 35-4. Example USB 2.0 Host Controller Port Routing Block Diagram

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)

35.5.3.2.1 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 35-36. 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.

35.5.3.2.2 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.

35.5.3.2.3 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.

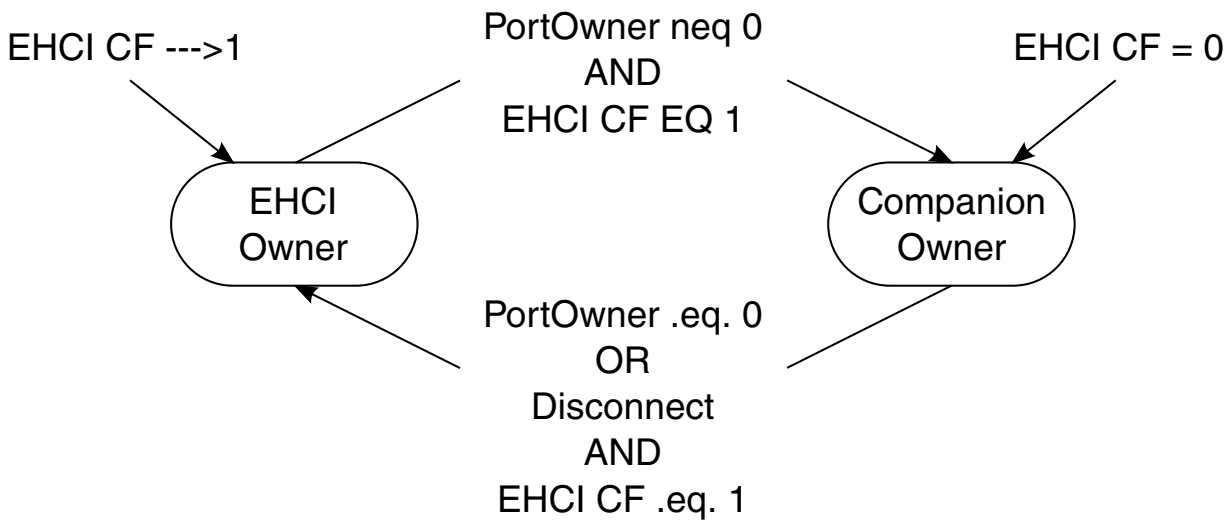


Figure 35-5. Port Owner Handoff State Machine

35.5.3.2.3.1 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.

35.5.3.2.3.2 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.

35.5.3.2.4 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 35-37. 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 35-37. 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).

35.5.3.2.5 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 35-38 for summary behavior for over-current detection

when the host controller is halted (suspended from a device component point of view).

35.5.3.3 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.

35.5.3.3.1 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 35-38. 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.

35.5.3.4 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.

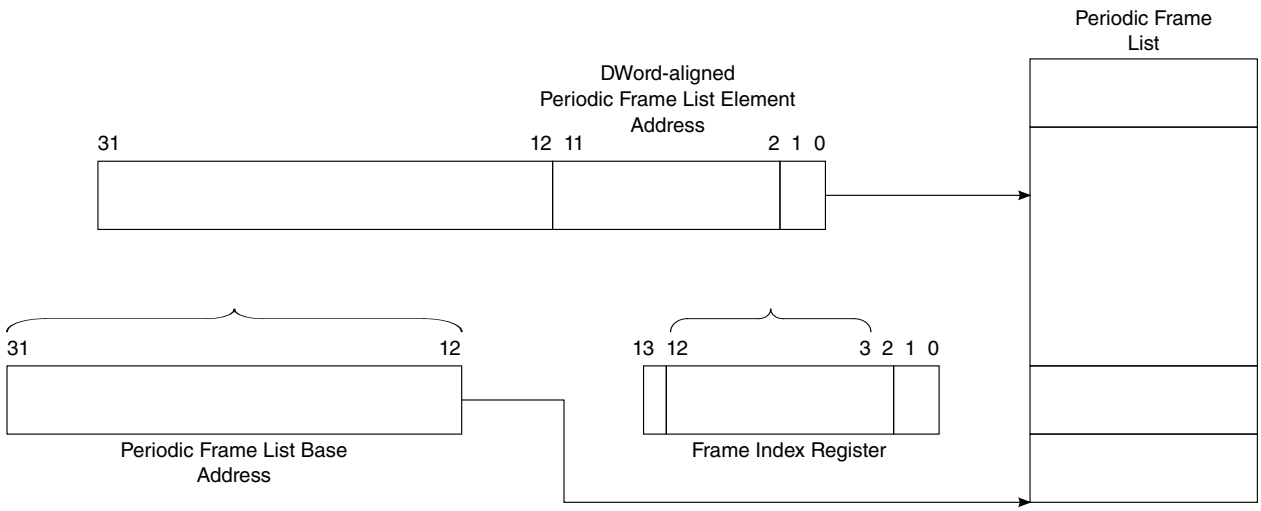


Figure 35-6. Derivation of Pointer into Frame List Array

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.

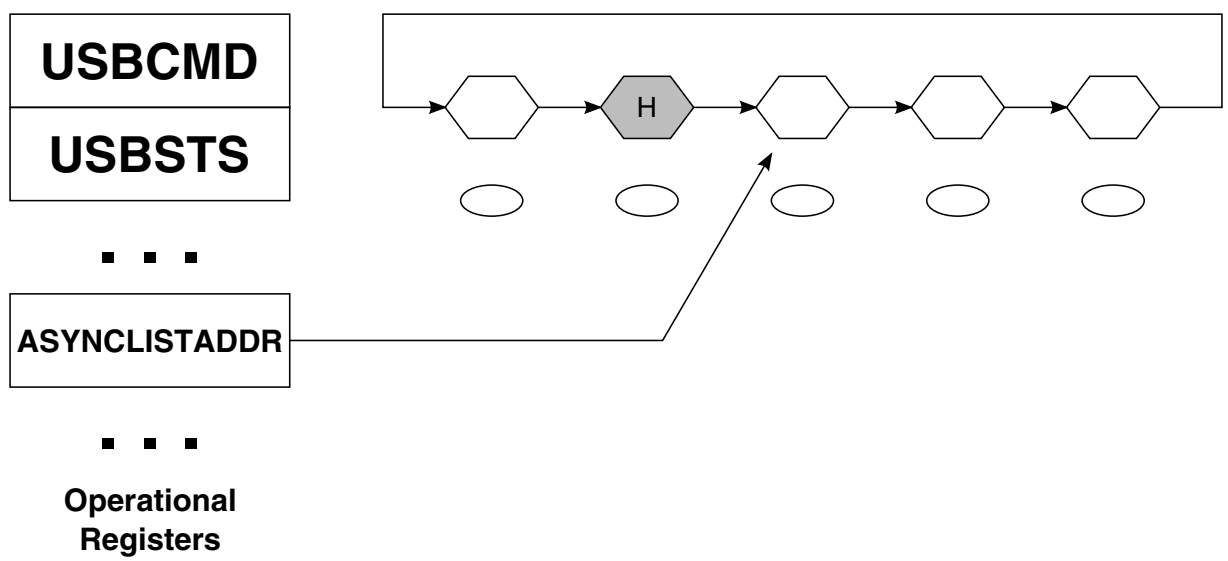


Figure 35-7. General Format of Asynchronous Schedule List

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.

35.5.3.4.1 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.

35.5.3.4.1.1 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 35-8. 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.

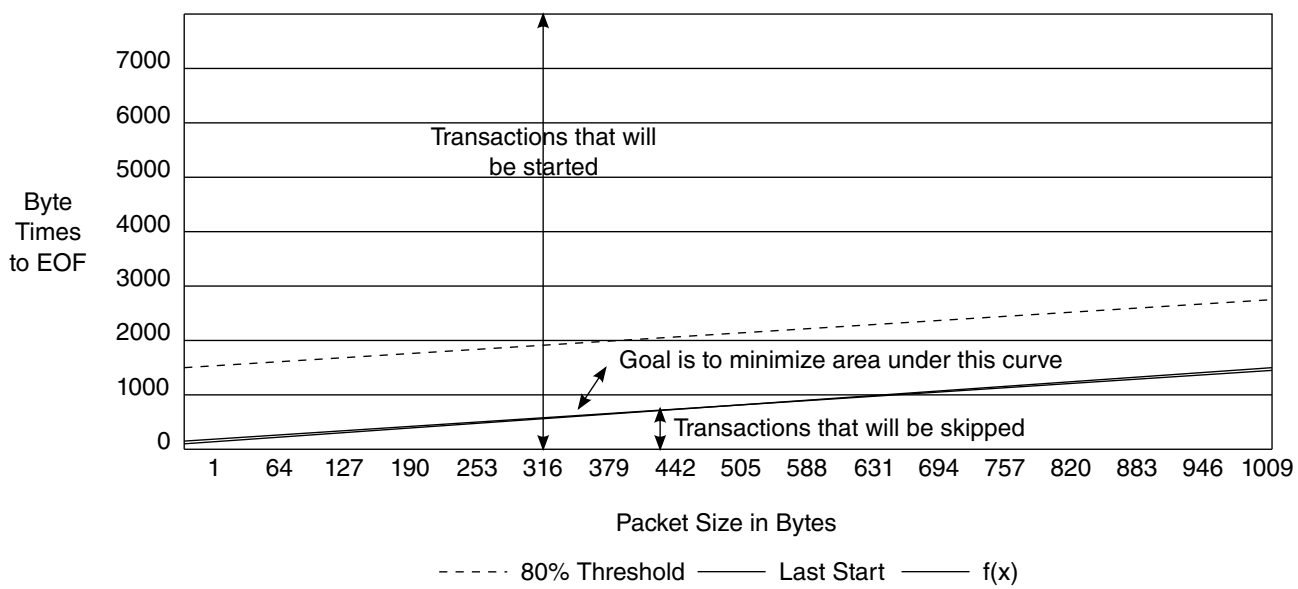


Figure 35-8. Best Fit Approximation

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 35-39. Example Worse-case Transaction Timing Components

Component	Bit time	Byte Time	Explanation

Table continues on the next page...

Table 35-39. 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 35-8 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.

35.5.3.5 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.

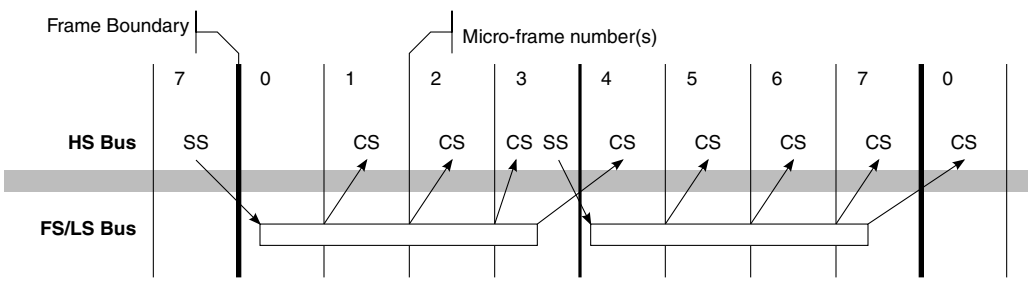


Figure 35-9. Frame Boundary Relationship between HS bus and FS/LS Bus

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.

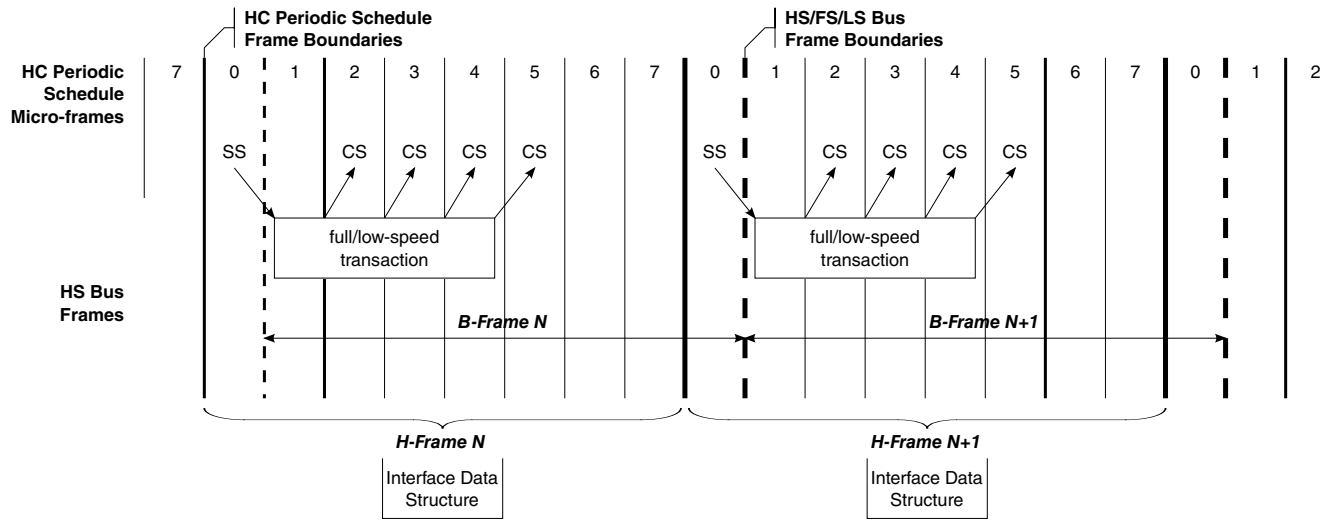


Figure 35-10. Relationship of Periodic Schedule Frame Boundaries to Bus Frame Boundaries

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 35-40. 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]

35.5.3.6 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.

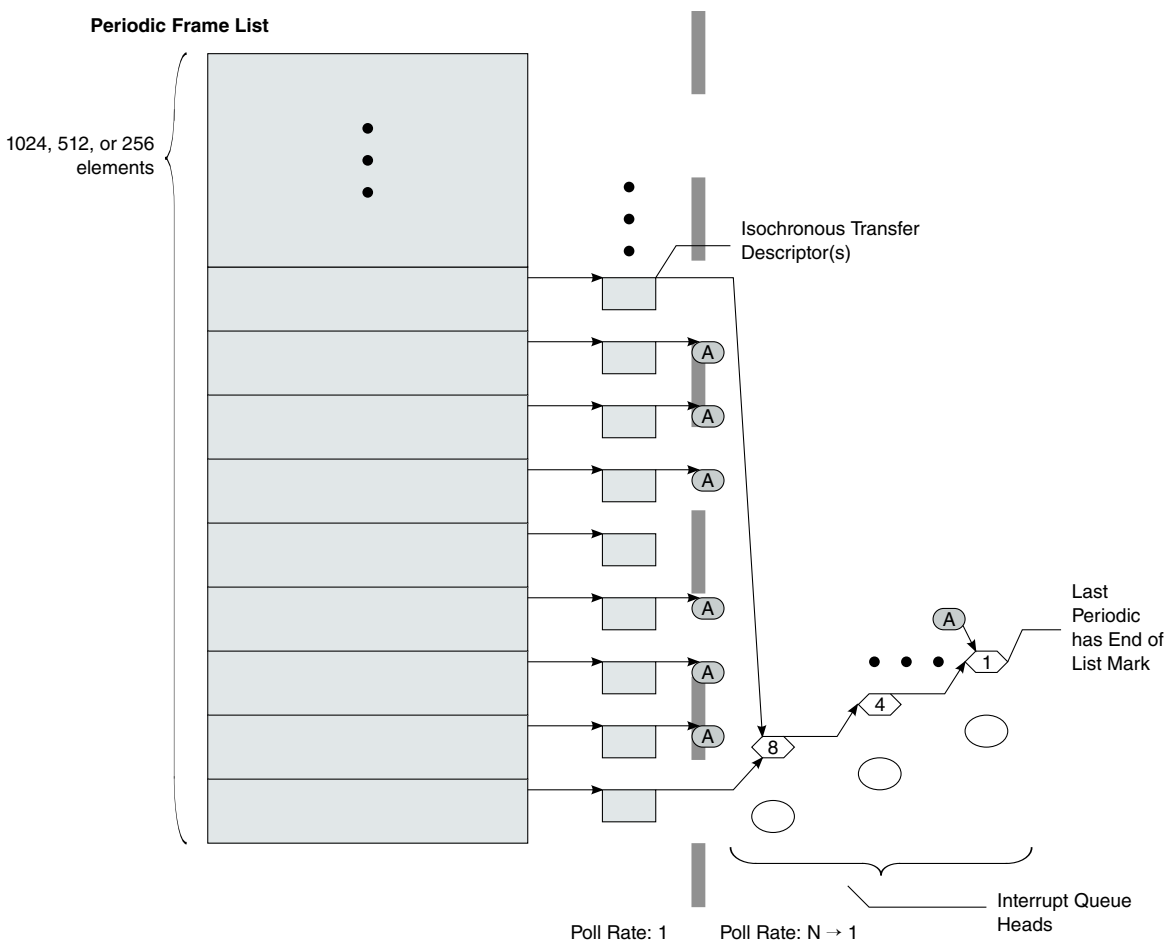


Figure 35-11. Example Periodic Schedule

35.5.3.7 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.

35.5.3.7.1 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.

35.5.3.7.2 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.

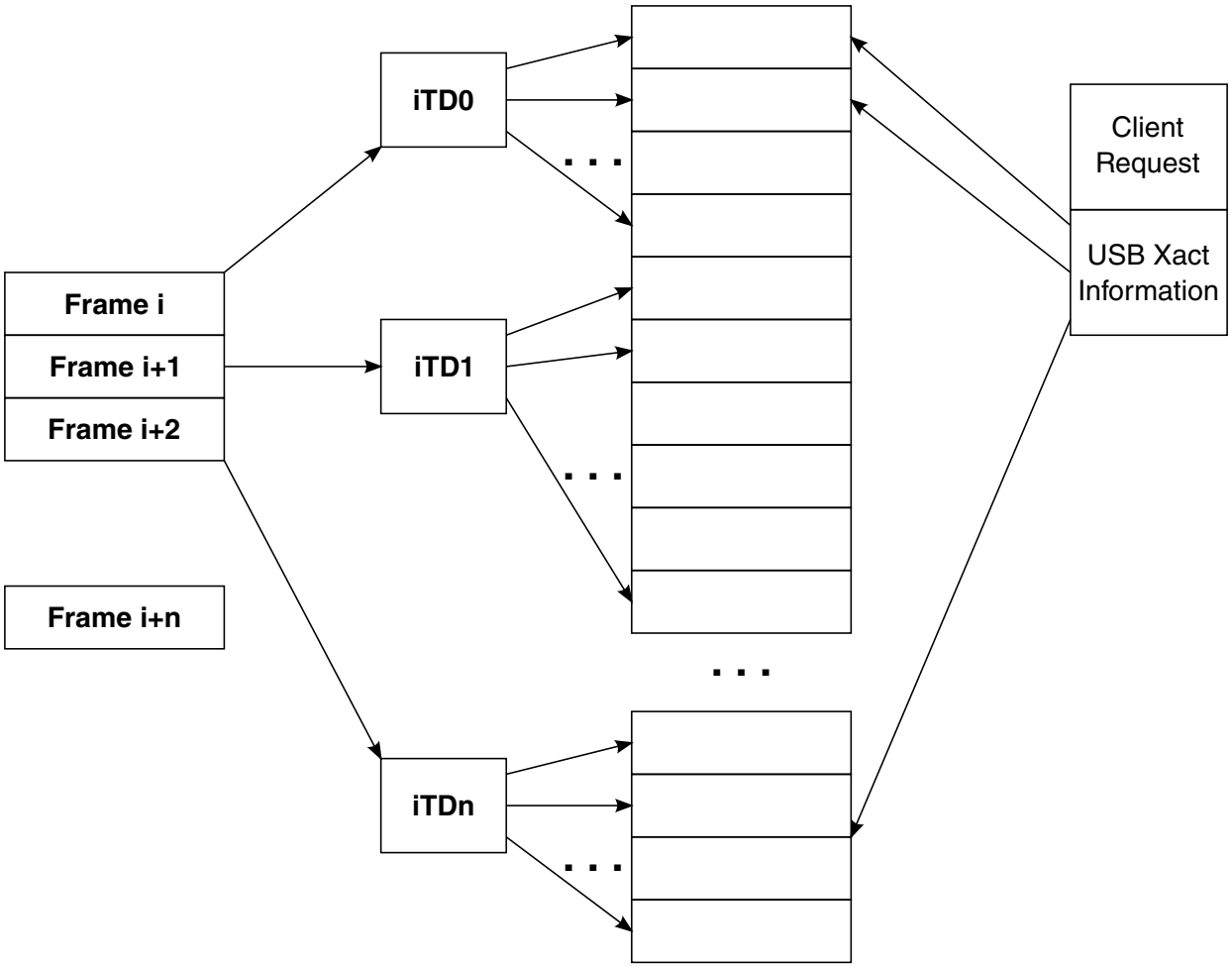


Figure 35-12. Example Association of iTDs to Client Request Buffer

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.

35.5.3.7.2.1 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.

35.5.3.8 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 35-7. 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.

35.5.3.8.1 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
```

35.5.3.8.2 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.

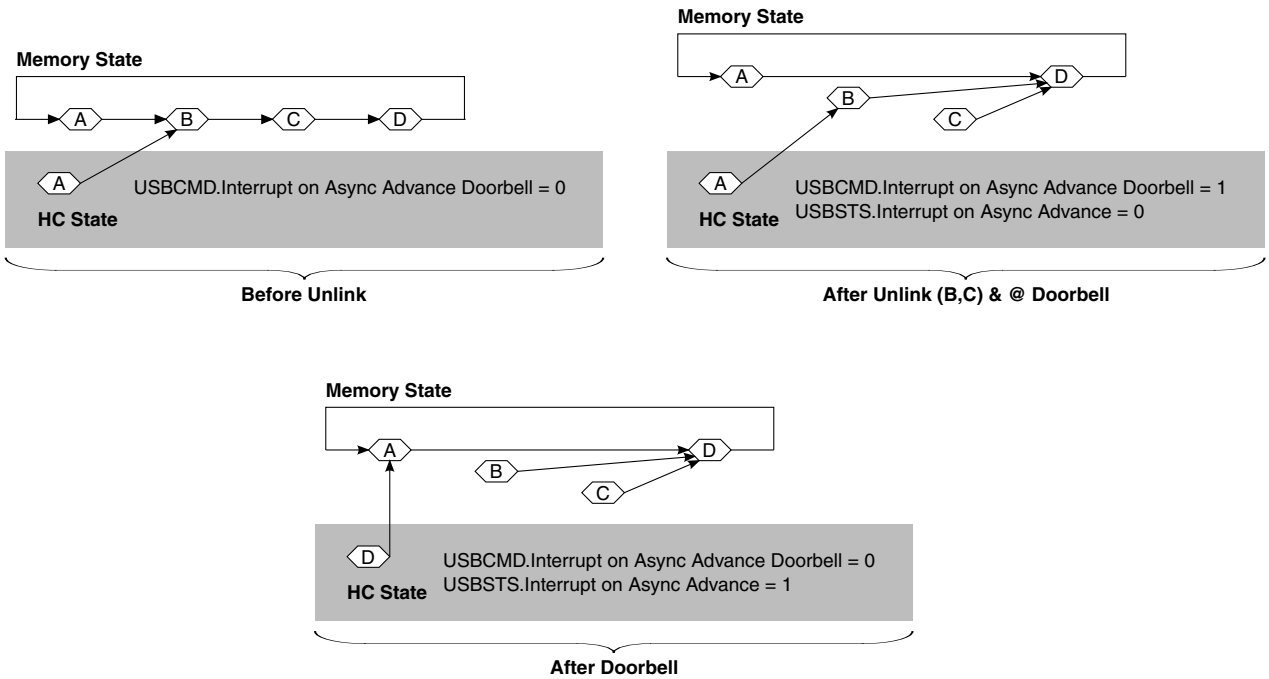


Figure 35-13. Generic Queue Head Unlink Scenario

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.

35.5.3.8.3 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 35-26) 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.

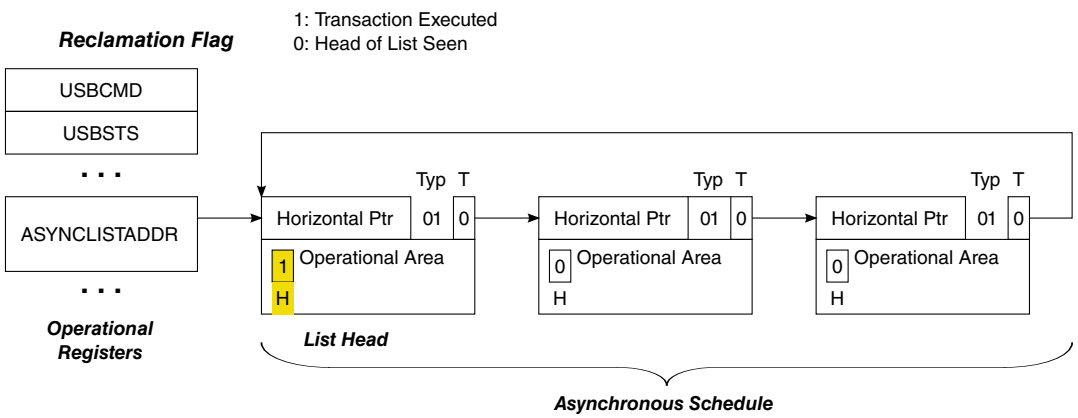


Figure 35-14. Asynchronous Schedule List w/Annotation to Mark Head of List

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.

35.5.3.8.4 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.

35.5.3.8.4.1 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.

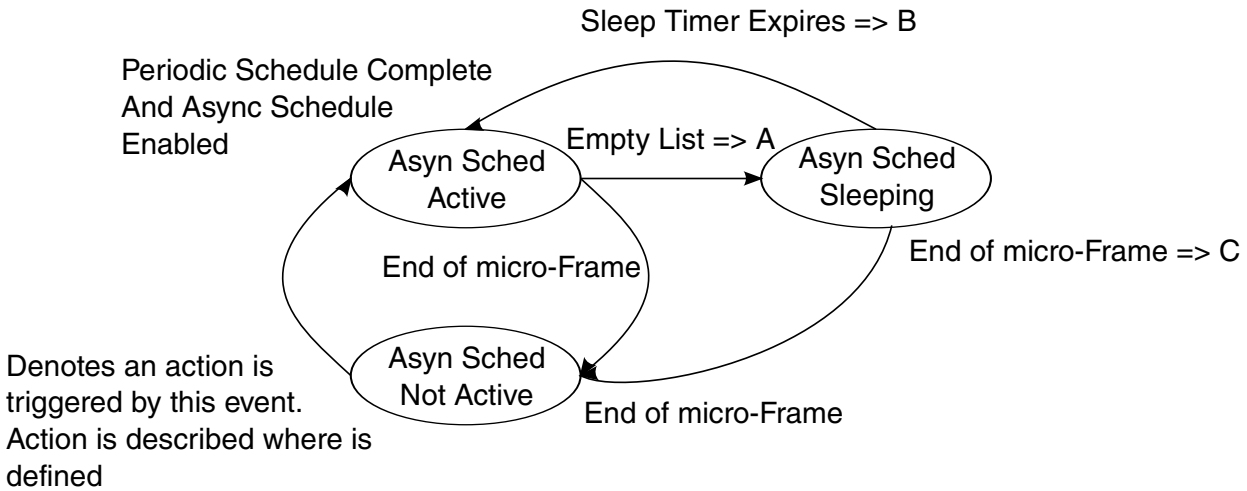


Figure 35-15. Example State Machine for Managing Asynchronous Schedule Traversal

The actions referred to in the figure above are defined in the following table.

Table 35-41. 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*).

35.5.3.8.4.2 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.

35.5.3.8.4.3 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.

35.5.3.8.4.4 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*.

35.5.3.8.4.5 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 35-42. 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.

35.5.3.8.5 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).

35.5.3.8.6 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.

35.5.3.9 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 35-43. 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.

35.5.3.9.1 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 35-26) 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 35-14).

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 35-26) is set to zero.

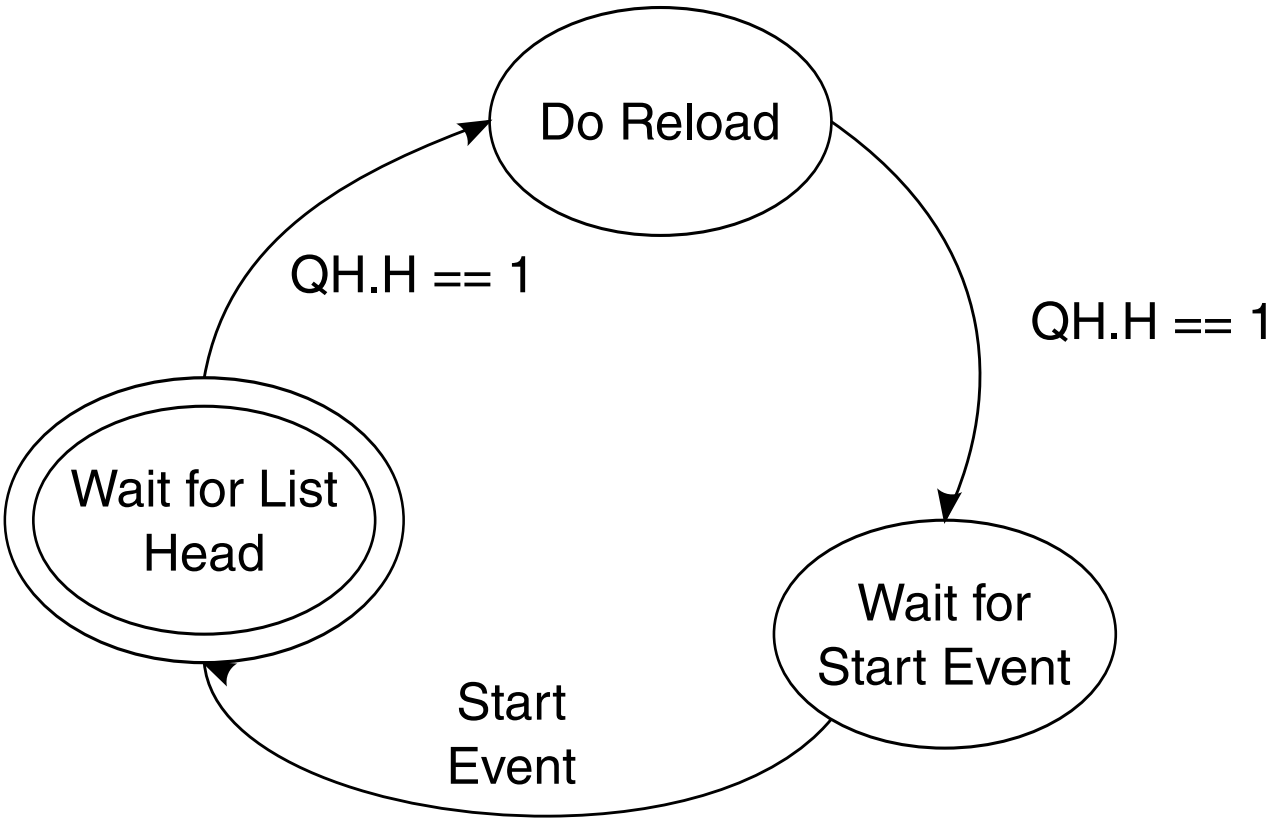


Figure 35-16. Example HC State Machine for Controlling Nak Counter Reloads

35.5.3.9.1.1 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.

35.5.3.9.1.2 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.

35.5.3.9.1.3 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.

35.5.3.10 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 35-26). 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.

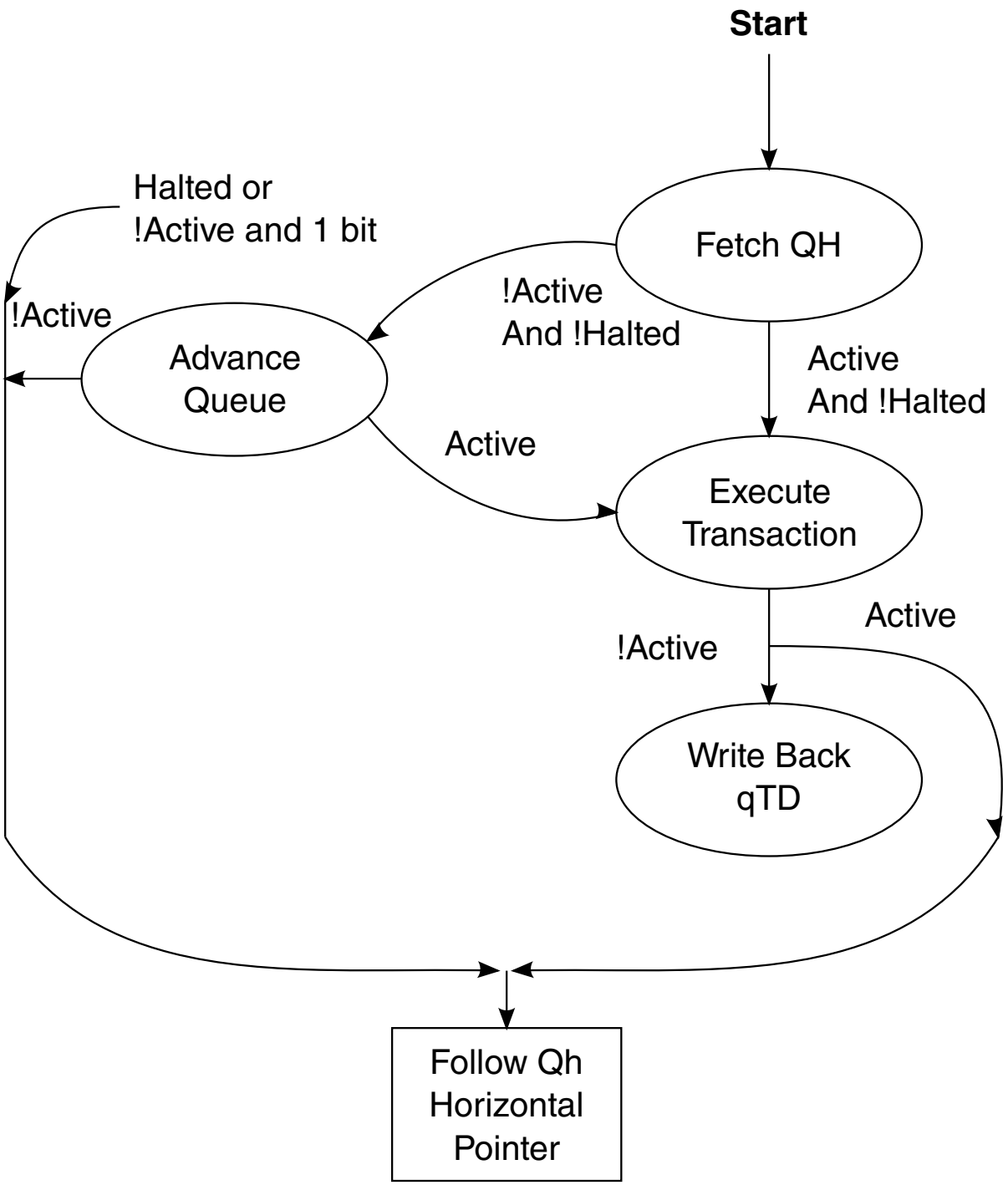


Figure 35-17. Host Controller Queue Head Traversal State Machine

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.

35.5.3.10.1 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 35-26.

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.

35.5.3.10.2 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 35-28).
- 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.

35.5.3.10.3 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 .

35.5.3.10.3.1 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.

35.5.3.10.3.2 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.

35.5.3.10.3.3 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.

35.5.3.10.3.4 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.

35.5.3.10.3.5 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 35-44. 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 35-44. 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.

35.5.3.10.4 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 35-44).

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.

35.5.3.10.5 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.

35.5.3.10.6 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.

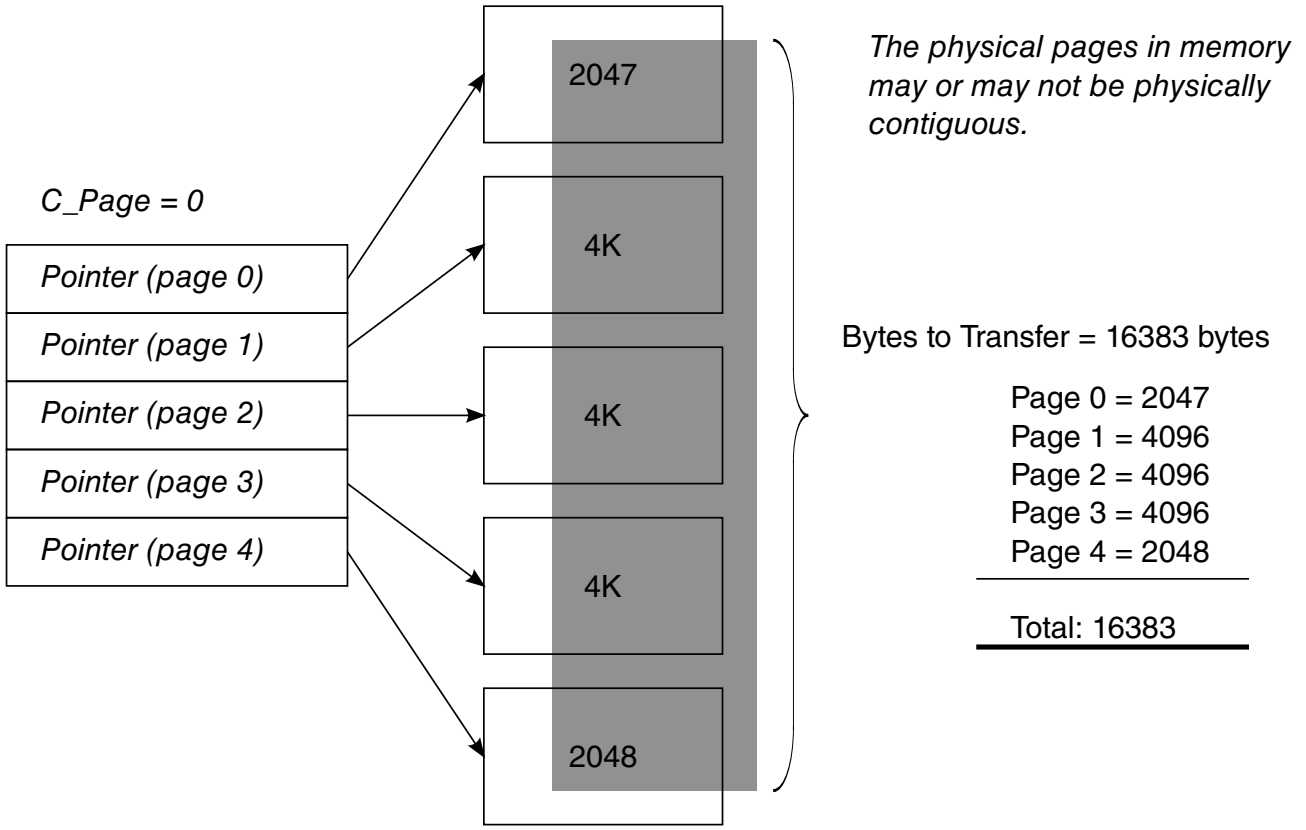


Figure 35-18. Example Mapping of qTD Buffer Pointers to Buffer Pages

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.

35.5.3.10.7 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 35-45. 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.

35.5.3.10.8 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.

35.5.3.11 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 35-46. 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 35-46. 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 35-47. 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.

35.5.3.12 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.

35.5.3.12.1 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.

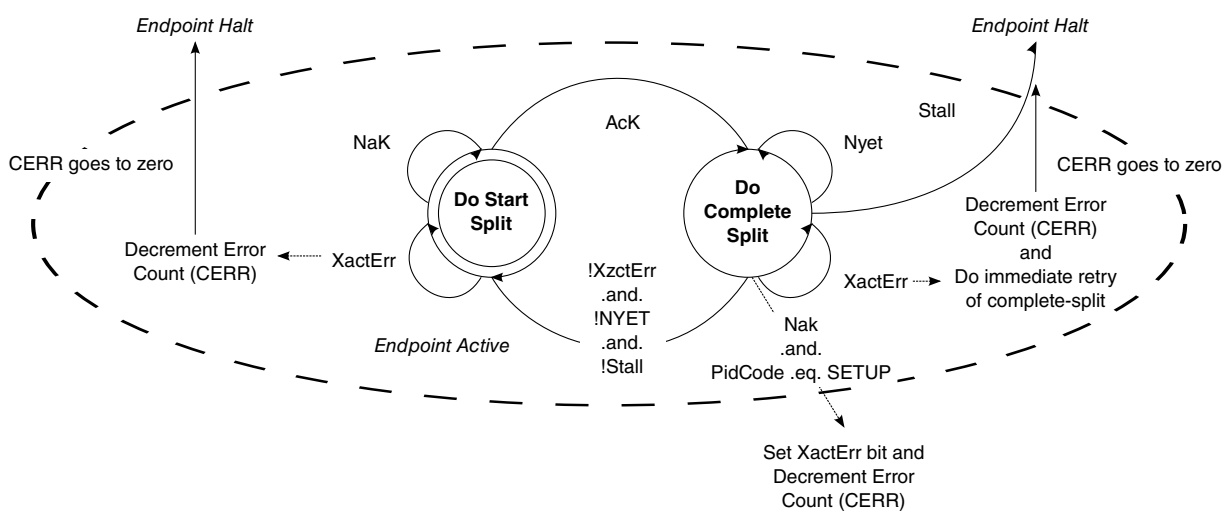


Figure 35-19. Host Controller Asynchronous Schedule Split-Transaction State Machine

35.5.3.12.1.1 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.

35.5.3.12.1.2 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).

35.5.3.12.2 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.

35.5.3.12.2.1 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).

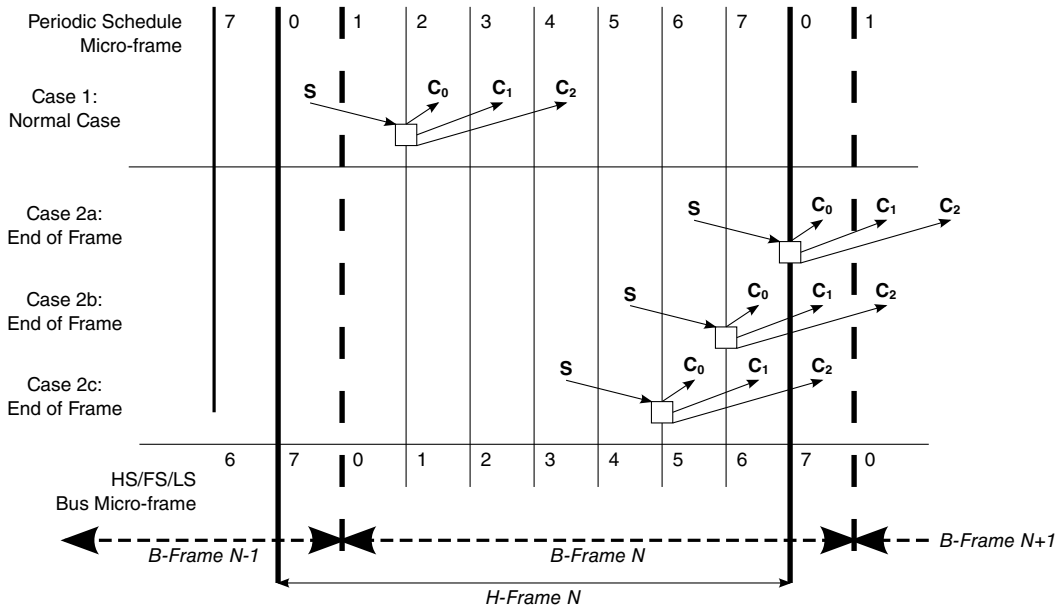


Figure 35-20. Split Transaction, Interrupt Scheduling Boundary Conditions

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.

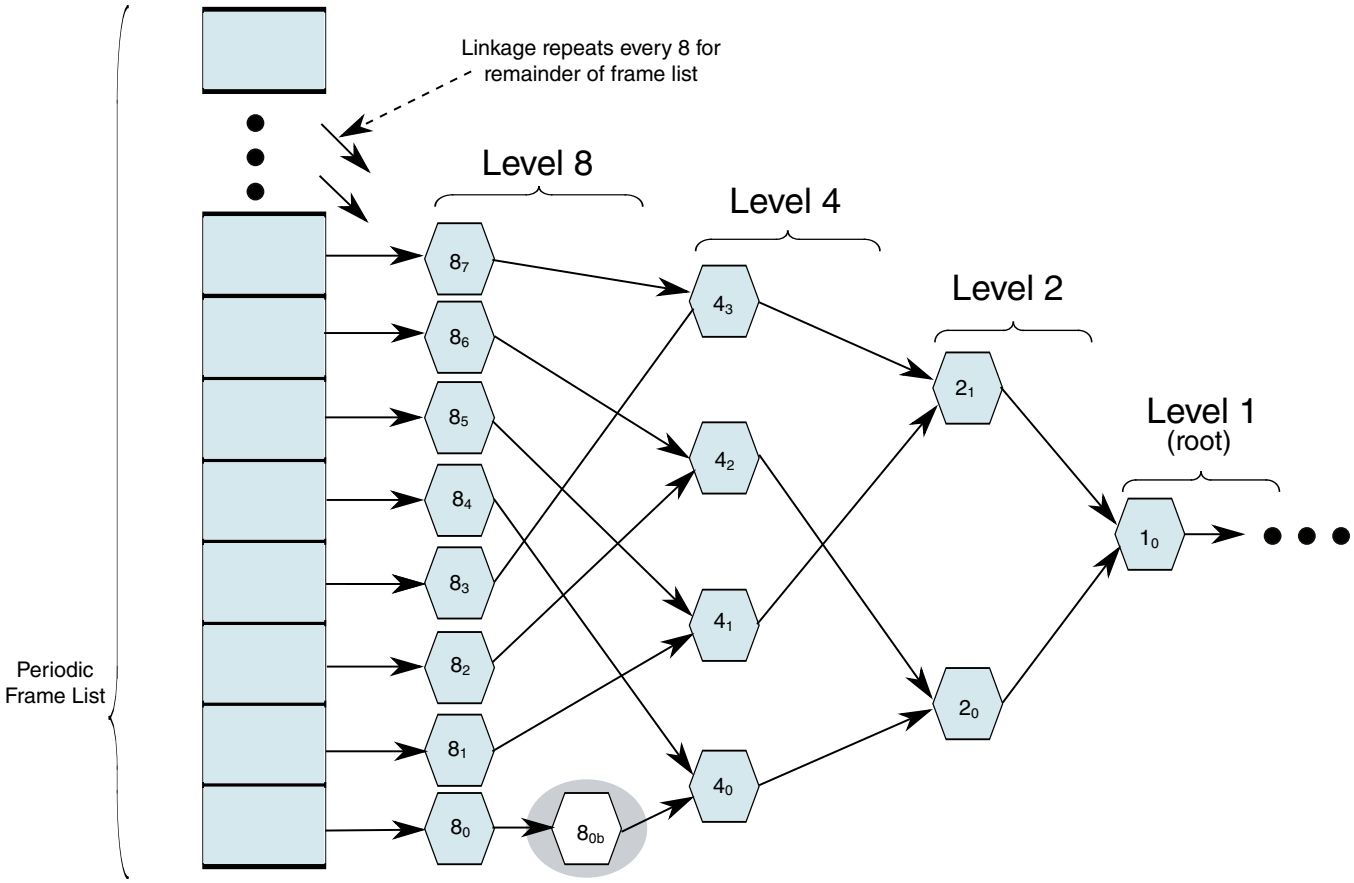


Figure 35-21. General Structure of EHCI Periodic Schedule Utilizing Interrupt Spreading

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 35-24). 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 35-20, 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 35-20, 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*

35.5.3.12.2.2 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.

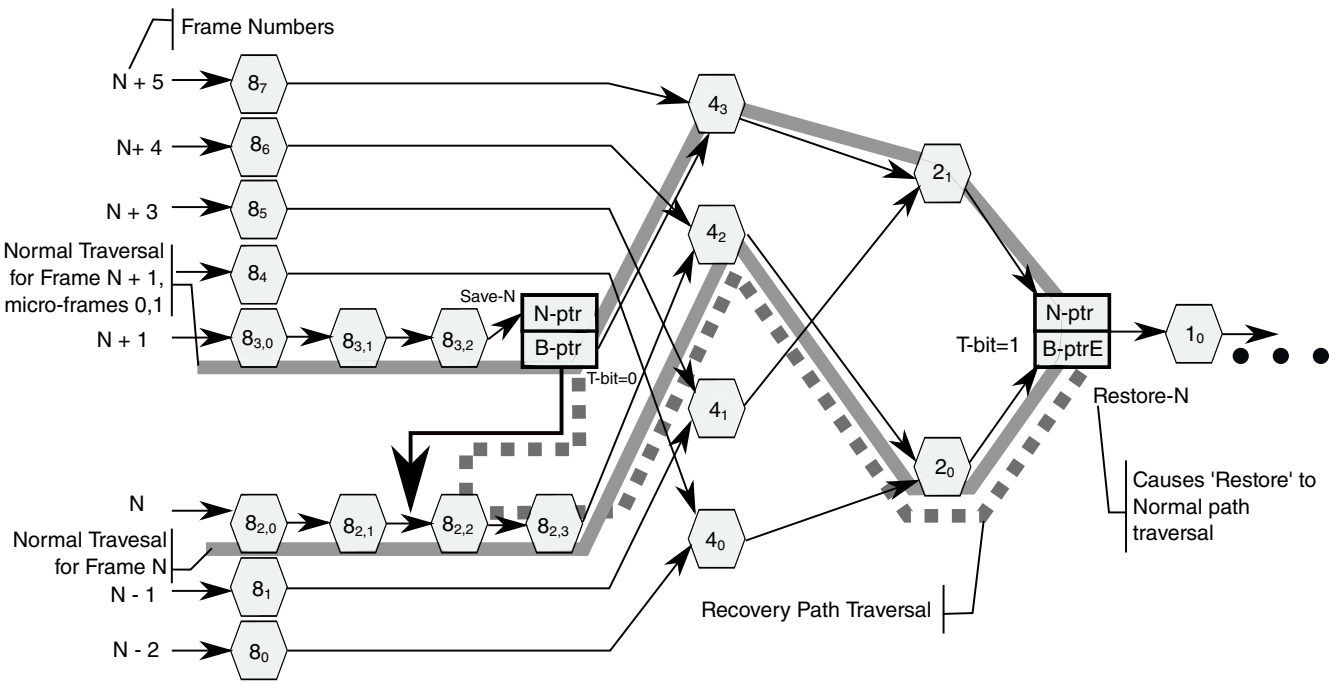


Figure 35-22. Example Host Controller Traversal of Recovery Path via FSTNs

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 …}.

35.5.3.12.2.3 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.

35.5.3.12.2.4 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).

35.5.3.12.2.5 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).

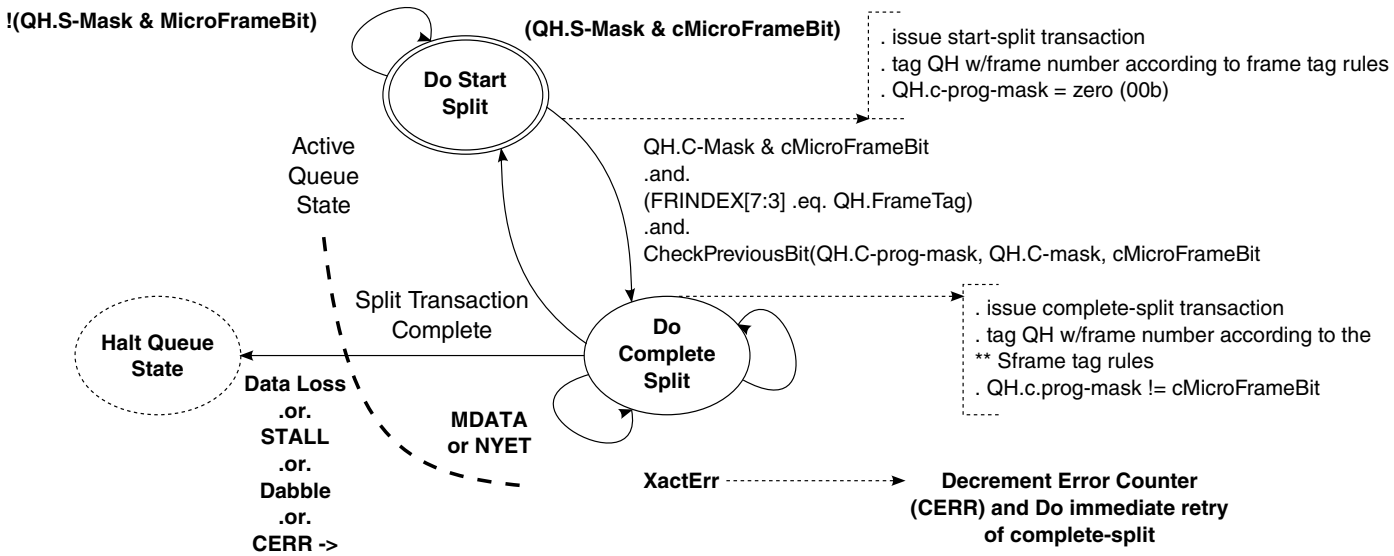


Figure 35-23. Split Transaction State Machine for Interrupt

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 35-48. 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 35-48. 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 35-20).
- 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 35-20).
- 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 35-20).

35.5.3.12.2.6 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.

35.5.3.12.3 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.

35.5.3.12.3.1 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*.

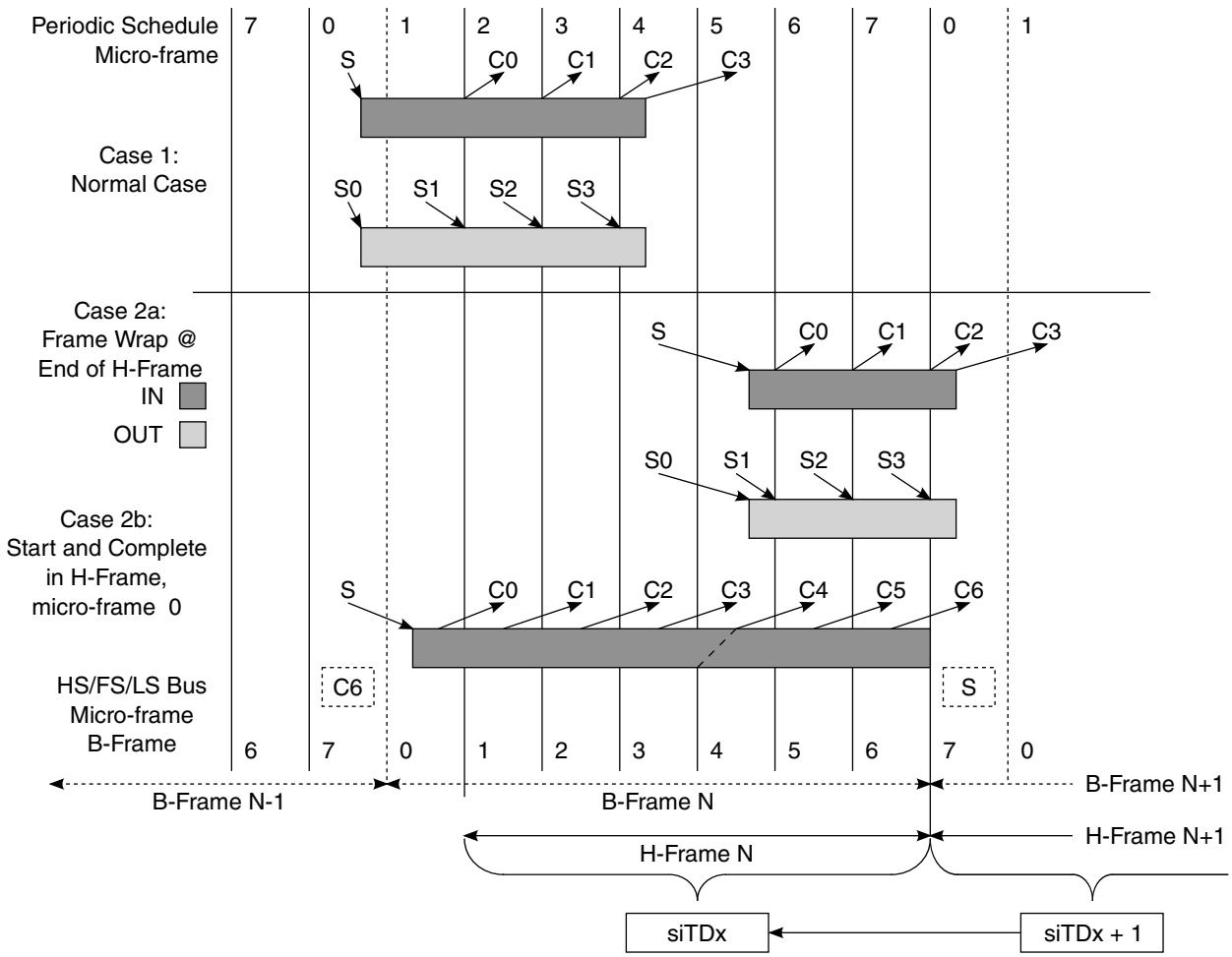


Figure 35-24. Split Transaction, Isochronous Scheduling Boundary Conditions

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 35-25). 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 35-24, 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 35-24, 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.

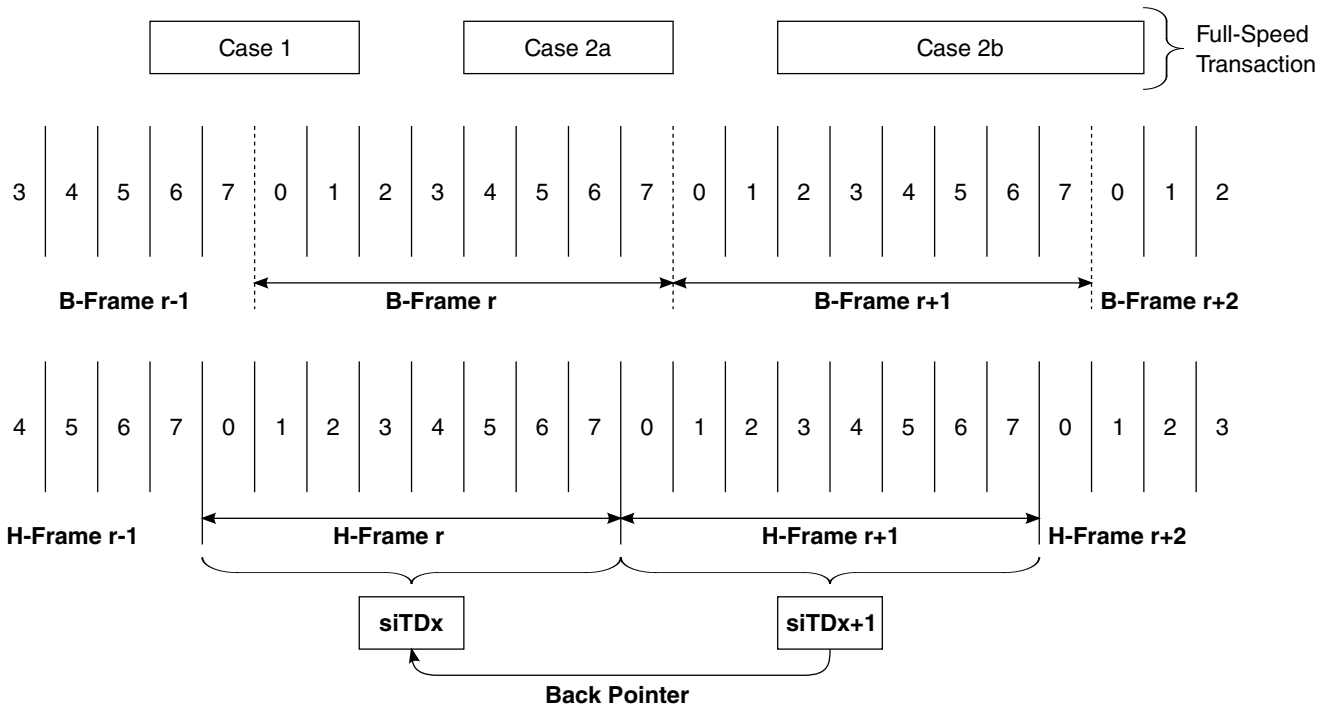


Figure 35-25. siTD Scheduling Boundary Examples

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.

35.5.3.12.3.2 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.

35.5.3.12.3.3 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.

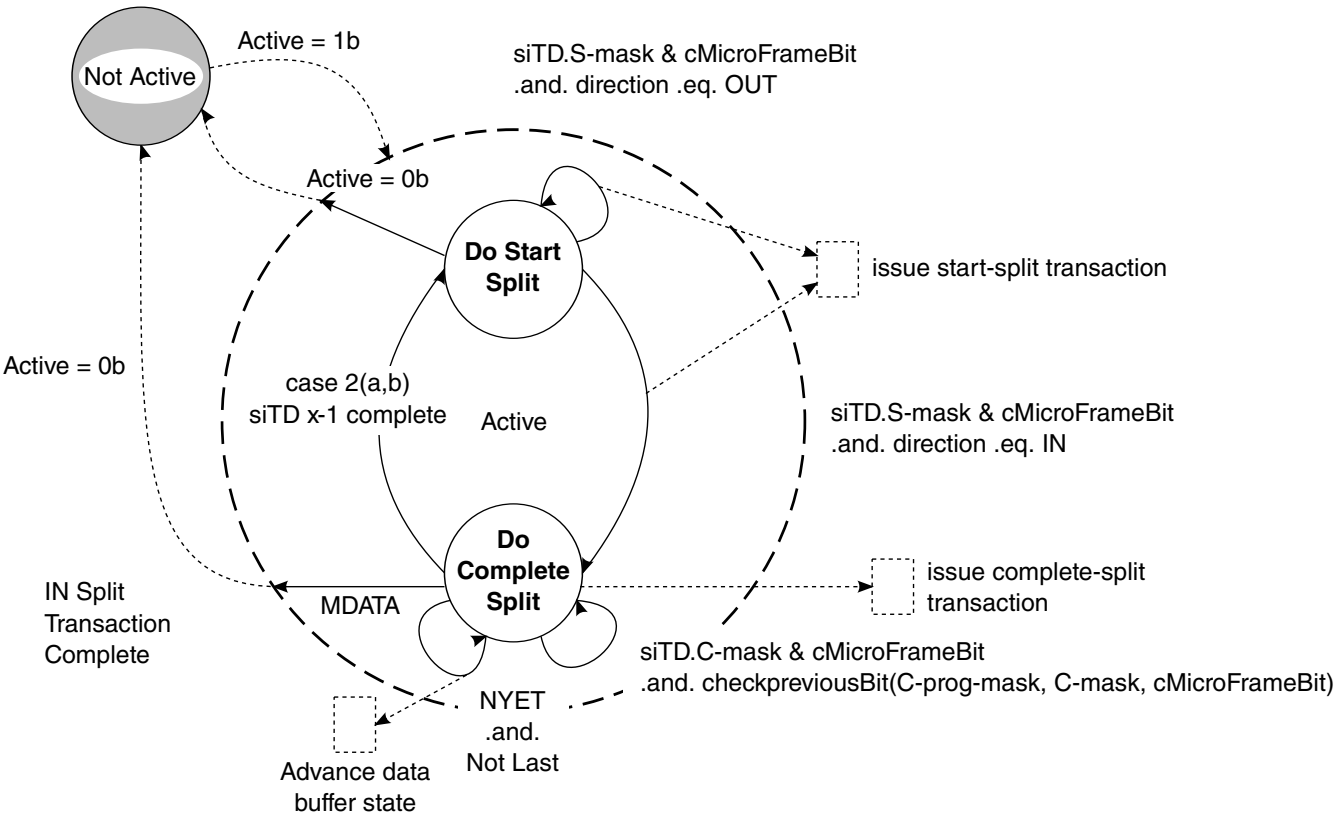


Figure 35-26. Split Transaction State Machine for Isochronous

35.5.3.12.3.4 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 35-25) is used to determine the initial value of *TP*. The initial cases are summarized in the following table.

Table 35-49. 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 35-50. 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 35-50.

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.

35.5.3.12.3.5 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 35-24). 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.

35.5.3.12.3.6 Complete-Split for Scheduling Boundary Cases 2a, 2b

Boundary cases 2a and 2b (INs only) (see Figure 35-24) 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 35-51. 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 35-51) 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.

35.5.3.12.3.7 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 35-17).

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 35-52. 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.

35.5.3.13 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).

35.5.3.14 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.

35.5.3.15 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.

35.5.3.15.1 Transfer/Transaction Based Interrupts

These interrupt sources are associated with transfer and transaction progress. They are all dependent on the next interrupt threshold.

35.5.3.15.1.1 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 35-53. 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.

35.5.3.15.1.2 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.

35.5.3.15.1.3 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.

35.5.3.15.1.4 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.

35.5.3.15.1.5 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.

35.5.3.15.2 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).

35.5.3.15.2.1 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

35.5.3.15.2.2 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.

35.5.3.15.2.3 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 .

35.5.3.15.2.4 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 35-54. 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.

35.5.4 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.

35.5.4.1 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.

35.5.4.1.1 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

35.5.4.1.2 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.

35.5.4.1.3 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 35-55. 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)]

35.5.4.1.4 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.

35.5.4.1.5 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.

35.5.4.1.5.1 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.)

35.5.4.1.5.2 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 35-56. 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

35.5.4.1.5.3 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.

35.5.4.1.5.3.1 USB 2.0 - 11.17.3

- Sequencing is provided & a packet length estimator ensures no full-speed/low-speed packet babbles into SOF time.

35.5.4.1.5.3.2 USB 2.0 - 11.17.4

- Transaction tracking for 2 data pipes.

35.5.4.1.5.4 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.

35.5.4.1.5.4.1 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

35.5.4.1.5.4.2 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.

35.5.4.1.5.5 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.

35.5.4.2 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.

35.5.4.2.1 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.

35.5.4.2.2 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 .

35.5.4.2.3 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.

35.5.4.3 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.

35.5.4.3.1 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.

35.5.4.4 Miscellaneous variations from EHCI

35.5.4.4.1 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.

35.5.4.4.2 Discovery

35.5.4.4.2.1 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.

35.5.4.4.2.2 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.*

35.5.4.4.3 Port Test Mode

Port Test Control mode behaves fully as described in EHCI. An alternate host controller driver procedure is not necessary or supported.

35.5.5 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.

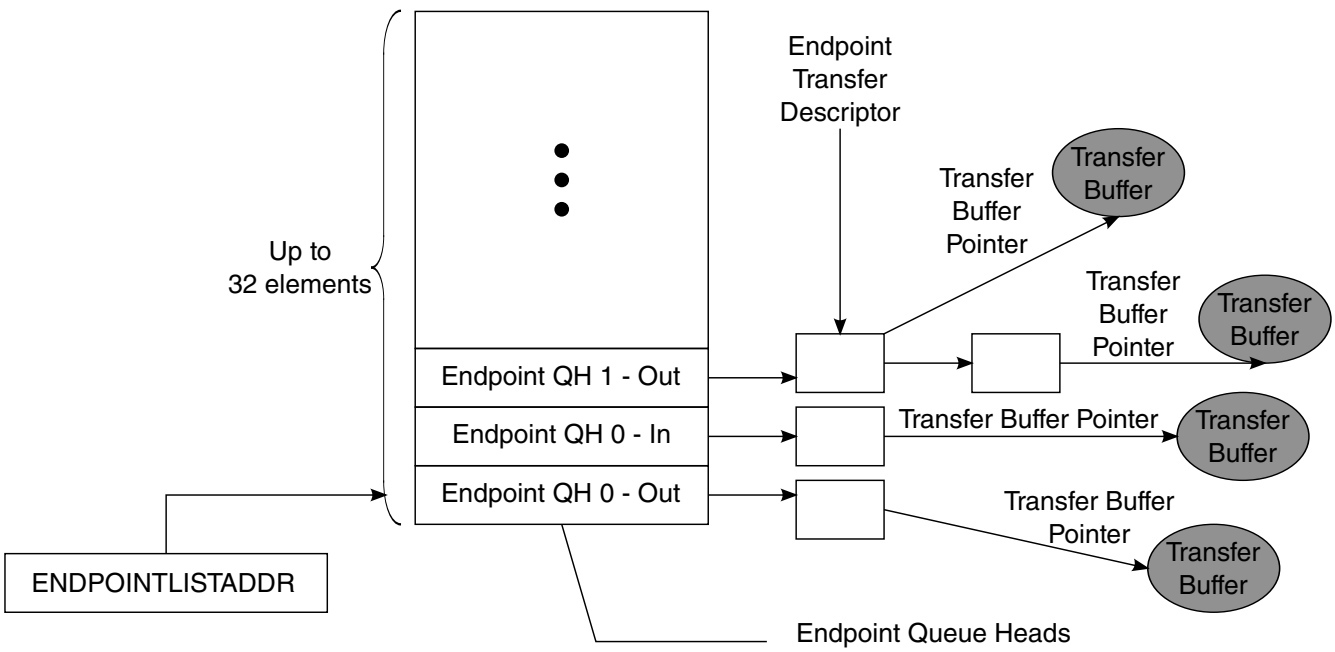


Figure 35-27. EndPoint Queue Head Organization

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.

35.5.5.1 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 35-57. 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

35.5.5.1.1 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 35-58 describes the endpoint capabilities.

Table 35-58. 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.

35.5.5.1.2 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.

35.5.5.1.3 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 35-59. 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.

35.5.5.1.4 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 35-60. 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.

35.5.5.2 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 35-61. 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 35-62. 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 35-63. 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 35-63. 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 35-64. 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.

35.5.6 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.

35.5.6.1 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.

35.5.6.2 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.

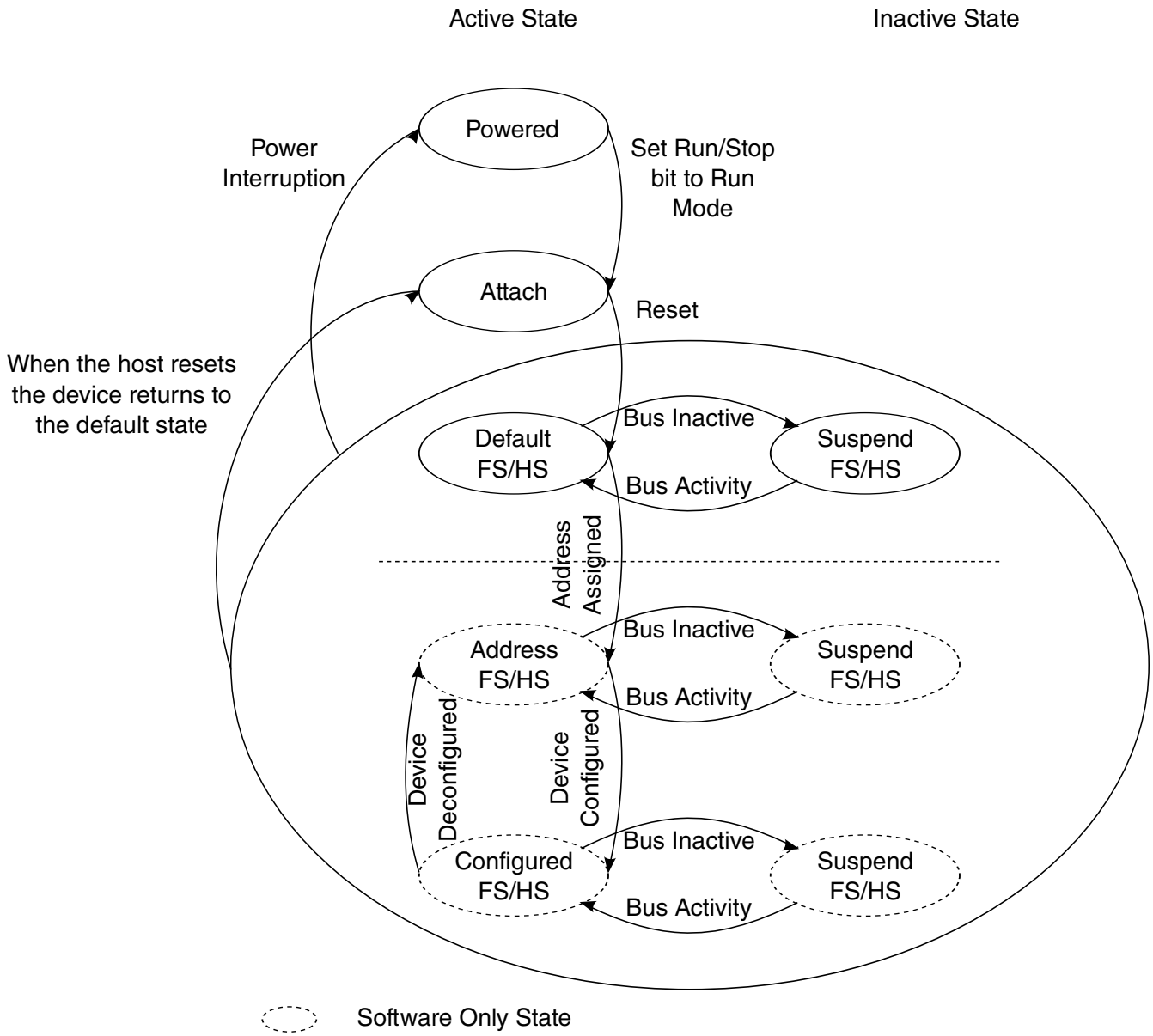


Figure 35-28. Device State Diagram

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 35-65. 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.

35.5.6.2.1 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.*

35.5.6.2.2 Suspend/Resume

The detials of suspend and resume are explained in these sections.

35.5.6.2.2.1 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.

NOTE

Review system level clocking issues defined in section (Ref: Signals-Clocking) for the clocking requirements of a suspended device controller.

35.5.6.2.2.2 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.

35.5.6.3 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.

35.5.6.3.1 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 35-66. Device Controller Endpoint Initialization

Field	Value
Data Toggle Reset	1
Data Toggle Inhibit	0
Endpoint Type	00 Control 01 Isochronous 10 Bulk

Table continues on the next page...

Table 35-66. Device Controller Endpoint Initialization (continued)

	11 Interrupt
Endpoint Stall	0

35.5.6.3.2 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 35-67. 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

Table continues on the next page...

Table 35-67. Device Controller Stall Response Matrix (continued)

IN/OUT/PING packet received by a control endpoint.	'0'	None.	ACK/ NAK/ NYET

35.5.6.3.3 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.

35.5.6.3.3.1 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.

35.5.6.3.3.2 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.

35.5.6.3.3.3 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 .

35.5.6.3.3.4 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.

35.5.6.4 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.

35.5.6.4.1 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 35-68. Variable Length Transfer Protocol Example (ZLT = 0)

Bytes (dTD)	Max. Packet Length (dQH)	N	P1	P2	P3
511	256	2	256	255	

Table continues on the next page...

Table 35-68. Variable Length Transfer Protocol Example (ZLT = 0) (continued)

512	256	3	256	256	0
512	512	2	512	0	

Table 35-69. 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.

35.5.6.4.1.1 Interrupt/Bulk Endpoint Bus Response Matrix

The table below shows the response matrix for Interrput/Bulk Endpoint Bus.

Table 35-70. 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.

35.5.6.4.2 Control Endpoint Operation Model

This section details the setup phase, data phase, status phase, and the control endpoint bus response matrix.

35.5.6.4.2.1 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.

35.5.6.4.2.2 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.

35.5.6.4.2.3 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.

35.5.6.4.2.4 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 35-71. 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

Table continues on the next page...

Table 35-71. Control Endpoint Bus Response Matrix (continued)

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.

35.5.6.4.3 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.

35.5.6.4.3.1 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.

35.5.6.4.3.2 Isochronous Endpoint Bus Response Matrix

The following table shows the response matrix for the Isochronous Endpoint Bus.

Table 35-72. 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

35.5.6.5 Managing Queue Heads

The following figure shows the End Point Queue Head.

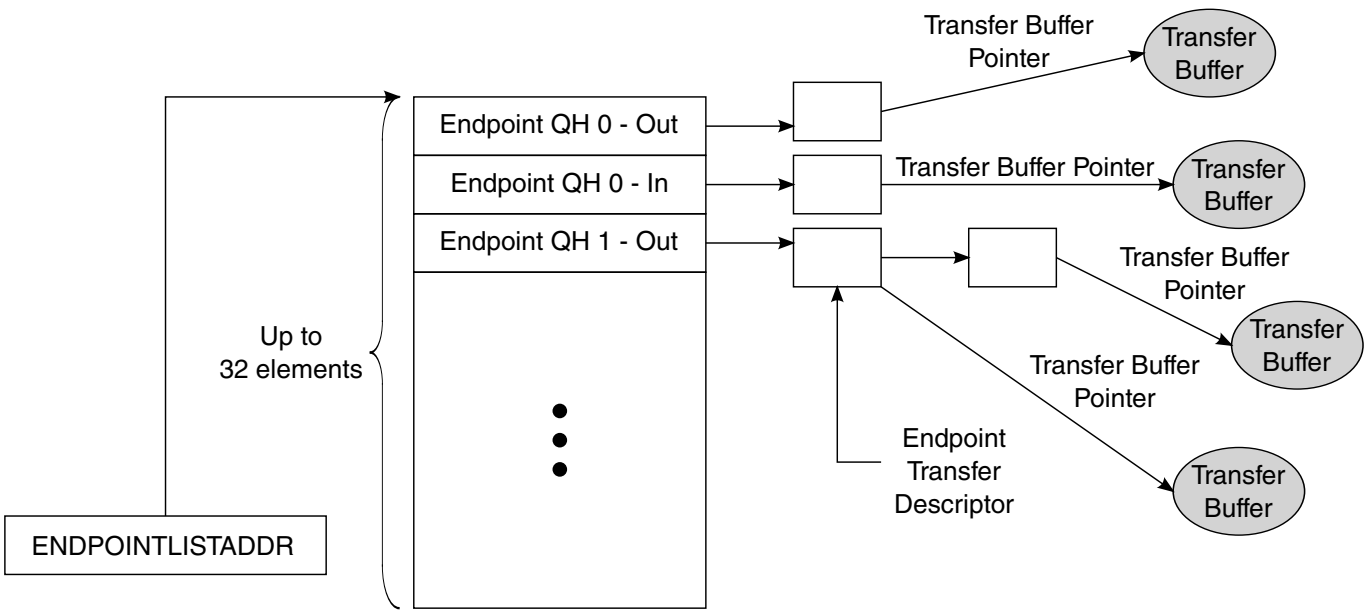


Figure 35-29. End Point Queue Head Diagram

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 35-29. 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.

35.5.6.5.1 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.

35.5.6.5.2 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.

35.5.6.6 Managing Transfers with Transfer Descriptors

35.5.6.6.1 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.

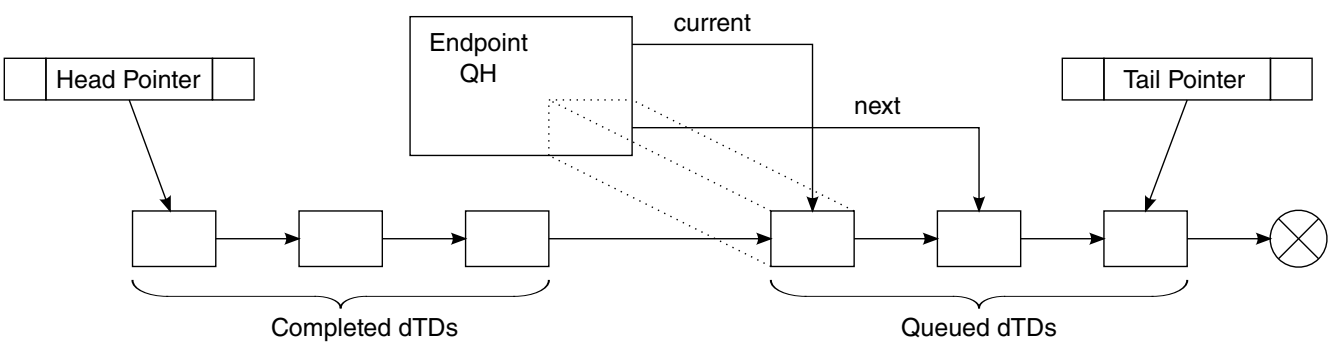


Figure 35-30. Software Link Pointers

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.

35.5.6.6.2 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.

35.5.6.6.3 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.

35.5.6.6.4 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.

35.5.6.6.5 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.

35.5.6.6.6 Device Error Matrix

The following table summarizes packet errors that are not automatically handled by the Device Controller.

Table 35-73. 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 35-74. 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.

35.5.6.7 Servicing Interrupts

The interrupt service routine must consider that there are high-frequency, low-frequency operations, and error operations and order accordingly.

35.5.6.7.1 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 35-75. 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 35-29 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 35-29 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.

35.5.6.7.2 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 35-76. 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.

35.5.6.7.3 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 35-77. 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.

35.6 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)
- USBNC_USB_" prefix in register name indicates it is a USB non-core register.

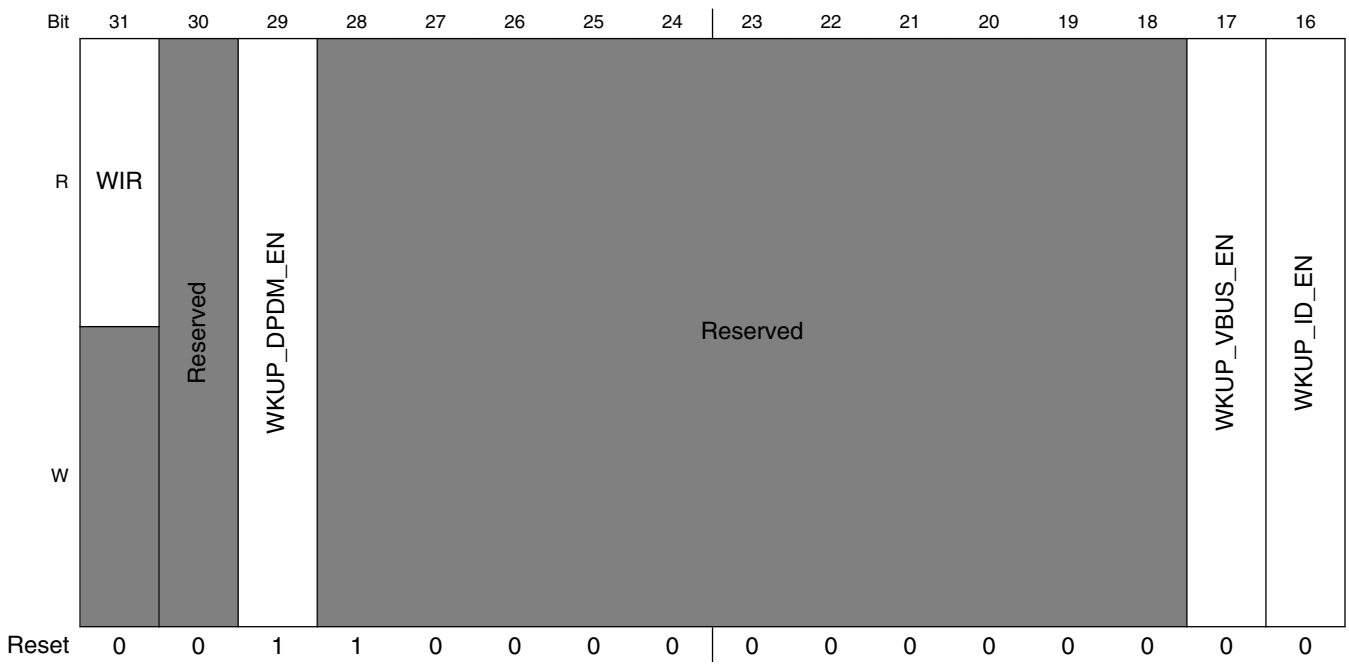
USBNC memory map

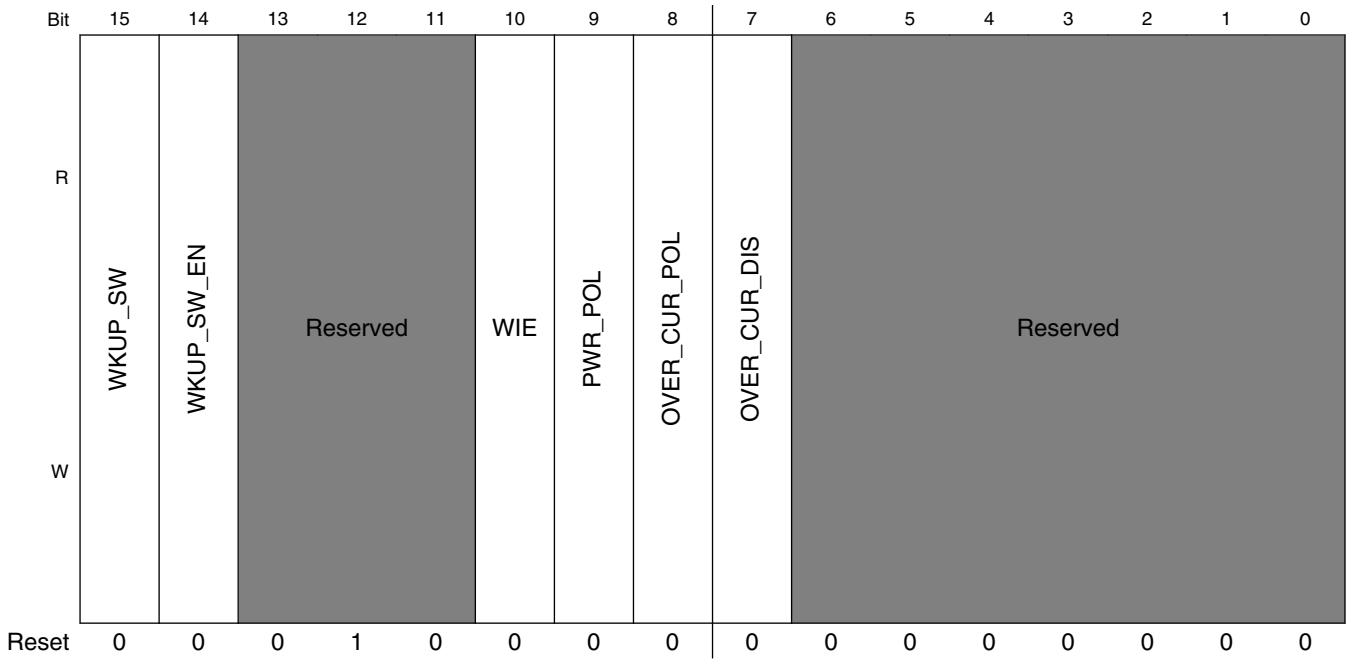
Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400E_4800	USB OTG1 Control Register (USBNC_USB_OTG1_CTRL)	32	R/W	3000_1000h	35.6.1/1298
400E_4818	OTG1 UTMI PHY Control 0 Register (USBNC_USB_OTG1_PHY_CTRL_0)	32	R/W	0000_0000h	35.6.2/1301

35.6.1 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: 400E_4000h base + 800h offset = 400E_4800h





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...

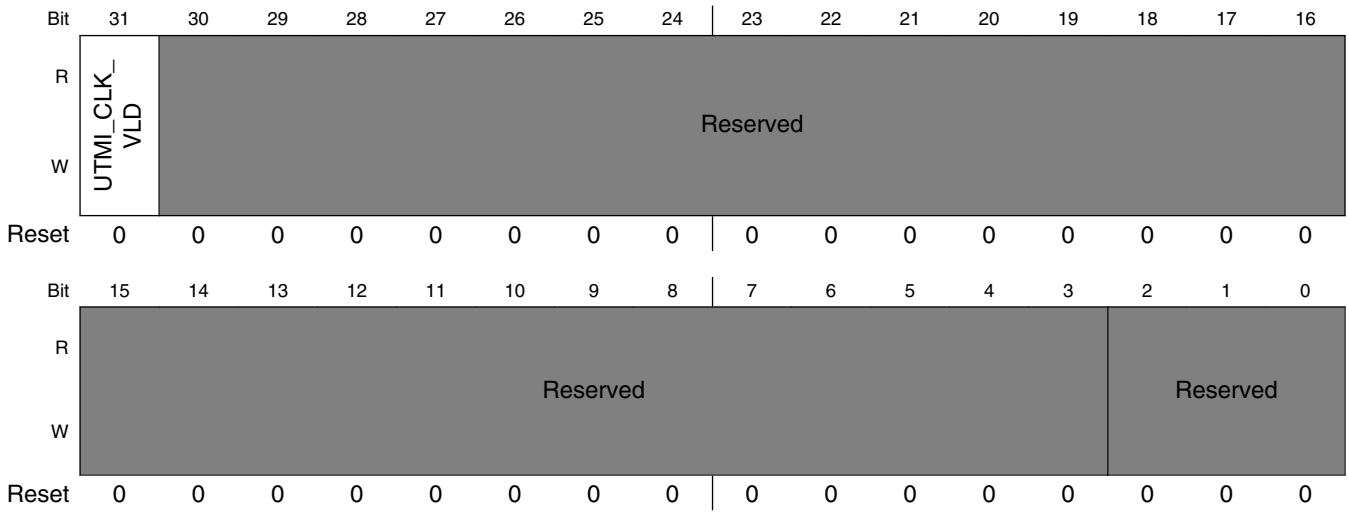
USBNC_USB_OTG1_CTRL field descriptions (continued)

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

35.6.2 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: 400E_4000h base + 818h offset = 400E_4818h



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

35.7 USB Core Memory Map/Register Definition

USB memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400E_4000	Identification register (USB_UOG1_ID)	32	R	E4A1_FA05h	35.7.1/1304

Table continues on the next page...

USB memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400E_4004	Hardware General (USB_UOG1_HWGENERAL)	32	R	0000_0035h	35.7.2/1304
400E_4008	Host Hardware Parameters (USB_UOG1_HWHOST)	32	R	1002_0001h	35.7.3/1306
400E_400C	Device Hardware Parameters (USB_UOG1_HWDEVICE)	32	R	0000_0011h	35.7.4/1306
400E_4010	TX Buffer Hardware Parameters (USB_UOG1_HWTXBUF)	32	R	8008_0B08h	35.7.5/1307
400E_4014	RX Buffer Hardware Parameters (USB_UOG1_HWRXBUF)	32	R	0000_0808h	35.7.6/1308
400E_4080	General Purpose Timer #0 Load (USB_UOG1_GPTIMER0LD)	32	R/W	0000_0000h	35.7.7/1308
400E_4084	General Purpose Timer #0 Controller (USB_UOG1_GPTIMER0CTRL)	32	R/W	0000_0000h	35.7.8/1309
400E_4088	General Purpose Timer #1 Load (USB_UOG1_GPTIMER1LD)	32	R/W	0000_0000h	35.7.9/1310
400E_408C	General Purpose Timer #1 Controller (USB_UOG1_GPTIMER1CTRL)	32	R/W	0000_0000h	35.7.10/ 1311
400E_4090	System Bus Config (USB_UOG1_SBUSCFG)	32	R/W	0000_0002h	35.7.11/ 1312
400E_4100	Capability Registers Length (USB_UOG1_CAPLENGTH)	8	R	40h	35.7.12/ 1313
400E_4102	Host Controller Interface Version (USB_UOG1_HCIVERSION)	16	R	0100h	35.7.13/ 1313
400E_4104	Host Controller Structural Parameters (USB_UOG1_HCSPARAMS)	32	R	0001_0011h	35.7.14/ 1314
400E_4108	Host Controller Capability Parameters (USB_UOG1_HCCPARAMS)	32	R	0000_0006h	35.7.15/ 1316
400E_4120	Device Controller Interface Version (USB_UOG1_DCIVERSION)	16	R	0001h	35.7.16/ 1318
400E_4124	Device Controller Capability Parameters (USB_UOG1_DCCPARAMS)	32	R	0000_0188h	35.7.17/ 1319
400E_4140	USB Command Register (USB_UOG1_USBCMD)	32	R/W	0008_0000h	35.7.18/ 1320
400E_4144	USB Status Register (USB_UOG1_USBSTS)	32	R/W	0000_0000h	35.7.19/ 1324
400E_4148	Interrupt Enable Register (USB_UOG1_USBINTR)	32	R/W	0000_0000h	35.7.20/ 1328
400E_414C	USB Frame Index (USB_UOG1_FRINDEX)	32	R/W	0000_0000h	35.7.21/ 1330
400E_4154	Frame List Base Address (USB_UOG1_PERIODICLISTBASE)	32	R/W	0000_0000h	35.7.22/ 1331
400E_4154	Device Address (USB_UOG1_DEVICEADDR)	32	R/W	0000_0000h	35.7.23/ 1331
400E_4158	Next Asynch. Address (USB_UOG1_ASYNCLISTADDR)	32	R/W	0000_0000h	35.7.24/ 1332
400E_4158	Endpoint List Address (USB_UOG1_ENDPTLISTADDR)	32	R/W	0000_0000h	35.7.25/ 1333

Table continues on the next page...

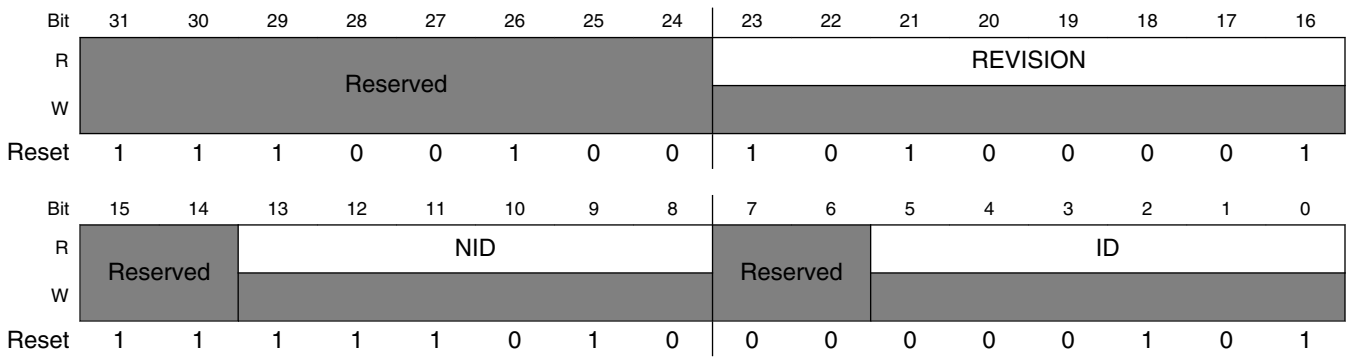
USB memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400E_4160	Programmable Burst Size (USB_UOG1_BURSTSIZE)	32	R/W	0000_0808h	35.7.26/ 1333
400E_4164	TX FIFO Fill Tuning (USB_UOG1_TXFILLTUNING)	32	R/W	0000_0000h	35.7.27/ 1334
400E_4178	Endpoint NAK (USB_UOG1_ENDPTNAK)	32	R/W	0000_0000h	35.7.28/ 1336
400E_417C	Endpoint NAK Enable (USB_UOG1_ENDPTNAKEN)	32	R/W	0000_0000h	35.7.29/ 1336
400E_4180	Configure Flag Register (USB_UOG1_CONFIGFLAG)	32	R/W	0000_0001h	35.7.30/ 1337
400E_4184	Port Status & Control (USB_UOG1_PORTSC1)	32	R/W	1000_0000h	35.7.31/ 1337
400E_41A4	On-The-Go Status & control (USB_UOG1_OTGSC)	32	R/W	0000_1120h	35.7.32/ 1344
400E_41A8	USB Device Mode (USB_UOG1_USBMODE)	32	R/W	0000_5000h	35.7.33/ 1348
400E_41AC	Endpoint Setup Status (USB_UOG1_ENDPTSETUPSTAT)	32	R/W	0000_0000h	35.7.34/ 1349
400E_41B0	Endpoint Prime (USB_UOG1_ENDPTPRIME)	32	R/W	0000_0000h	35.7.35/ 1350
400E_41B4	Endpoint Flush (USB_UOG1_ENDPTFLUSH)	32	R/W	0000_0000h	35.7.36/ 1351
400E_41B8	Endpoint Status (USB_UOG1_ENDPTSTAT)	32	R	0000_0000h	35.7.37/ 1351
400E_41BC	Endpoint Complete (USB_UOG1_ENDPTCOMPLETE)	32	R/W	0000_0000h	35.7.38/ 1352
400E_41C0	Endpoint Control0 (USB_UOG1_ENDPTCTRL0)	32	R/W	0080_0080h	35.7.39/ 1353
400E_41C4	Endpoint Control 1 (USB_UOG1_ENDPTCTRL1)	32	R/W	0000_0000h	35.7.40/ 1355
400E_41C8	Endpoint Control 2 (USB_UOG1_ENDPTCTRL2)	32	R/W	0000_0000h	35.7.41/ 1358
400E_41CC	Endpoint Control 3 (USB_UOG1_ENDPTCTRL3)	32	R/W	0000_0000h	35.7.42/ 1360
400E_41D0	Endpoint Control 4 (USB_UOG1_ENDPTCTRL4)	32	R/W	0000_0000h	35.7.43/ 1363
400E_41D4	Endpoint Control 5 (USB_UOG1_ENDPTCTRL5)	32	R/W	0000_0000h	35.7.44/ 1366
400E_41D8	Endpoint Control 6 (USB_UOG1_ENDPTCTRL6)	32	R/W	0000_0000h	35.7.45/ 1369
400E_41DC	Endpoint Control 7 (USB_UOG1_ENDPTCTRL7)	32	R/W	0000_0000h	35.7.46/ 1372

35.7.1 Identification register (USB_*n*ID)

The ID register identifies the USB 2.0 High-Speed core and its revision.

Address: 400E_4000h base + 0h offset + (65536d × i), where i=0d to 0d



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.

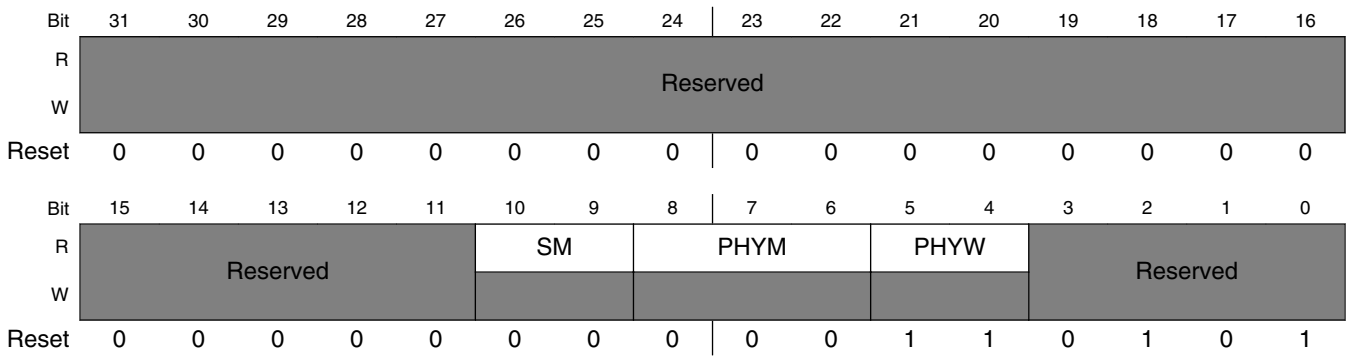
35.7.2 Hardware General (USB_*n*HWGENERAL)

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: 400E_4000h base + 4h offset + (65536d × i), where i=0d to 0d

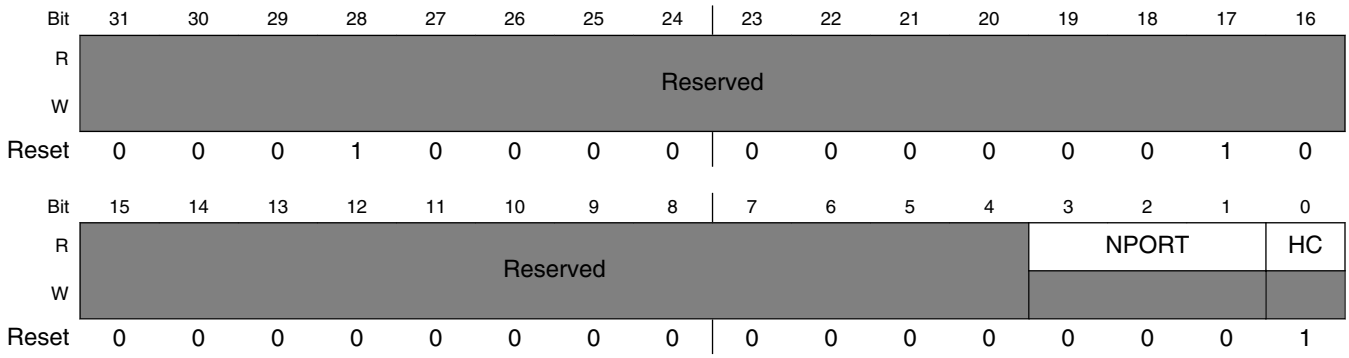


USB_*n*HWGENERAL field descriptions

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

35.7.3 Host Hardware Parameters (USB_*n*HWHOST)

Address: 400E_4000h base + 8h offset + (65536d × i), where i=0d to 0d

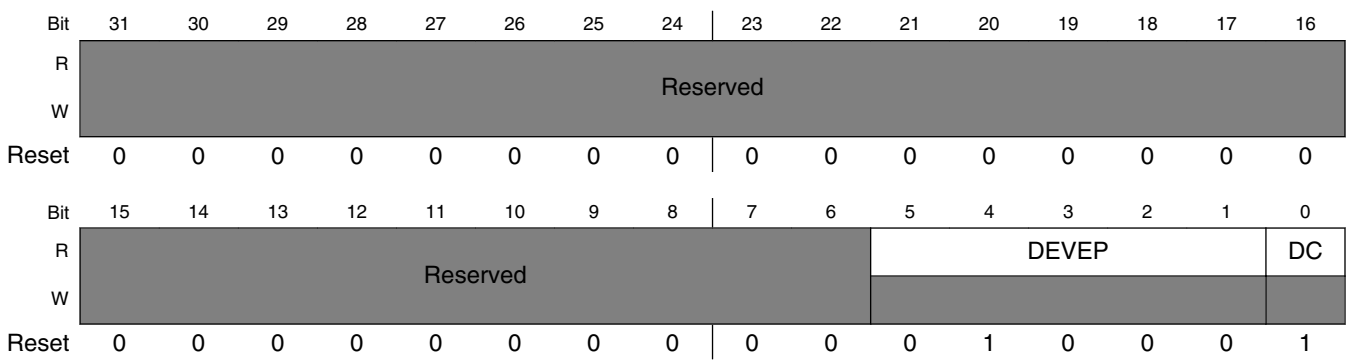


USB_*n*HWHOST field descriptions

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

35.7.4 Device Hardware Parameters (USB_*n*HWDEVICE)

Address: 400E_4000h base + Ch offset + (65536d × i), where i=0d to 0d

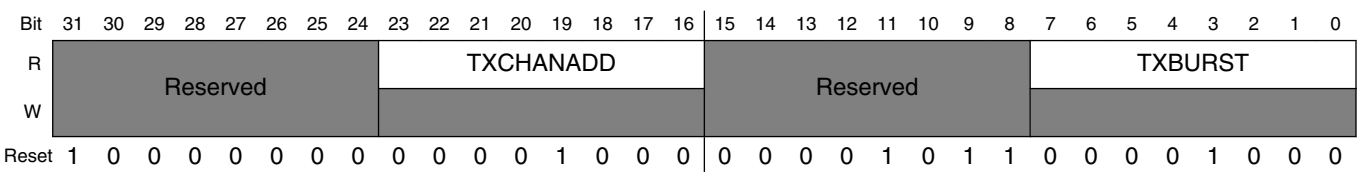


USB_*n*HWDEVICE field descriptions

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

35.7.5 TX Buffer Hardware Parameters (USB_*n*HWTXBUF)

Address: 400E_4000h base + 10h offset + (65536d × i), where i=0d to 0d

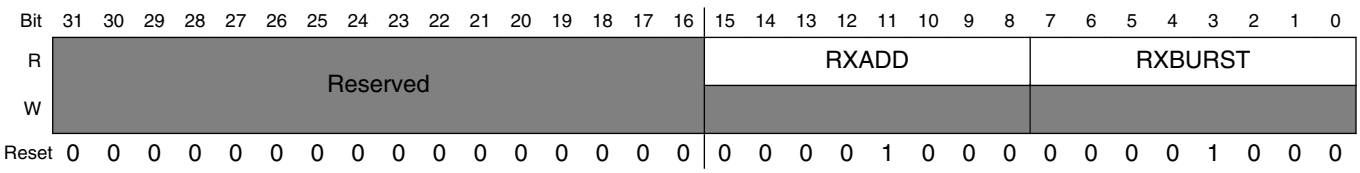


USB_*n*HWTXBUF field descriptions

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) .

35.7.6 RX Buffer Hardware Parameters (USB_*n*HWRXBUF)

Address: 400E_4000h base + 14h offset + (65536d × i), where i=0d to 0d



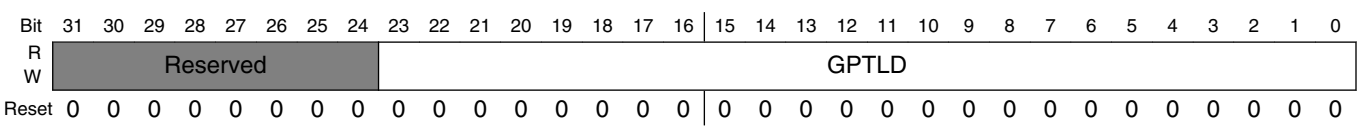
USB_*n*HWRXBUF field descriptions

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) .

35.7.7 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: 400E_4000h base + 80h offset + (65536d × i), where i=0d to 0d



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'.

Table continues on the next page...

USB_*n*GPTIMER0LD field descriptions (continued)

Field	Description
	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.

35.7.8 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: 400E_4000h base + 84h offset + (65536d × i), where i=0d to 0d

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.

35.7.9 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: 400E_4000h base + 88h offset + (65536d × i), where i=0d to 0d

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.

35.7.10 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: 400E_4000h base + 8Ch offset + (65536d × i), where i=0d to 0d

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.

35.7.11 System Bus Config (USB_*n*SBUSCFG)

Address: 400E_4000h base + 90h offset + (65536d × i), where i=0d to 0d

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

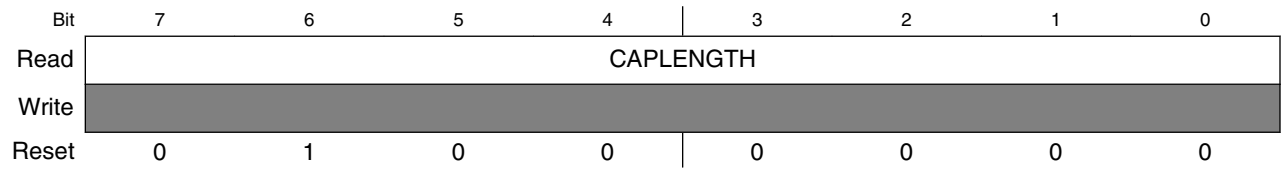
USB_*n*SBUSCFG field descriptions

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

35.7.12 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: 400E_4000h base + 100h offset + (65536d × i), where i=0d to 0d



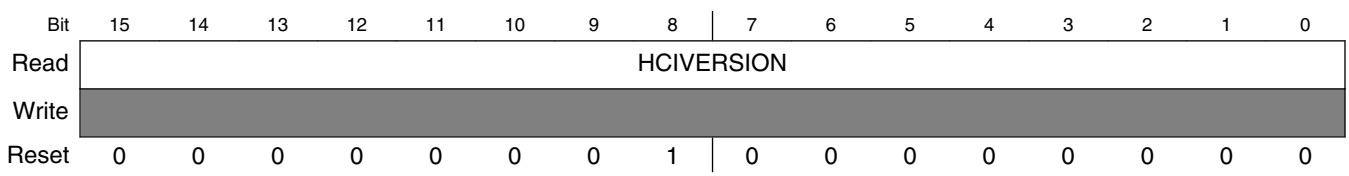
USB_*n*CAPLENGTH field descriptions

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'.

35.7.13 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: 400E_4000h base + 102h offset + (65536d × i), where i=0d to 0d



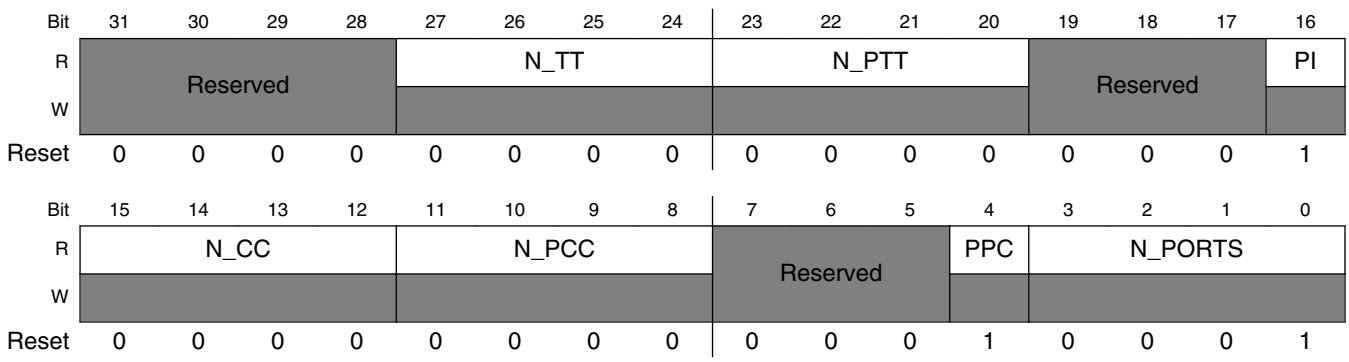
USB_*n*HCIVERSION field descriptions

Field	Description
HCIVERSION	Host Controller Interface Version Number Default value is '10h', which means EHCI rev1.0.

35.7.14 Host Controller Structural Parameters (USB_*n*HCSPARAMS)

The following figure shows the port steering logic capabilities of Host Control Structural Parameters (n_HCSPARAMS).

Address: 400E_4000h base + 104h offset + (65536d × i), where i=0d to 0d



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)

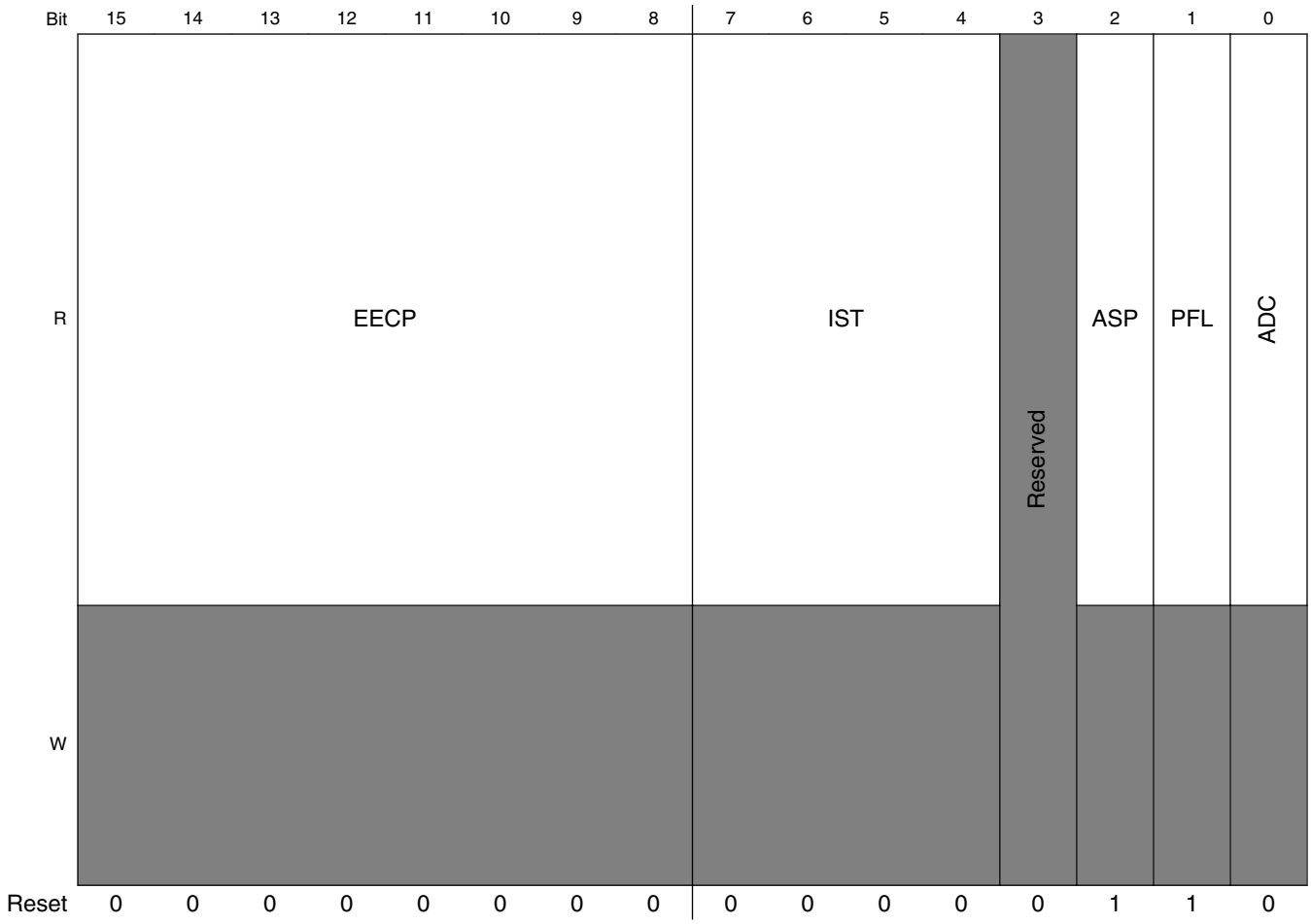
Field	Description
11–8 N_PCC	Number of Ports per Companion Controller This field indicates the number of ports supported per internal Companion Controller. It is used to indicate the port routing configuration to the system software. For example, if N_PORTS has a value of 6 and N_CC has a value of 2 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.

35.7.15 Host Controller Capability Parameters (USB_*n*HCCPARAMS)

This register identifies multiple mode control (time-base bit functionality), addressing capability.

Address: 400E_4000h base + 108h offset + (65536d × i), where i=0d to 0d

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.

35.7.16 Device Controller Interface Version (USB_*n*DCIVERSION)

This register indicates the two-byte BCD encoding of the device controller interface version number.

Address: 400E_4000h base + 120h offset + (65536d × i), where i=0d to 0d

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.

35.7.17 Device Controller Capability Parameters (USB_*n*DCCPARAMS)

These fields describe the overall device capability of the controller.

Address: 400E_4000h base + 124h offset + (65536d × i), where i=0d to 0d

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							HC	DC	Reserved		DEN				
W																
Reset	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0

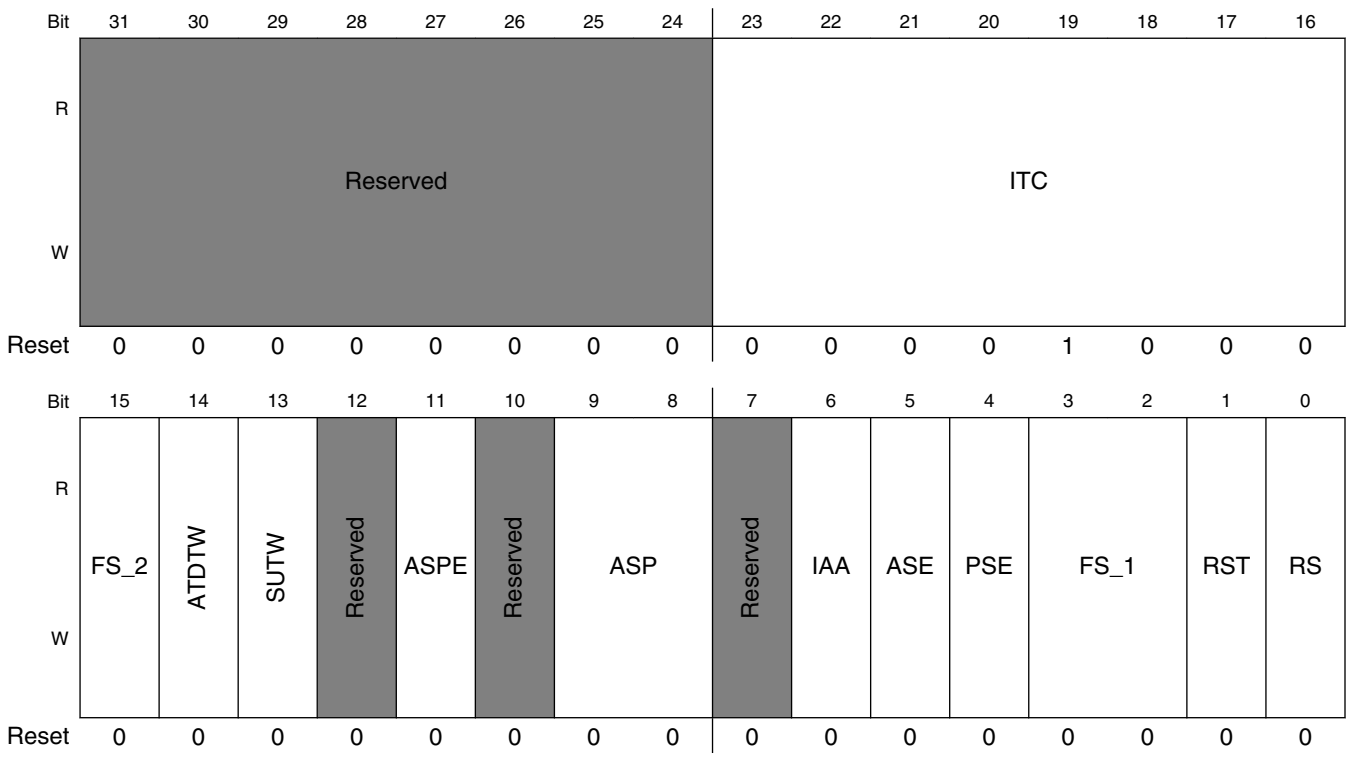
USB_*n*DCCPARAMS field descriptions

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.

35.7.18 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: 400E_4000h base + 140h offset + (65536d × i), where i=0d to 0d



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...

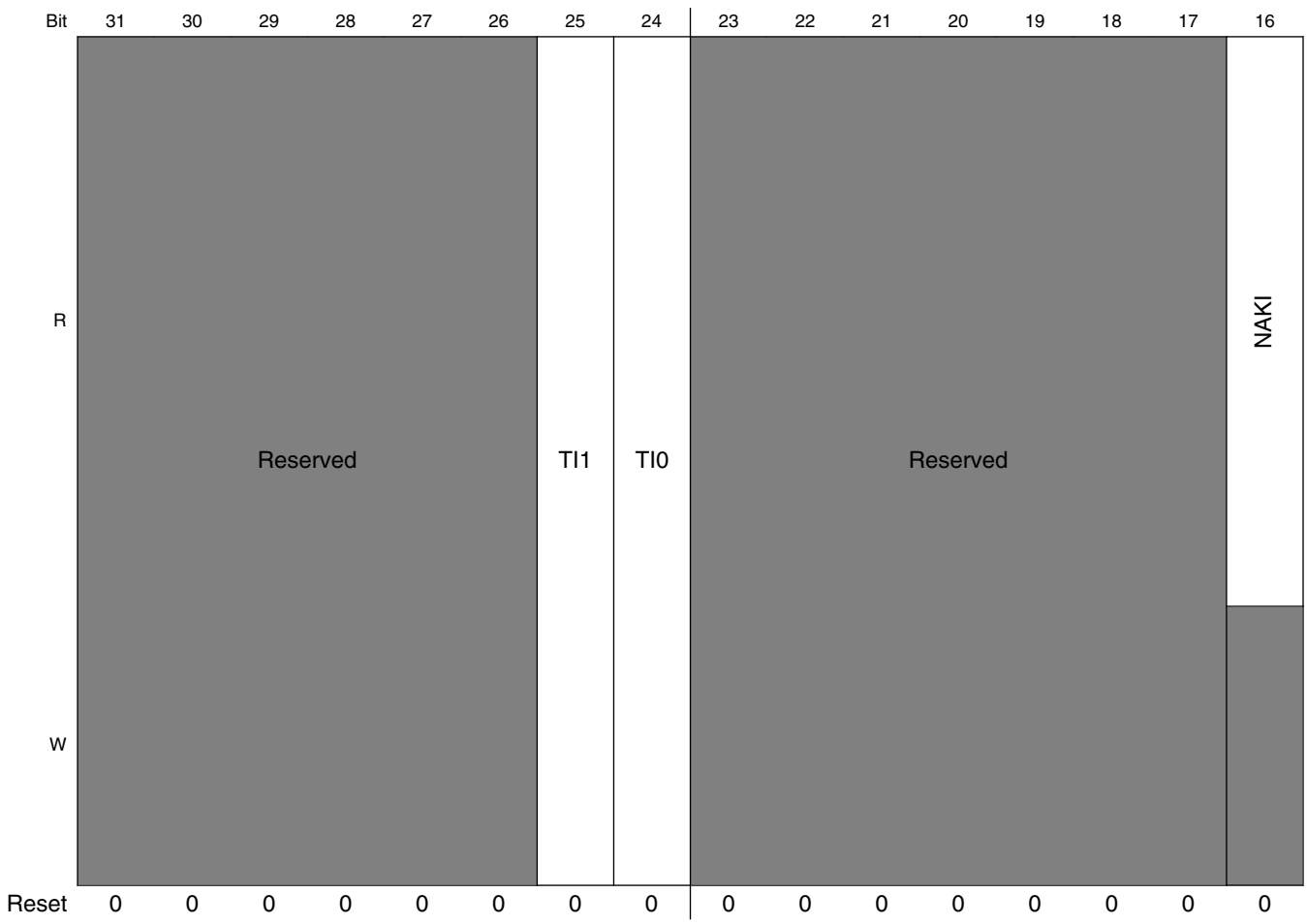
USB_*n*USBCMD field descriptions (continued)

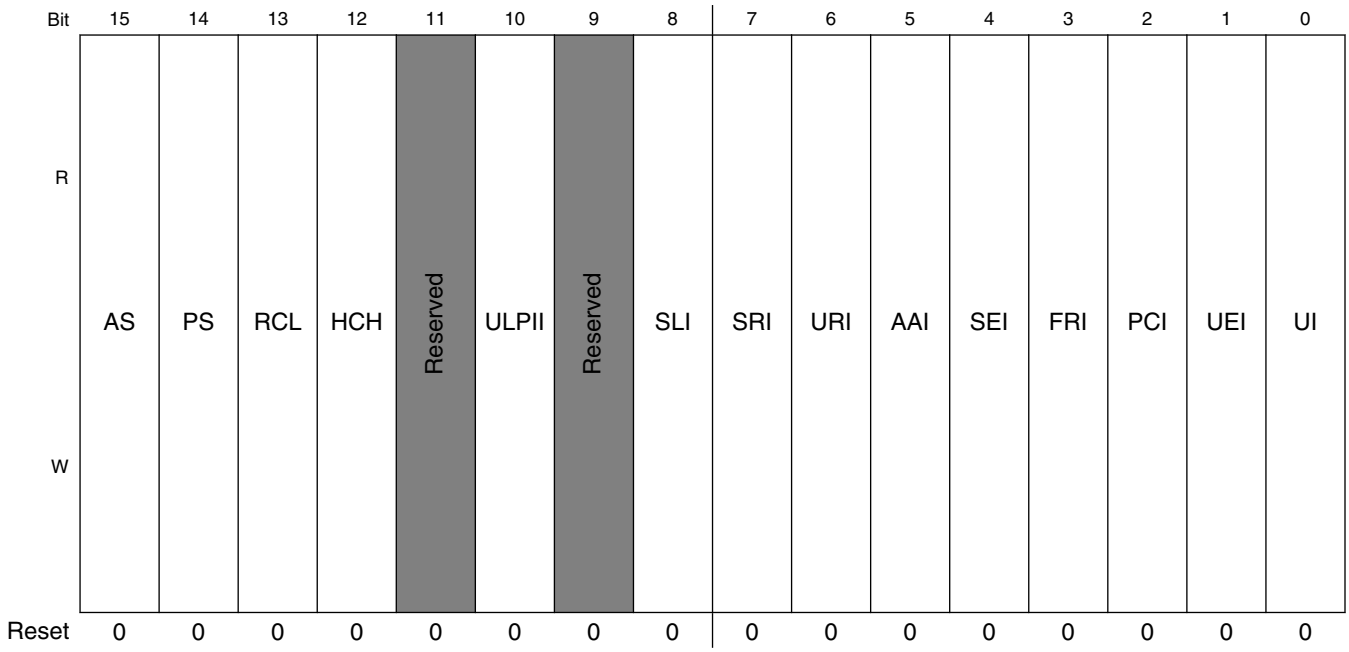
Field	Description
	When software writes a one to this bit, the Controller resets its internal pipelines, timers, counters, state machines etc. to their initial value. Writing a one to this bit when the device is in the attached state is not recommended, because the effect on an attached host is undefined. 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.

35.7.19 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: 400E_4000h base + 144h offset + (65536d × i), where i=0d to 0d





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...

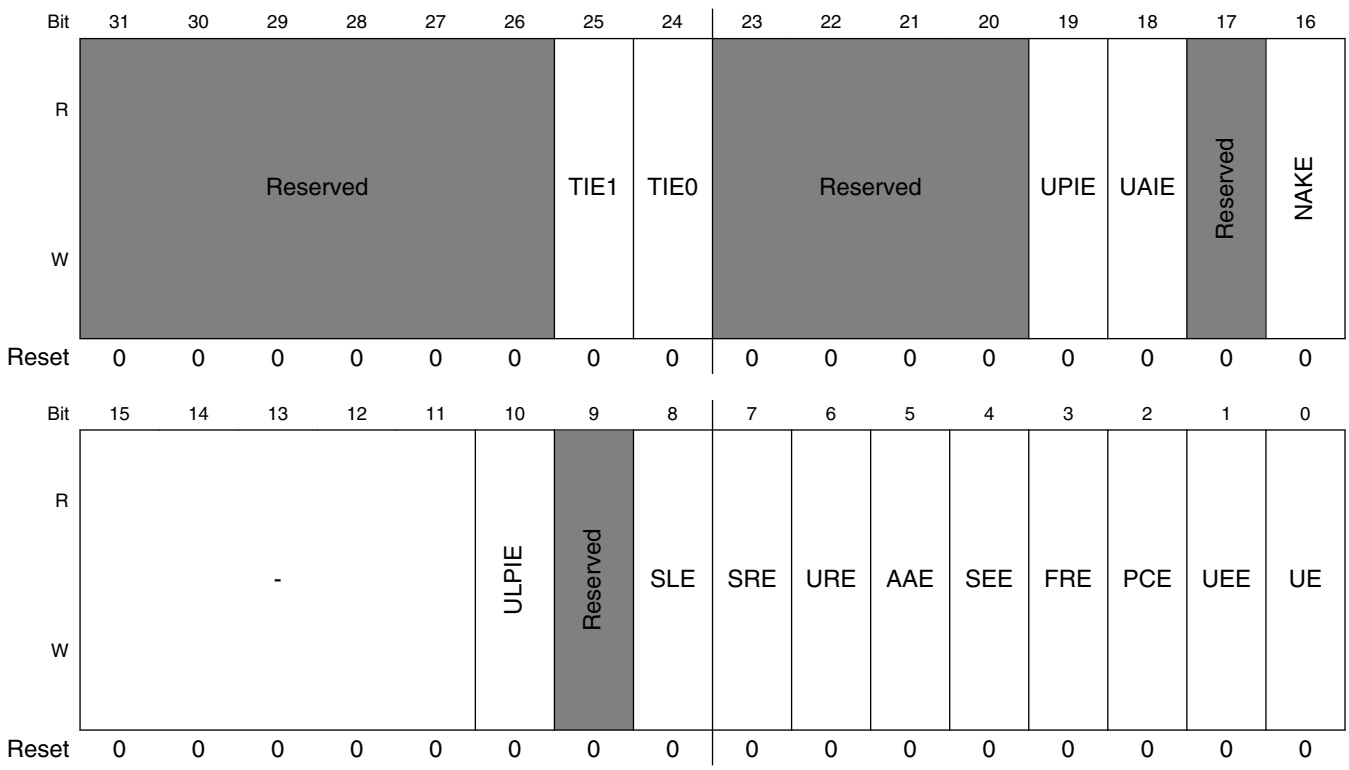
USB_*n*USBSTS field descriptions (continued)

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.

35.7.20 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: 400E_4000h base + 148h offset + (65536d × i), where i=0d to 0d



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.

35.7.21 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: 400E_4000h base + 14Ch offset + (65536d × i), where i=0d to 0d

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)

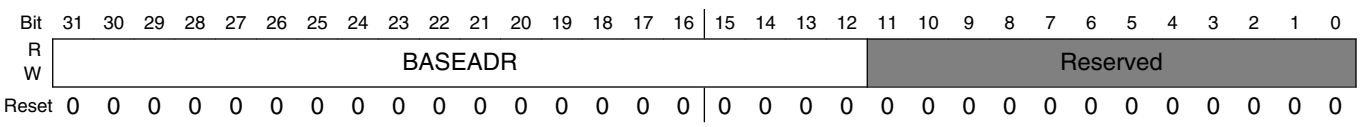
Field	Description
	010 (256) 10 011 (128) 9 100 (64) 8 101 (32) 7 110 (16) 6 111 (8) 5

35.7.22 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: 400E_4000h base + 154h offset + (65536d × i), where i=0d to 0d



USB_*n*PERIODICLISTBASE field descriptions

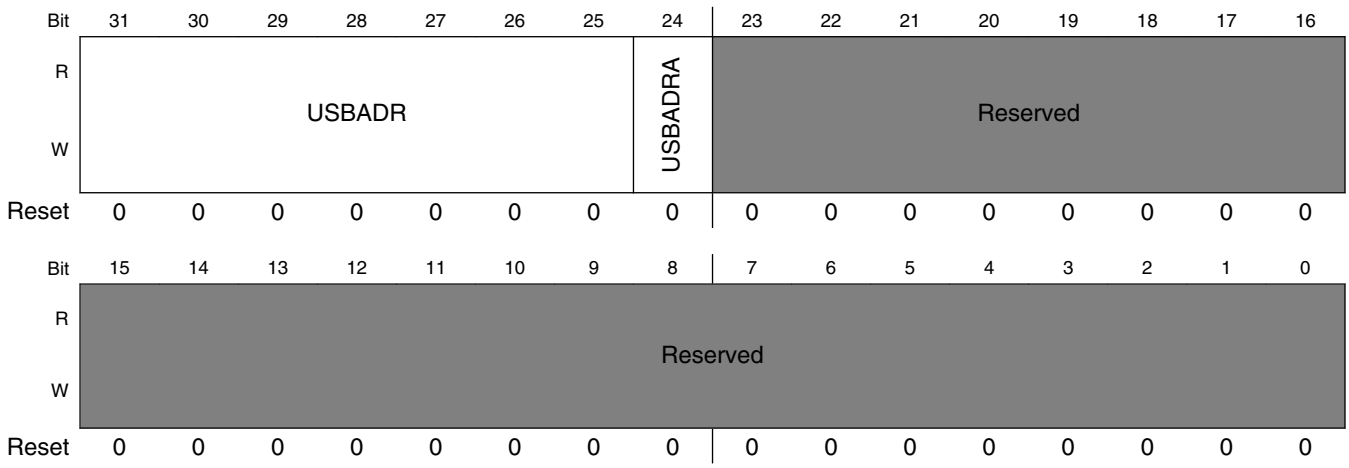
Field	Description
31–12 BASEADR	Base Address (Low). These bits correspond to memory address signals [31:12], respectively. Only used by the host controller.
-	This field is reserved. Reserved

35.7.23 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: 400E_4000h base + 154h offset + (65536d × i), where i=0d to 0d



USB_*n*DEVICEADDR field descriptions

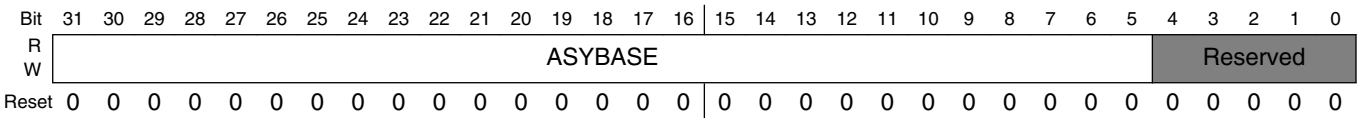
Field	Description
31–25 USBADR	Device Address. These bits correspond to the USB device address
24 USBADRA	Device Address Advance. Default=0. When this bit is '0', any writes to USBADR are instantaneous. When this bit is written to a '1' at the same time or before USBADR is written, the write to the USBADR 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

35.7.24 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: 400E_4000h base + 158h offset + (65536d × i), where i=0d to 0d



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

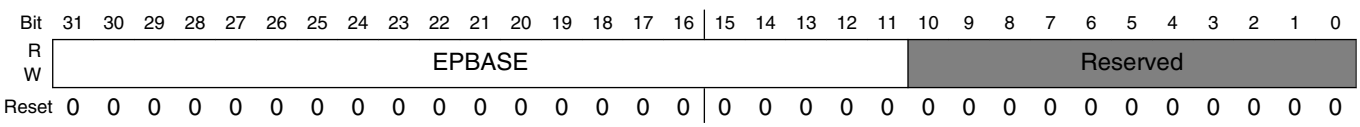
35.7.25 Endpoint List Address (USB_*n*ENDPTLISTADDR)

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: 400E_4000h base + 158h offset + (65536d × i), where i=0d to 0d



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

35.7.26 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: 400E_4000h base + 160h offset + (65536d × i), where i=0d to 0d

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.

35.7.27 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

$T_p = T_{ff} + T_0 + T_1$

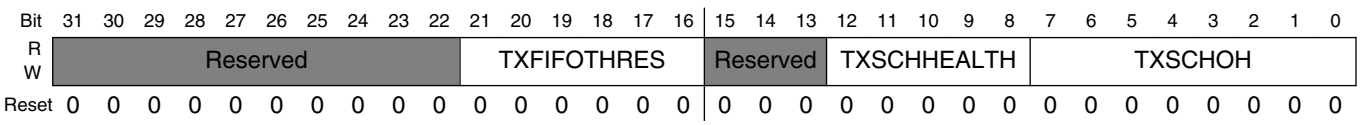
Upon discovery of a transmit (OUT/SETUP) packet in the data structures, host controller checks to ensure T_p remains before the end of the [micro]frame. If so it proceeds to pre-fill the TX FIFO. If at anytime during the pre-fill operation the time remaining the [micro]frame is < T_s then the packet 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: 400E_4000h base + 164h offset + (65536d × i), where i=0d to 0d

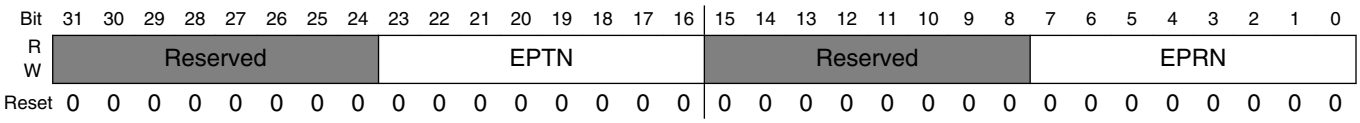


USB_*n*TXFILLTUNING field descriptions

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.

35.7.28 Endpoint NAK (USB_*n*ENDPTNAK)

Address: 400E_4000h base + 178h offset + (65536d × i), where i=0d to 0d

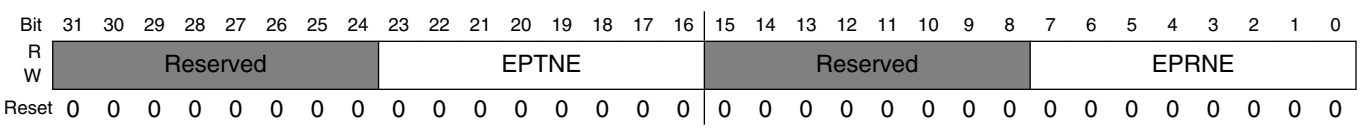


USB_*n*ENDPTNAK field descriptions

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

35.7.29 Endpoint NAK Enable (USB_*n*ENDPTNAKEN)

Address: 400E_4000h base + 17Ch offset + (65536d × i), where i=0d to 0d



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

Table continues on the next page...

USB_*n*ENDPTNAKEN field descriptions (continued)

Field	Description
EPRNE	RX Endpoint NAK Enable - R/W. 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

35.7.30 Configure Flag Register (USB_*n*CONFIGFLAG)

Address: 400E_4000h base + 180h offset + (65536d × i), where i=0d to 0d

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.

35.7.31 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: 400E_4000h base + 184h offset + (65536d × i), where i=0d to 0d

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	PTS_1		STS	PTW	PSPD		PTS_2	PFSC	PHCD	WKOC	WKDC	WKCN	PTC			
Reset	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	PIC		PO	PP	LS		HSP	PR	SUSP	FPR	OCC	OCA	PEC	PE	CSC	CCS
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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 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.

35.7.32 On-The-Go Status & control (USB_*n*OTGSC)

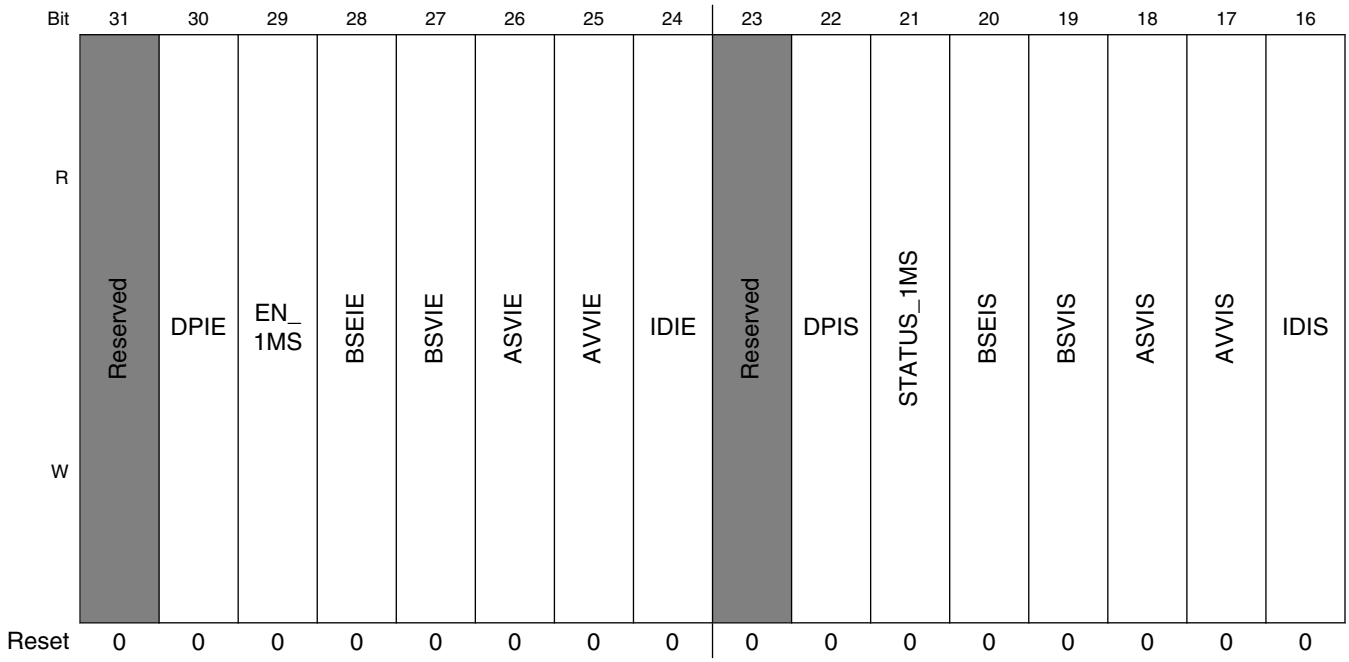
This register is availabe only in OTG controller core. It has four sections:

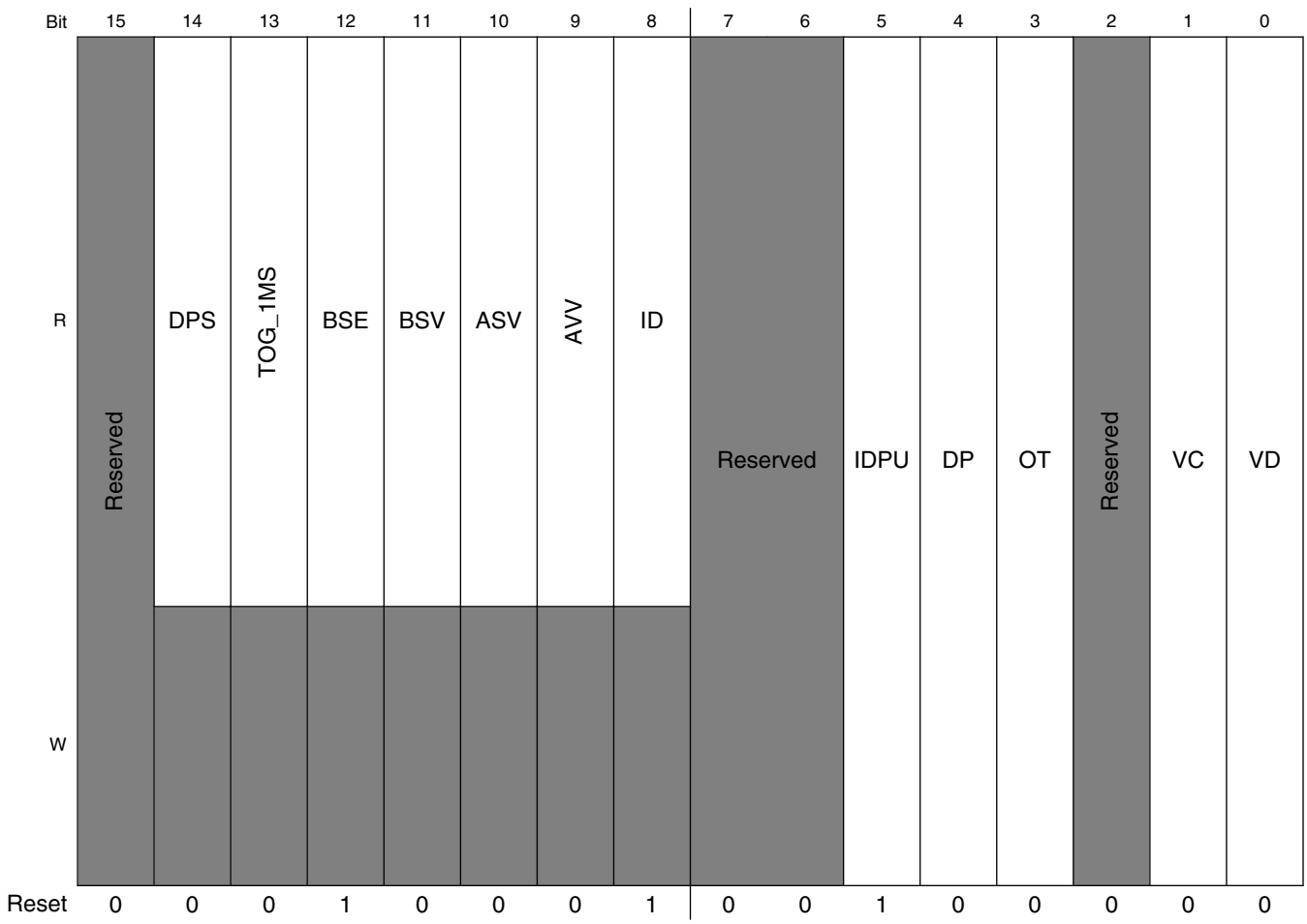
- OTG Interrupt enables (Read/Write)
- OTG Interrupt status (Read/Write to Clear)
- OTG Status inputs (Read Only)
- OTG Controls (Read/Write)

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: 400E_4000h base + 1A4h offset + (65536d × i), where i=0d to 0d





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.

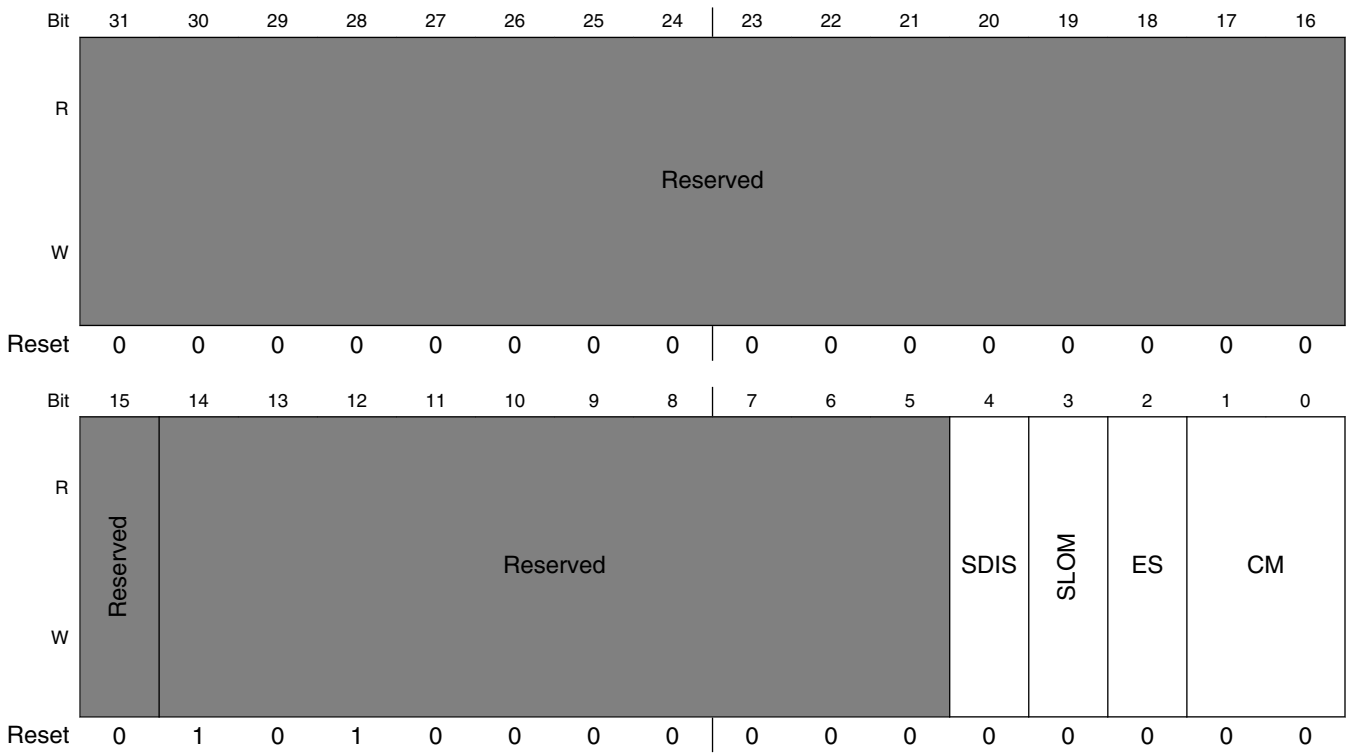
Table continues on the next page...

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.

35.7.33 USB Device Mode (USB_*n*USBMODE)

Address: 400E_4000h base + 1A8h offset + (65536d × i), where i=0d to 0d



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]

35.7.34 Endpoint Setup Status (USB_*n*ENDPTSETUPSTAT)

Address: 400E_4000h base + 1ACh offset + (65536d × i), where i=0d to 0d

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

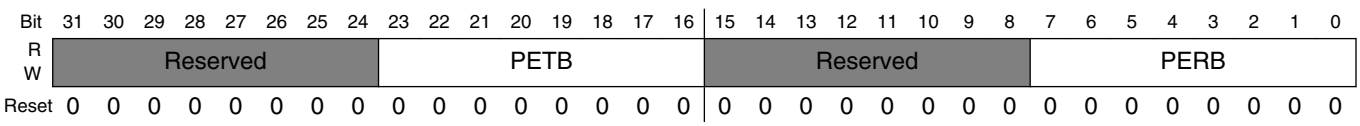
Field	Description
31–16 -	This field is reserved. Reserved
ENDPTSETUPSTAT	Setup Endpoint Status. For every setup transaction that is received, a corresponding bit in this register is set to one. Software must clear or acknowledge the setup transfer by writing a one to a respective bit after it has read 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.

35.7.35 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: 400E_4000h base + 1B0h offset + (65536d × i), where i=0d to 0d



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

35.7.36 Endpoint Flush (USB_*n*ENDPTFLUSH)

This register is only used in device mode.

Address: 400E_4000h base + 1B4h offset + (65536d × i), where i=0d to 0d

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\

35.7.37 Endpoint Status (USB_*n*ENDPTSTAT)

This register is only used in device mode.

Address: 400E_4000h base + 1B8h offset + (65536d × i), where i=0d to 0d

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

35.7.38 Endpoint Complete (USB_*n*ENDPTCOMPLETE)

This register is only used in device mode.

Address: 400E_4000h base + 1BCh offset + (65536d × i), where i=0d to 0d

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

35.7.39 Endpoint Control0 (USB_*n*ENDPTCTRL0)

Every Device implements Endpoint 0 as a control endpoint.

Address: 400E_4000h base + 1C0h offset + (65536d × i), where i=0d to 0d

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.

35.7.40 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: 400E_4000h base + 1C4h offset + (65536d × i), where i=0d to 0d

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.

35.7.41 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: 400E_4000h base + 1C8h offset + (65536d × i), where i=0d to 0d

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*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.

35.7.42 Endpoint Control 3 (USB_*n*ENDPTCTRL3)

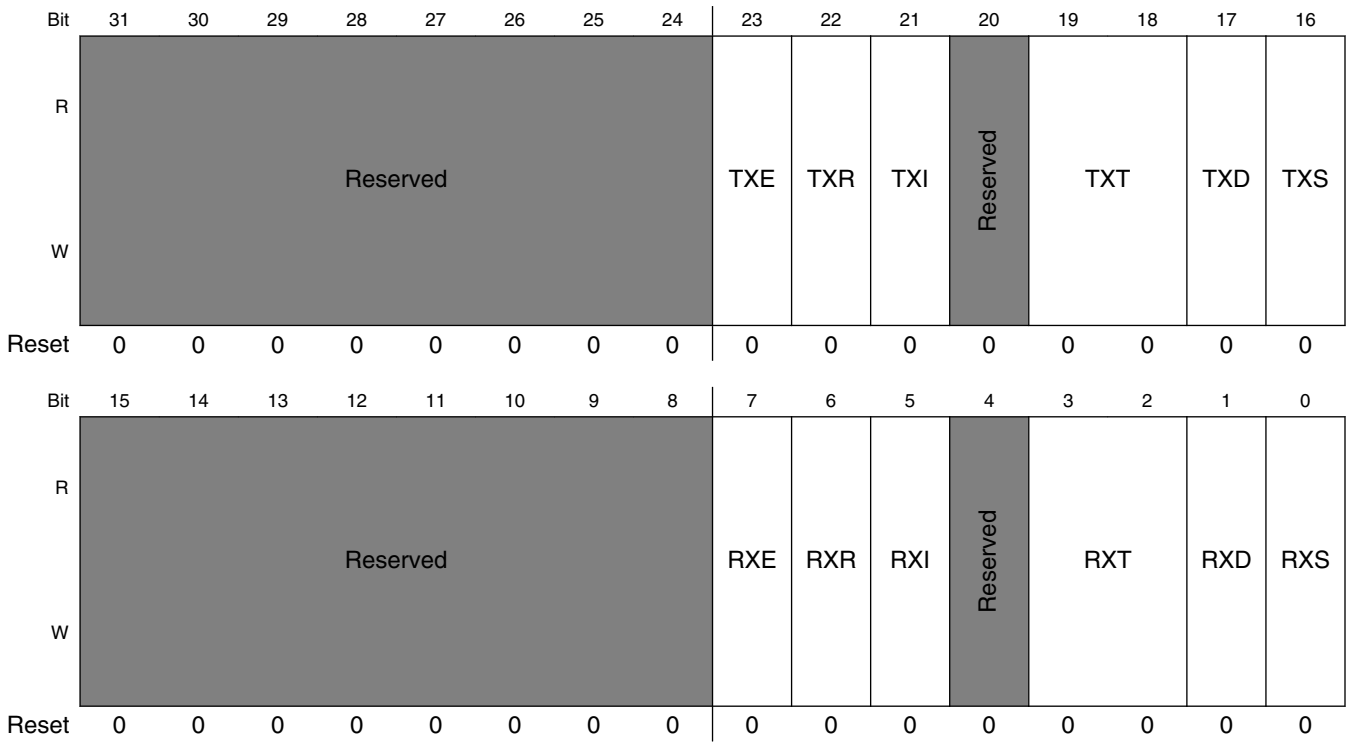
This is endpoint control register for endpoint 3 in device operation mode.

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: 400E_4000h base + 1CCh offset + (65536d × i), where i=0d to 0d



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.

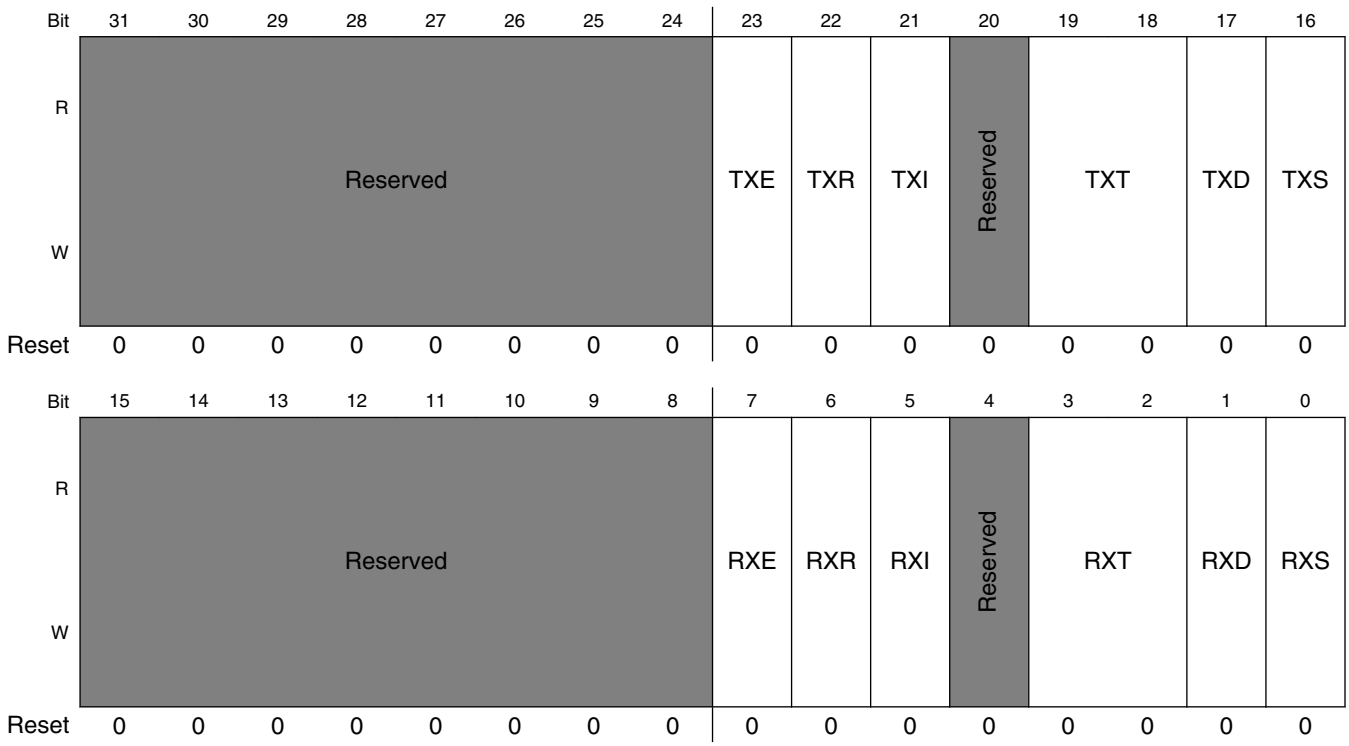
35.7.43 Endpoint Control 4 (USB_*n*ENDPTCTRL4)

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: 400E_4000h base + 1D0h offset + (65536d × i), where i=0d to 0d



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.

35.7.44 Endpoint Control 5 (USB_*n*ENDPTCTRL5)

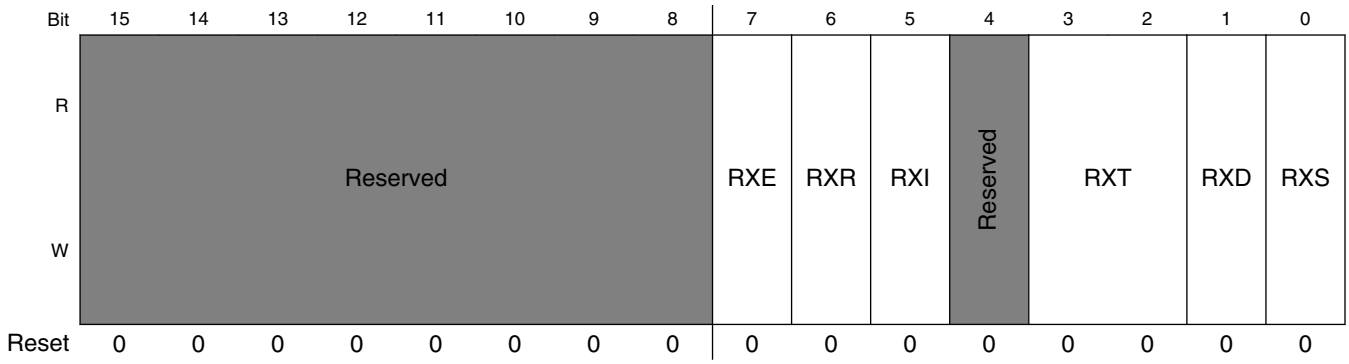
This is endpoint control register for endpoint 5 in device operation mode.

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: 400E_4000h base + 1D4h offset + (65536d × i), where i=0d to 0d

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.

35.7.45 Endpoint Control 6 (USB_*n*ENDPTCTRL6)

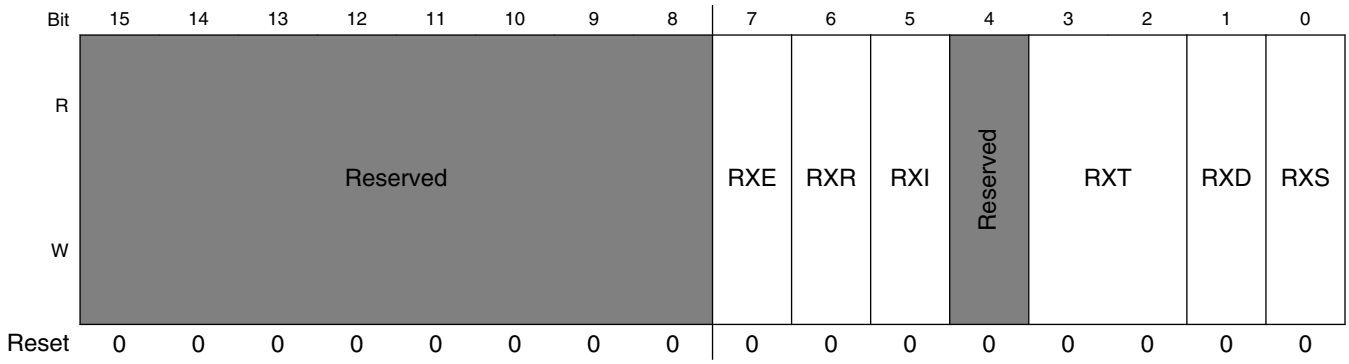
This is endpoint control register for endpoint 6 in device operation mode.

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: 400E_4000h base + 1D8h offset + (65536d × i), where i=0d to 0d

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.

35.7.46 Endpoint Control 7 (USB_*n*ENDPTCTRL7)

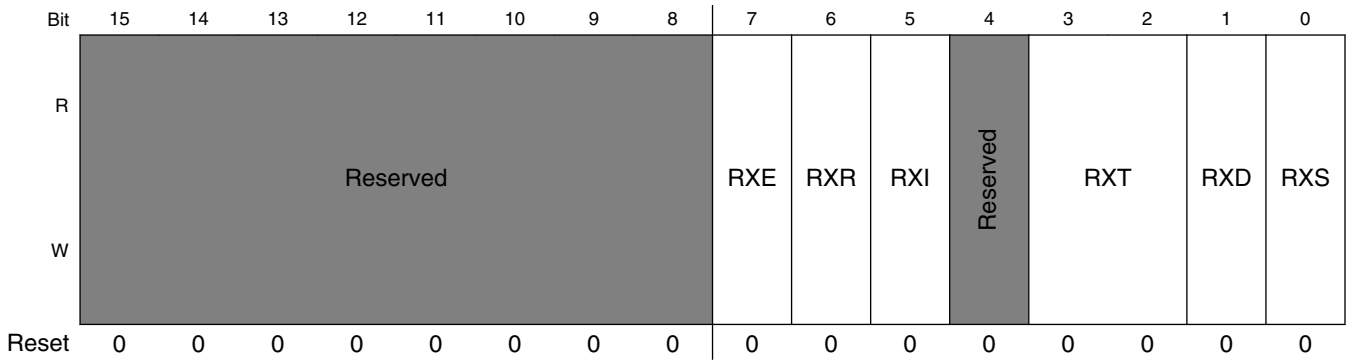
This is endpoint control register for endpoint 7 in device operation mode.

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: 400E_4000h base + 1DCh offset + (65536d × i), where i=0d to 0d

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 36
Universal Serial Bus 2.0 Integrated PHY (USB-PHY)

36.1 Chip-specific USB-PHY 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 USB PHY Overview

The chip contains one integrated USB 2.0 PHY macrocell 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.

36.3 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 36-1.

36.3.1 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.

36.3.2 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.

36.3.3 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).

36.3.4 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.

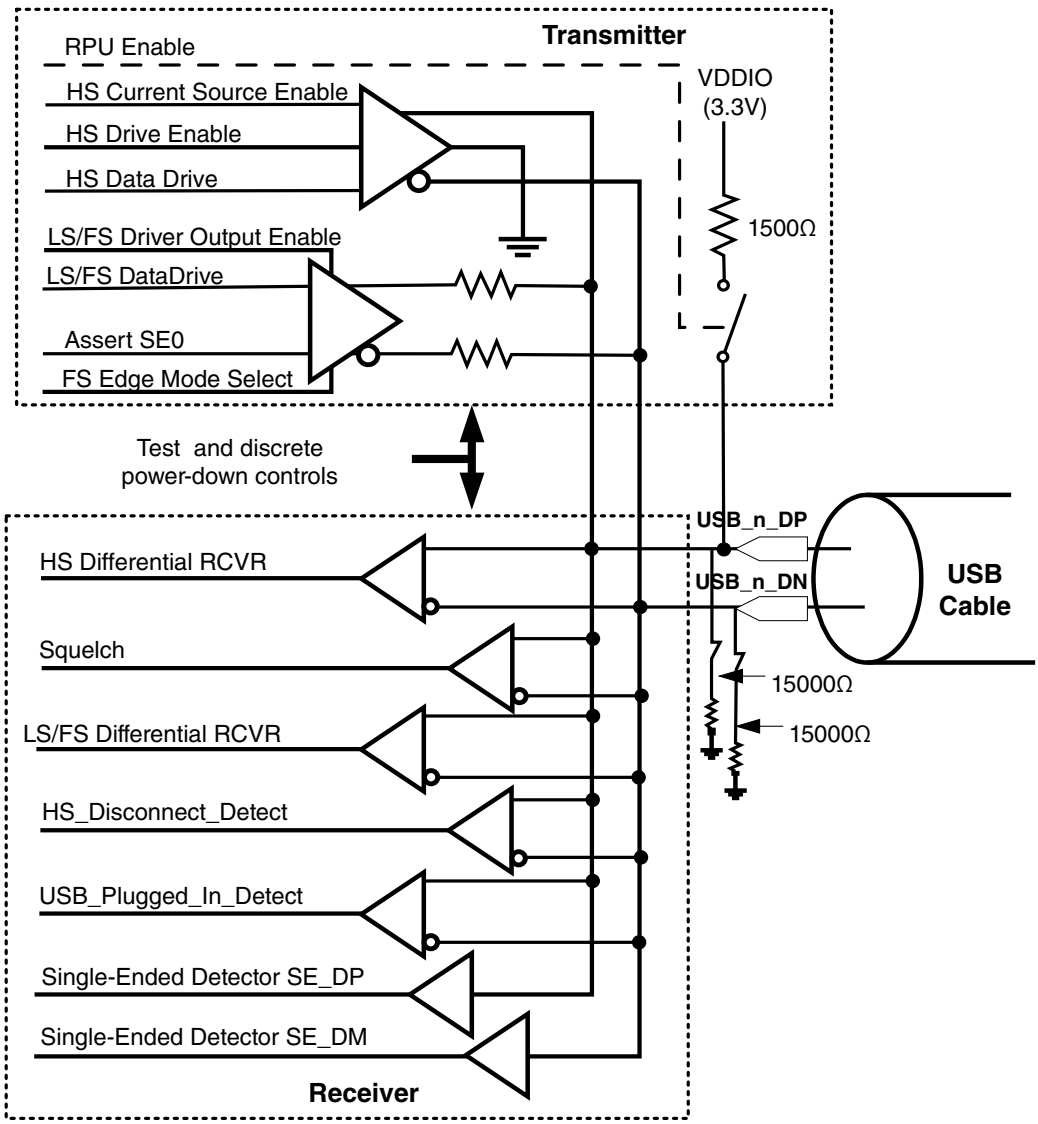


Figure 36-1. USB 2.0 PHY Analog Transceiver Block Diagram

36.3.4.1 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.

36.3.4.2 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.

36.3.4.3 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.

36.3.4.4 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.

36.3.4.5 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.

36.3.4.6 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.

36.3.4.7 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.

36.3.4.8 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.

36.3.5 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 36-1.

36.3.5.1 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

36-2. 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.

36.3.5.2 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.

36.3.5.3 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.

36.3.5.4 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.

36.3.5.5 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.

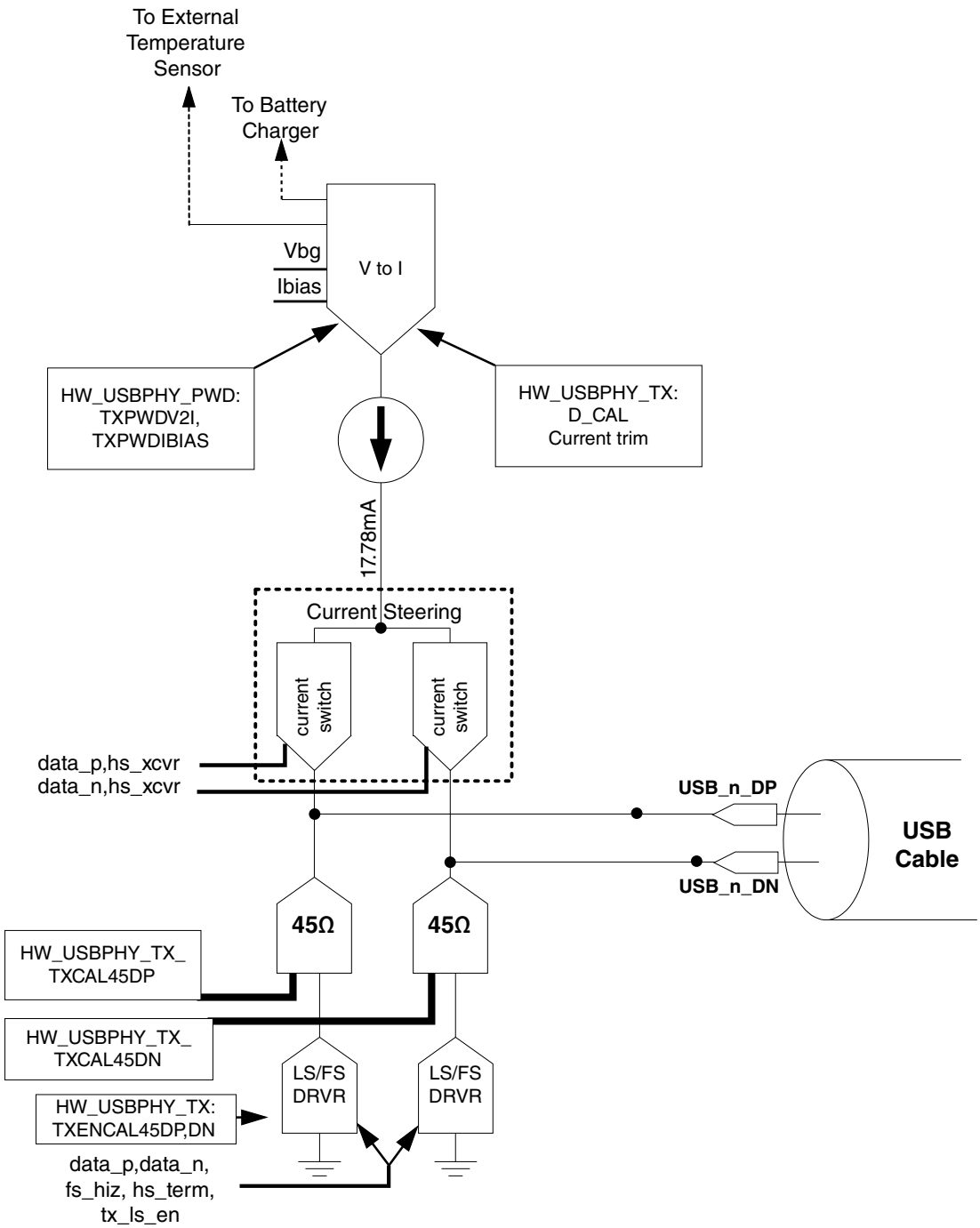


Figure 36-2. USB 2.0 PHY Transmitter Block Diagram

36.3.6 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 = 0x0
HW_USBPHY_TX_TXCAL45DN = 0x0
HW_USBPHY_TX_D_CAL = 0x7

36.3.7 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.

36.3.7.1 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 36-2. 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

36.3.7.2 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.

36.3.7.3 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.

36.3.7.4 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.

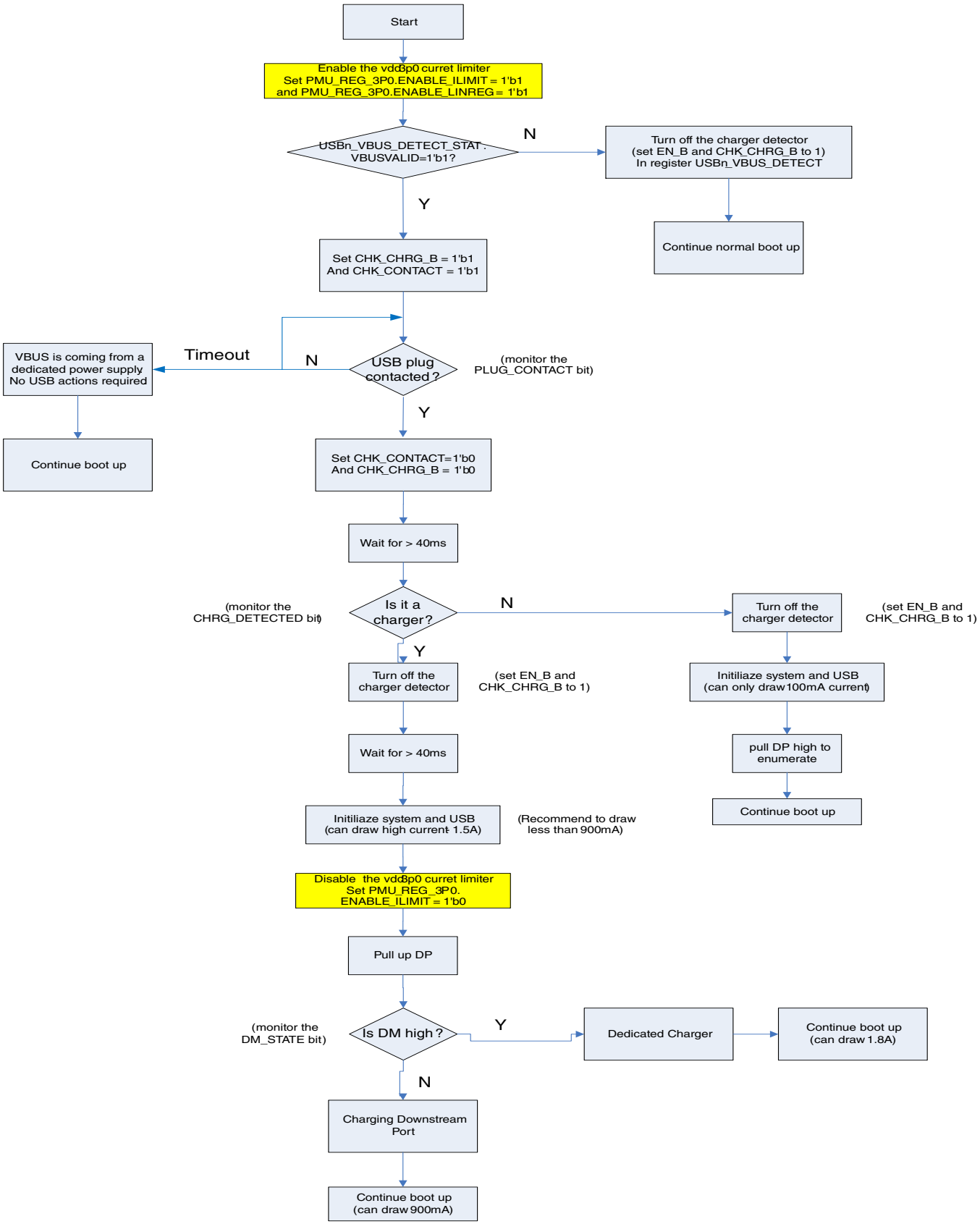


Figure 36-3. USBPHY Charger Detection Software Flow

36.3.7.5 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.

36.4 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 (USBPHY_PWD)	32	R/W	001E_1C00h	36.4.1/1390
400D_9004	USB PHY Power-Down Register (USBPHY_PWD_SET)	32	R/W	001E_1C00h	36.4.1/1390
400D_9008	USB PHY Power-Down Register (USBPHY_PWD_CLR)	32	R/W	001E_1C00h	36.4.1/1390
400D_900C	USB PHY Power-Down Register (USBPHY_PWD_TOG)	32	R/W	001E_1C00h	36.4.1/1390
400D_9010	USB PHY Transmitter Control Register (USBPHY_TX)	32	R/W	1006_0607h	36.4.2/1392

Table continues on the next page...

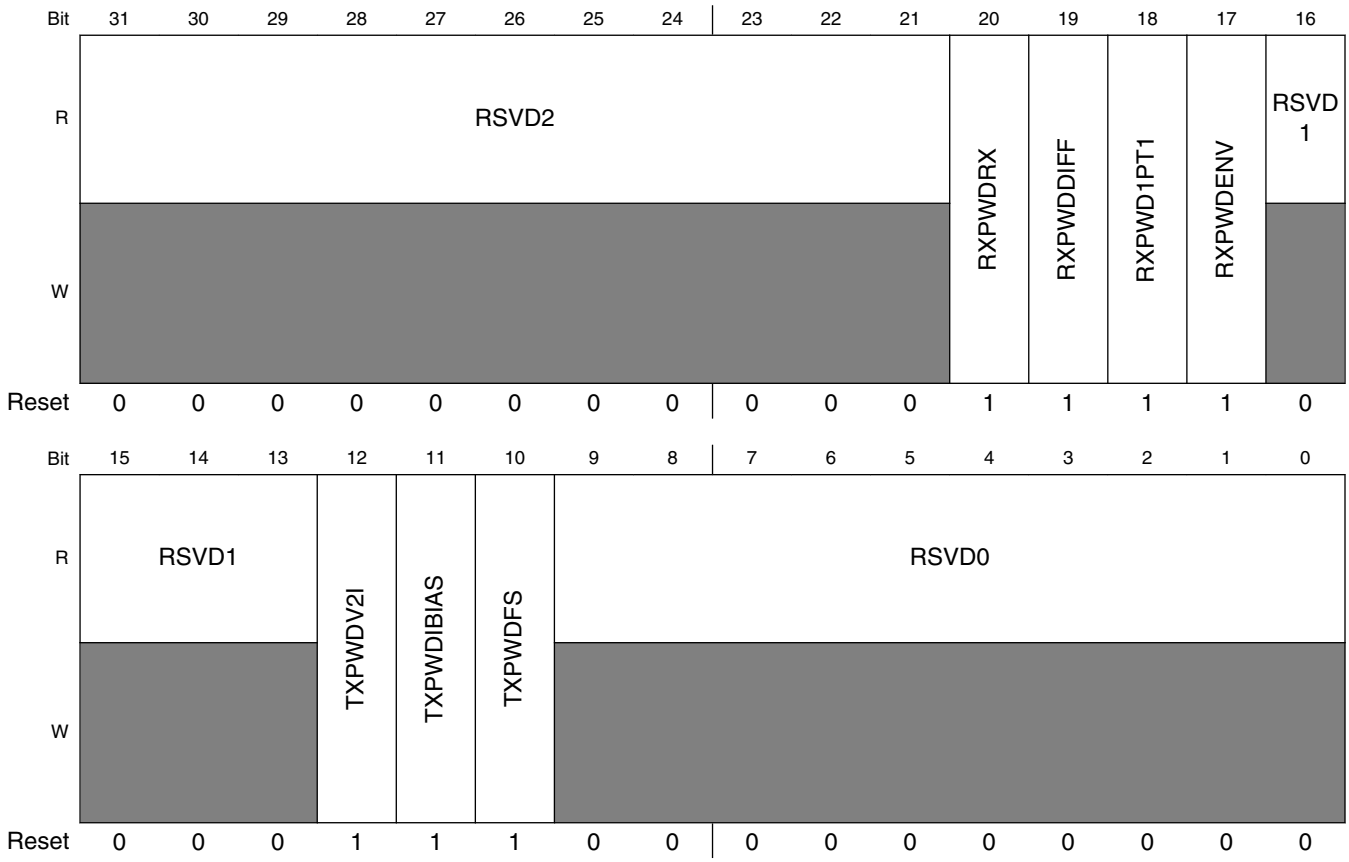
USBPHY memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
400D_9014	USB PHY Transmitter Control Register (USBPHY_TX_SET)	32	R/W	1006_0607h	36.4.2/1392
400D_9018	USB PHY Transmitter Control Register (USBPHY_TX_CLR)	32	R/W	1006_0607h	36.4.2/1392
400D_901C	USB PHY Transmitter Control Register (USBPHY_TX_TOG)	32	R/W	1006_0607h	36.4.2/1392
400D_9020	USB PHY Receiver Control Register (USBPHY_RX)	32	R/W	0000_0000h	36.4.3/1393
400D_9024	USB PHY Receiver Control Register (USBPHY_RX_SET)	32	R/W	0000_0000h	36.4.3/1393
400D_9028	USB PHY Receiver Control Register (USBPHY_RX_CLR)	32	R/W	0000_0000h	36.4.3/1393
400D_902C	USB PHY Receiver Control Register (USBPHY_RX_TOG)	32	R/W	0000_0000h	36.4.3/1393
400D_9030	USB PHY General Control Register (USBPHY_CTRL)	32	R/W	C020_0000h	36.4.4/1395
400D_9034	USB PHY General Control Register (USBPHY_CTRL_SET)	32	R/W	C020_0000h	36.4.4/1395
400D_9038	USB PHY General Control Register (USBPHY_CTRL_CLR)	32	R/W	C020_0000h	36.4.4/1395
400D_903C	USB PHY General Control Register (USBPHY_CTRL_TOG)	32	R/W	C020_0000h	36.4.4/1395
400D_9040	USB PHY Status Register (USBPHY_STATUS)	32	R/W	0000_0000h	36.4.5/1398
400D_9050	USB PHY Debug Register (USBPHY_DEBUG)	32	R/W	7F18_0000h	36.4.6/1400
400D_9054	USB PHY Debug Register (USBPHY_DEBUG_SET)	32	R/W	7F18_0000h	36.4.6/1400
400D_9058	USB PHY Debug Register (USBPHY_DEBUG_CLR)	32	R/W	7F18_0000h	36.4.6/1400
400D_905C	USB PHY Debug Register (USBPHY_DEBUG_TOG)	32	R/W	7F18_0000h	36.4.6/1400
400D_9060	UTMI Debug Status Register 0 (USBPHY_DEBUG0_STATUS)	32	R	0000_0000h	36.4.7/1402
400D_9070	UTMI Debug Status Register 1 (USBPHY_DEBUG1)	32	R/W	0000_1000h	36.4.8/1403
400D_9074	UTMI Debug Status Register 1 (USBPHY_DEBUG1_SET)	32	R/W	0000_1000h	36.4.8/1403
400D_9078	UTMI Debug Status Register 1 (USBPHY_DEBUG1_CLR)	32	R/W	0000_1000h	36.4.8/1403
400D_907C	UTMI Debug Status Register 1 (USBPHY_DEBUG1_TOG)	32	R/W	0000_1000h	36.4.8/1403
400D_9080	UTMI RTL Version (USBPHY_VERSION)	32	R	0403_0000h	36.4.9/1404

36.4.1 USB PHY Power-Down Register (USBPHY_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: 400D_9000h base + 0h offset + (4d × i), where i=0d to 3d



USBPHY_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...

USBPHY_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.

36.4.2 USB PHY Transmitter Control Register (USBPHY_TX*n*)

The USB PHY Transmitter Control Register handles the transmit controls.

Address: 400D_9000h 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	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

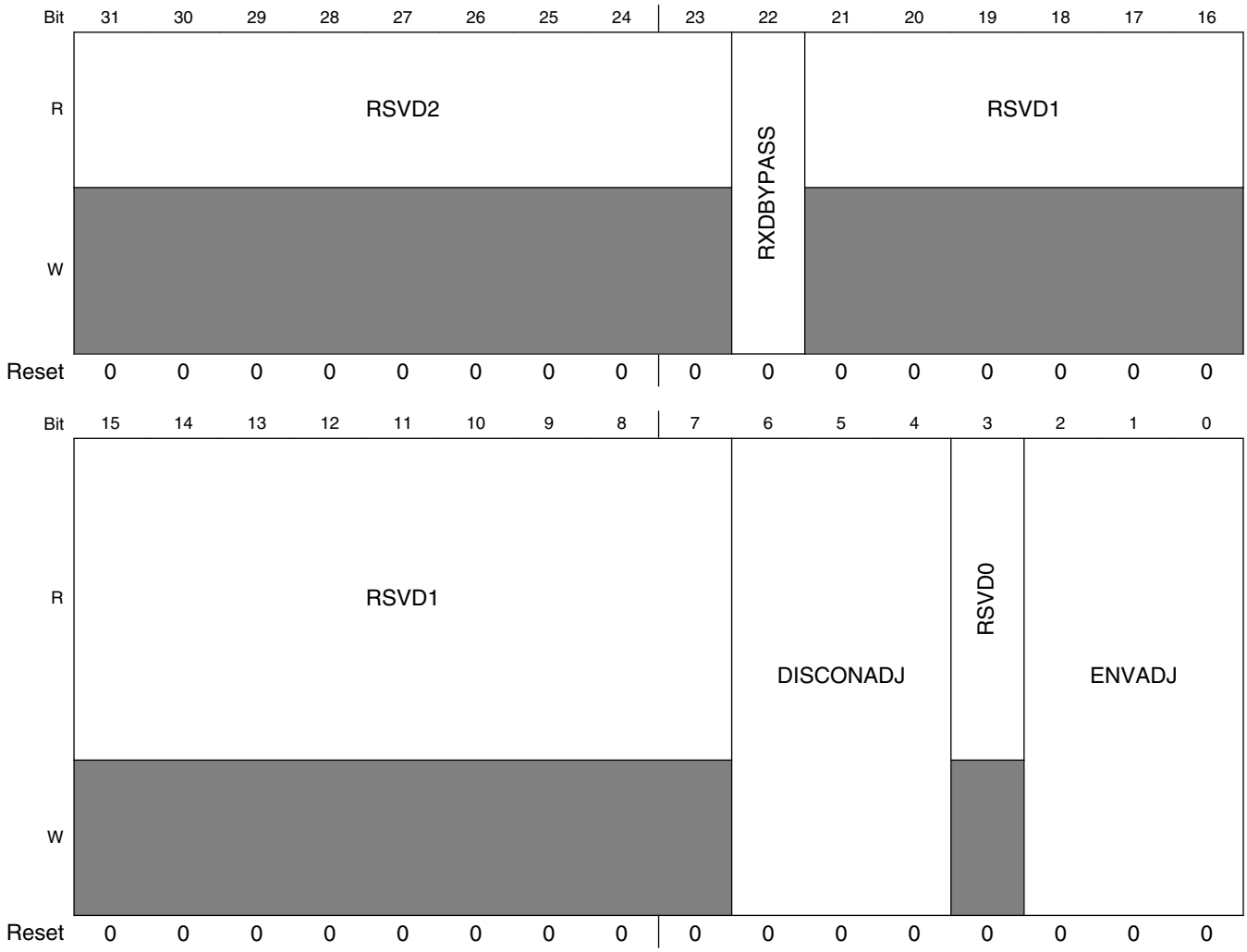
USBPHY_TX*n* field descriptions

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%

36.4.3 USB PHY Receiver Control Register (USBPHY_RX*n*)

The USB PHY Receiver Control Register handles receive path controls.

Address: 400D_9000h base + 20h offset + (4d × i), where i=0d to 3d



USBPHY_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...

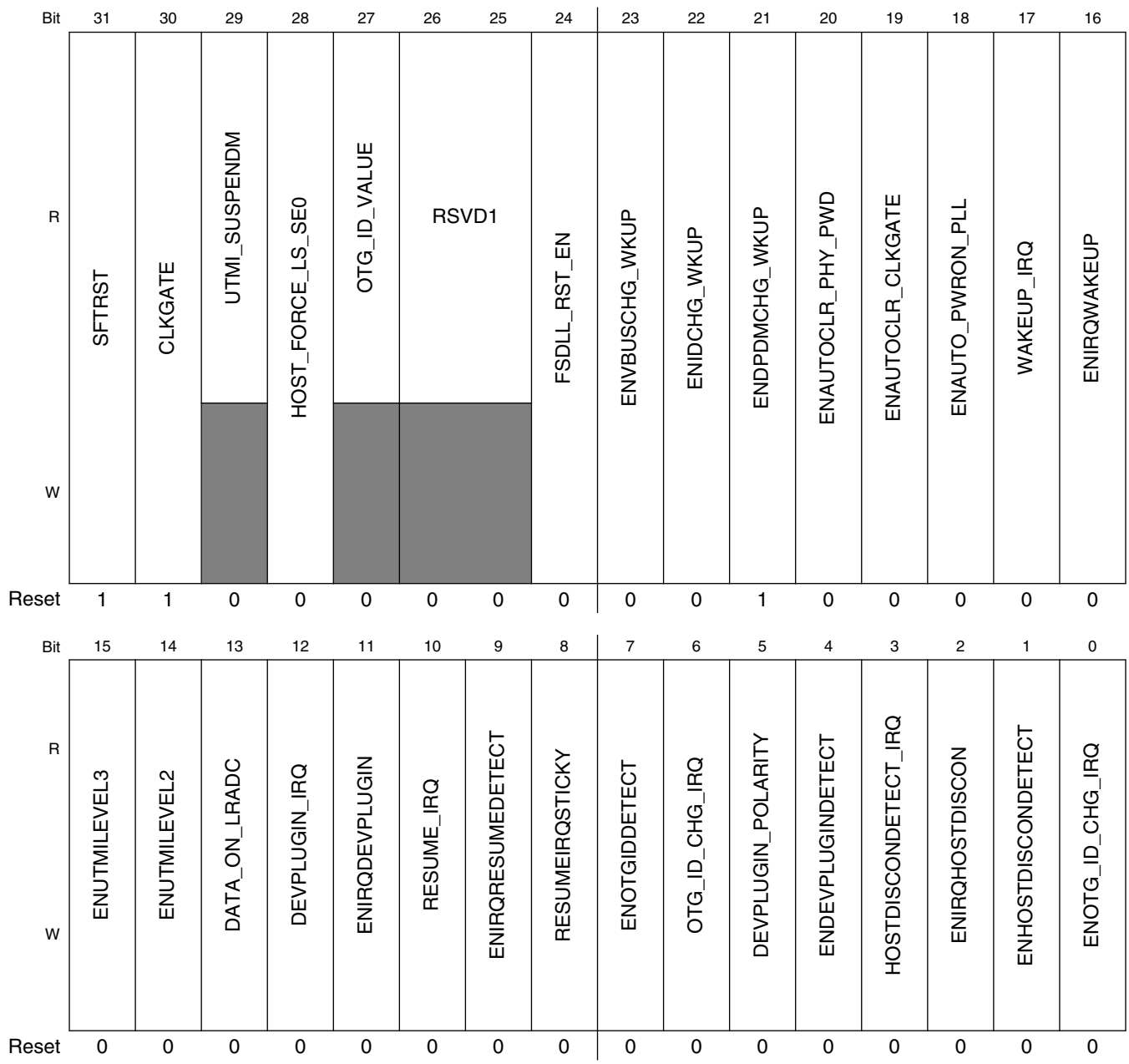
USBPHY_RX*n* field descriptions (continued)

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

36.4.4 USB PHY General Control Register (USBPHY_CTRL*n*)

The USB PHY General Control Register handles OTG and Host controls. This register also includes interrupt enables and connectivity detect enables and results.

Address: 400D_9000h base + 30h offset + (4d × i), where i=0d to 3d



USBPHY_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...

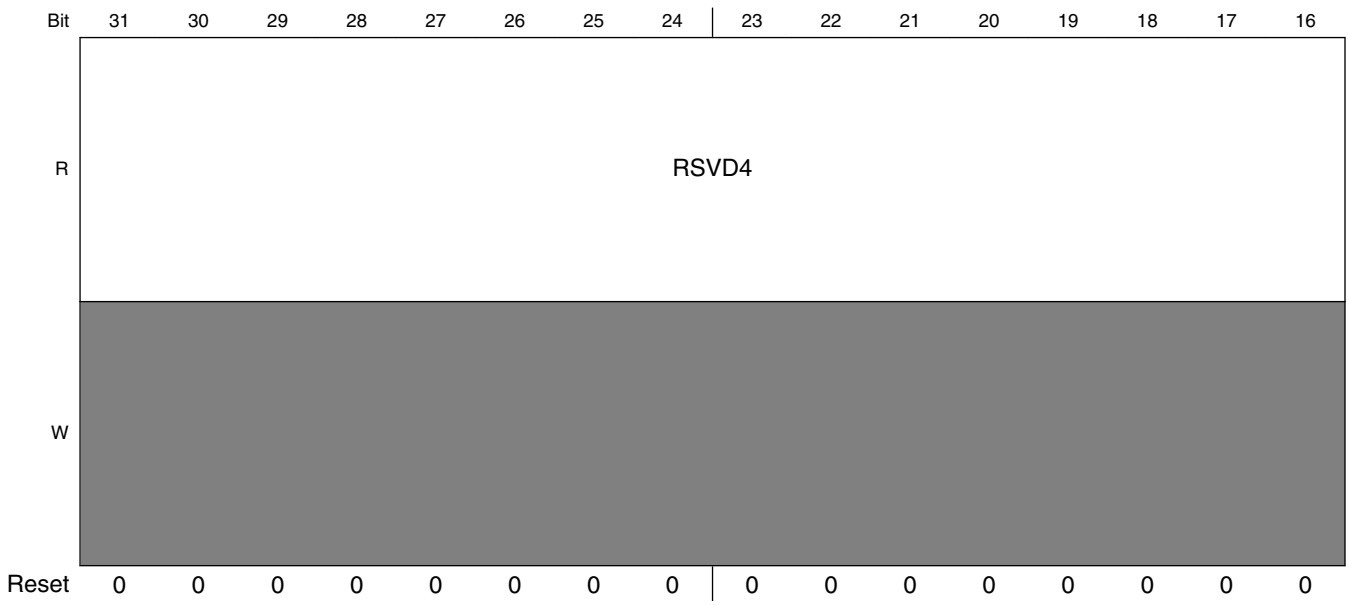
USBPHY_CTRL*n* field descriptions (continued)

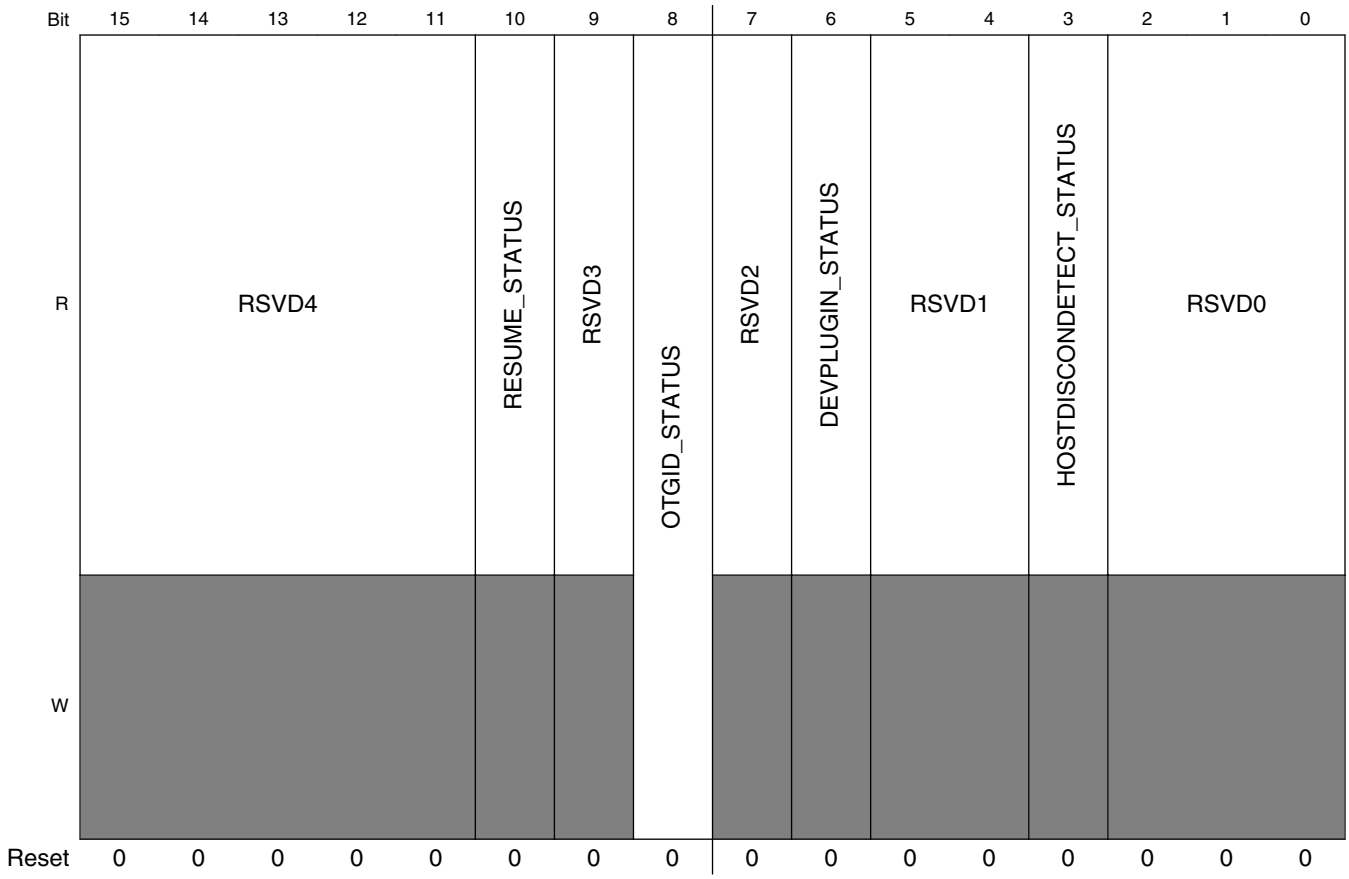
Field	Description
11 ENIRQDEVPLUGIN	Enables interrupt for the detection of connectivity to the USB line.
10 RESUME_IRQ	Indicates that the host is sending a wake-up after suspend. This bit is also set on a reset during suspend. Use this bit to wake up from suspend for either the resume or the reset case. 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.

36.4.5 USB PHY Status Register (USBPHY_STATUS)

The USB PHY Status Register holds results of IRQ and other detects.

Address: 400D_9000h base + 40h offset = 400D_9040h





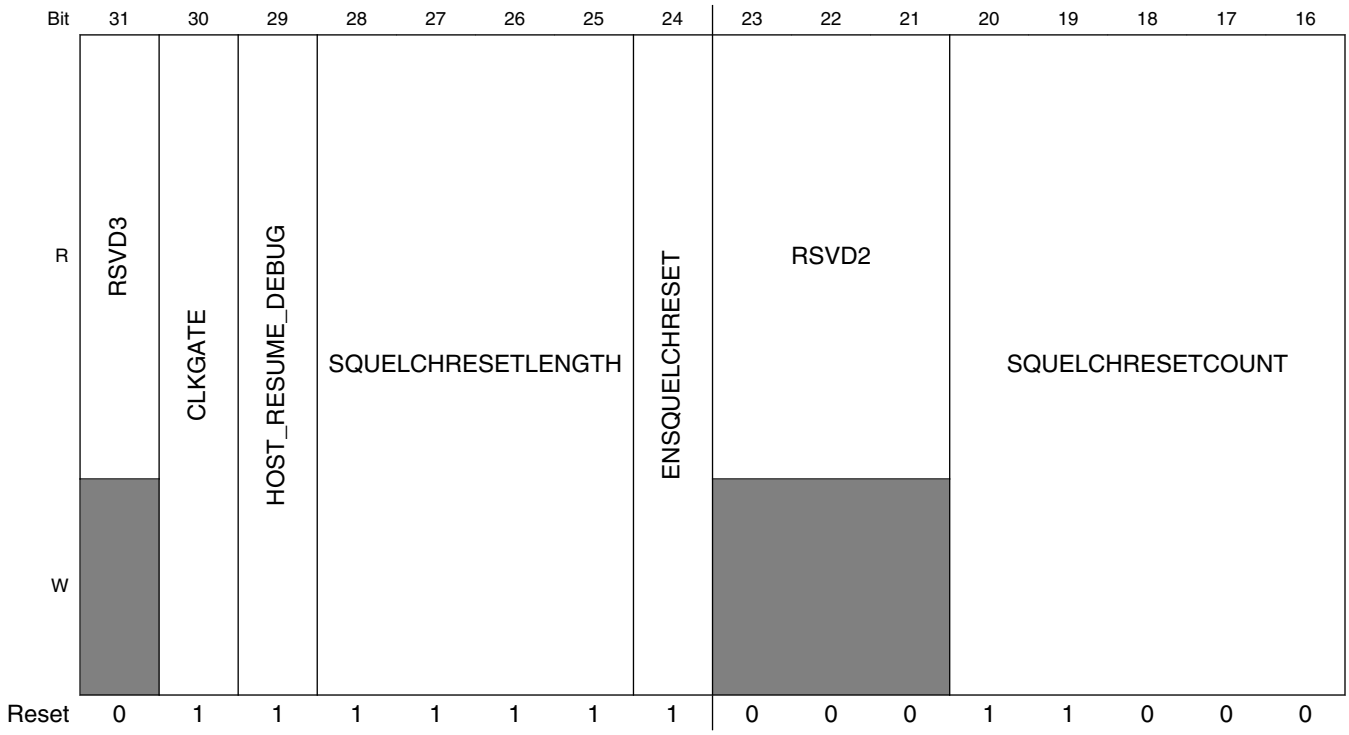
USBPHY_STATUS field descriptions

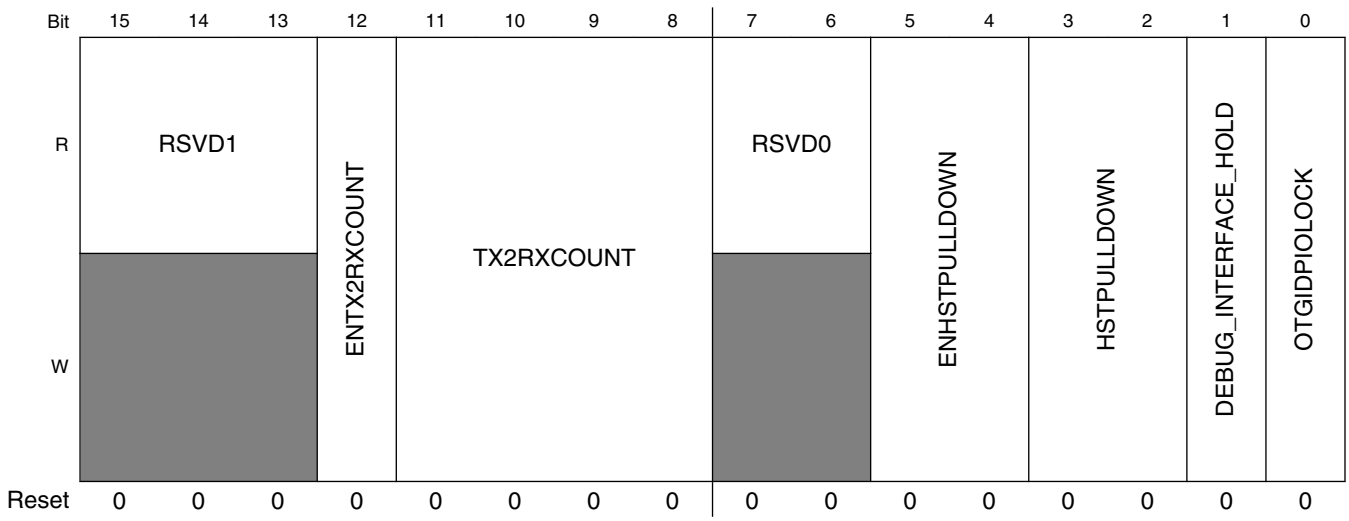
Field	Description
31–11 RSVD4	Reserved.
10 RESUME_STATUS	Indicates that the host is sending a wake-up after suspend and has triggered an interrupt.
9 RSVD3	Reserved.
8 OTGID_STATUS	Indicates the results of ID pin on MiniAB plug. False (0) is when ID resistance is less than Ra_Plug_ID, indicating host (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.

36.4.6 USB PHY Debug Register (USBPHY_DEBUG*n*)

This register is used to debug the USB PHY.

Address: 400D_9000h base + 50h offset + (4d × i), where i=0d to 3d





USBPHY_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.

36.4.7 UTMI Debug Status Register 0 (USBPHY_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: 400D_9000h base + 60h offset = 400D_9060h

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

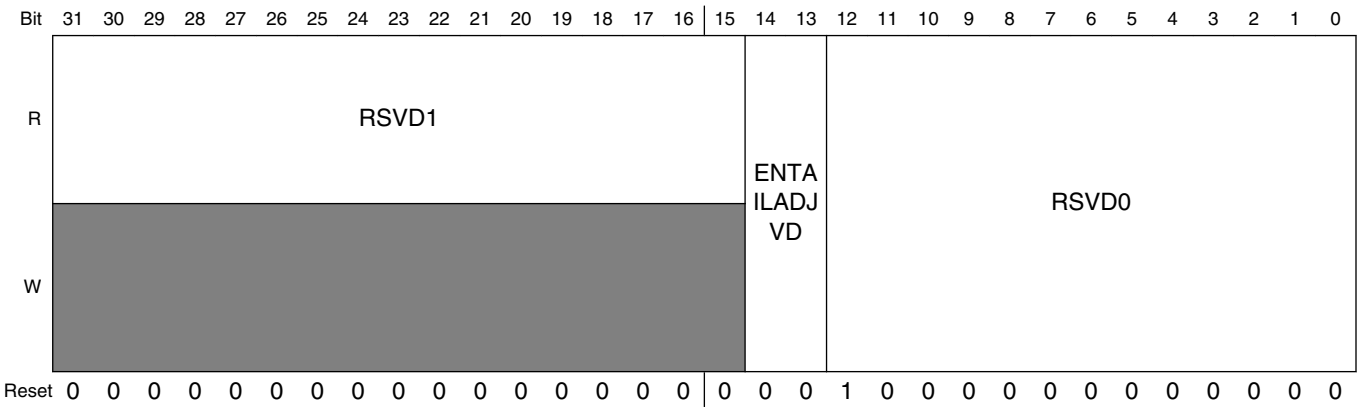
USBPHY_DEBUG0_STATUS field descriptions

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.

36.4.8 UTMI Debug Status Register 1 (USBPHY_DEBUG1*n*)

Chooses the muxing of the debug register to be shown in USBPHYx_DEBUG0_STATUS.

Address: 400D_9000h base + 70h offset + (4d × i), where i=0d to 3d



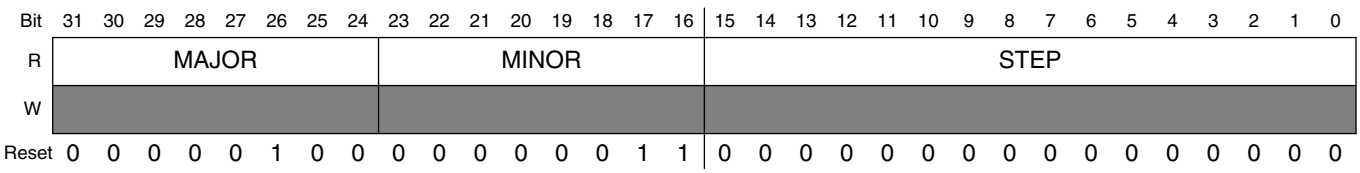
USBPHY_DEBUG1*n* field descriptions

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.

36.4.9 UTMI RTL Version (USBPHY_VERSION)

Fields for RTL Version.

Address: 400D_9000h base + 80h offset = 400D_9080h



USBPHY_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.

36.5 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	36.5.1/1405
400D_81A4	USB VBUS Detect Register (USB_ANALOG_USB1_VBUS_DETECT_SET)	32	R/W	0010_0004h	36.5.1/1405
400D_81A8	USB VBUS Detect Register (USB_ANALOG_USB1_VBUS_DETECT_CLR)	32	R/W	0010_0004h	36.5.1/1405
400D_81AC	USB VBUS Detect Register (USB_ANALOG_USB1_VBUS_DETECT_TOG)	32	R/W	0010_0004h	36.5.1/1405
400D_81B0	USB Charger Detect Register (USB_ANALOG_USB1_CHRG_DETECT)	32	R/W	0000_0000h	36.5.2/1407
400D_81B4	USB Charger Detect Register (USB_ANALOG_USB1_CHRG_DETECT_SET)	32	R/W	0000_0000h	36.5.2/1407
400D_81B8	USB Charger Detect Register (USB_ANALOG_USB1_CHRG_DETECT_CLR)	32	R/W	0000_0000h	36.5.2/1407
400D_81BC	USB Charger Detect Register (USB_ANALOG_USB1_CHRG_DETECT_TOG)	32	R/W	0000_0000h	36.5.2/1407

Table continues on the next page...

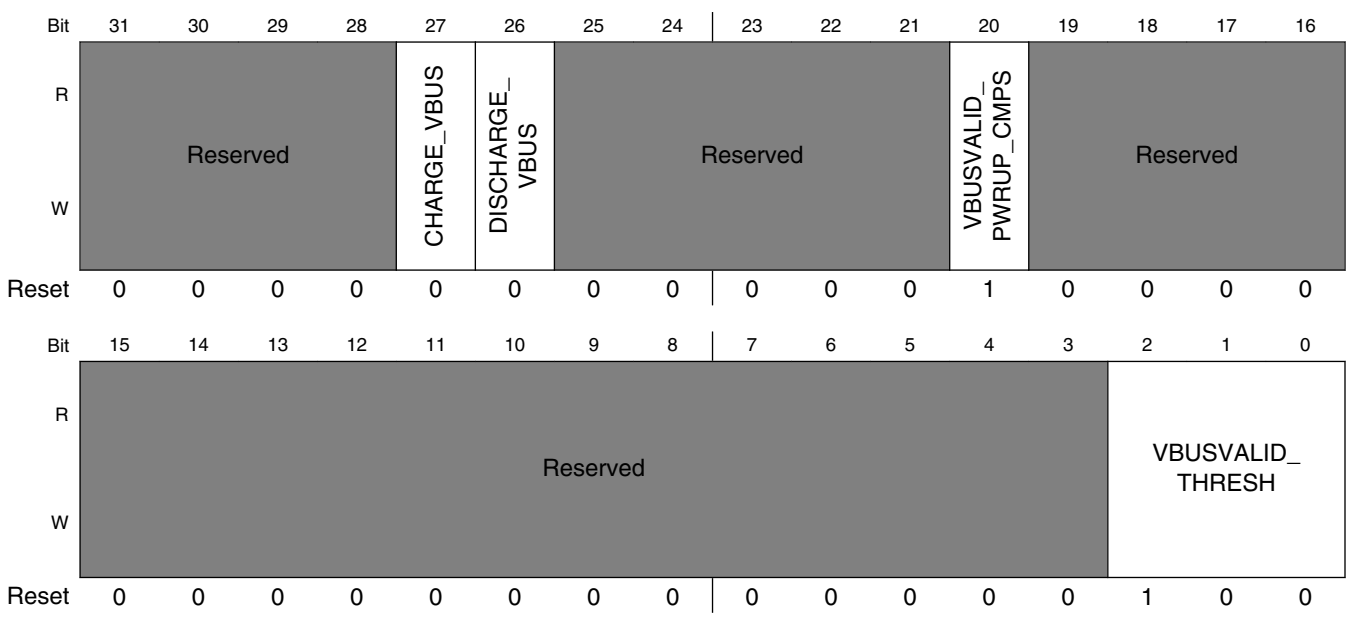
USB_ANALOG memory map (continued)

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	36.5.3/1409
400D_81D0	USB Charger Detect Status Register (USB_ANALOG_USB1_CHRG_DETECT_STAT)	32	R	0000_0000h	36.5.4/1411
400D_81E0	USB Loopback Test Register (USB_ANALOG_USB1_LOOPBACK)	32	R/W	0000_0000h	36.5.5/1412
400D_81E4	USB Loopback Test Register (USB_ANALOG_USB1_LOOPBACK_SET)	32	R/W	0000_0000h	36.5.5/1412
400D_81E8	USB Loopback Test Register (USB_ANALOG_USB1_LOOPBACK_CLR)	32	R/W	0000_0000h	36.5.5/1412
400D_81EC	USB Loopback Test Register (USB_ANALOG_USB1_LOOPBACK_TOG)	32	R/W	0000_0000h	36.5.5/1412
400D_81F0	USB Misc Register (USB_ANALOG_USB1_MISC)	32	R/W	0000_0002h	36.5.6/1413
400D_81F4	USB Misc Register (USB_ANALOG_USB1_MISC_SET)	32	R/W	0000_0002h	36.5.6/1413
400D_81F8	USB Misc Register (USB_ANALOG_USB1_MISC_CLR)	32	R/W	0000_0002h	36.5.6/1413
400D_81FC	USB Misc Register (USB_ANALOG_USB1_MISC_TOG)	32	R/W	0000_0002h	36.5.6/1413
400D_8260	Chip Silicon Version (USB_ANALOG_DIGPROG)	32	R	006D_0000h	36.5.7/1414

36.5.1 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



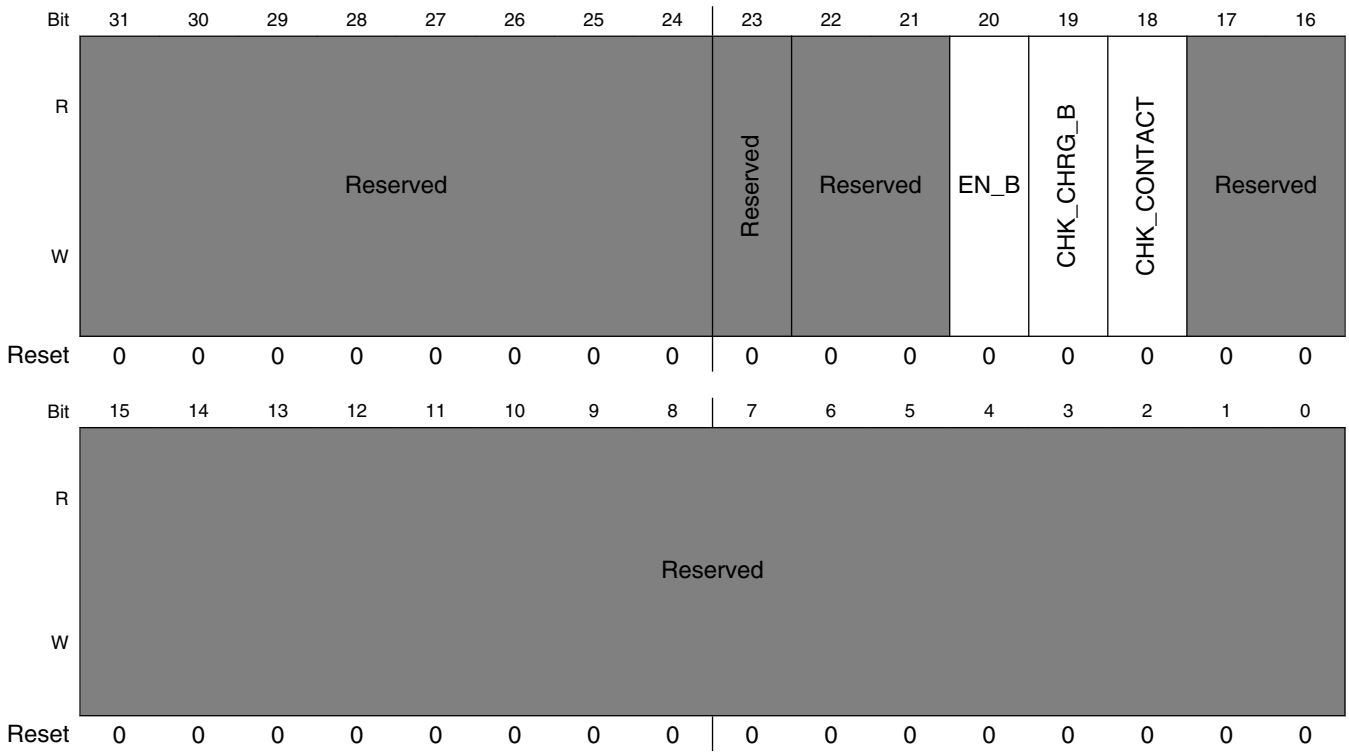
USB_ANALOG_USB1_VBUS_DETECT*n* field descriptions

Field	Description
31–28 -	This field is reserved. Reserved.
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

36.5.2 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...

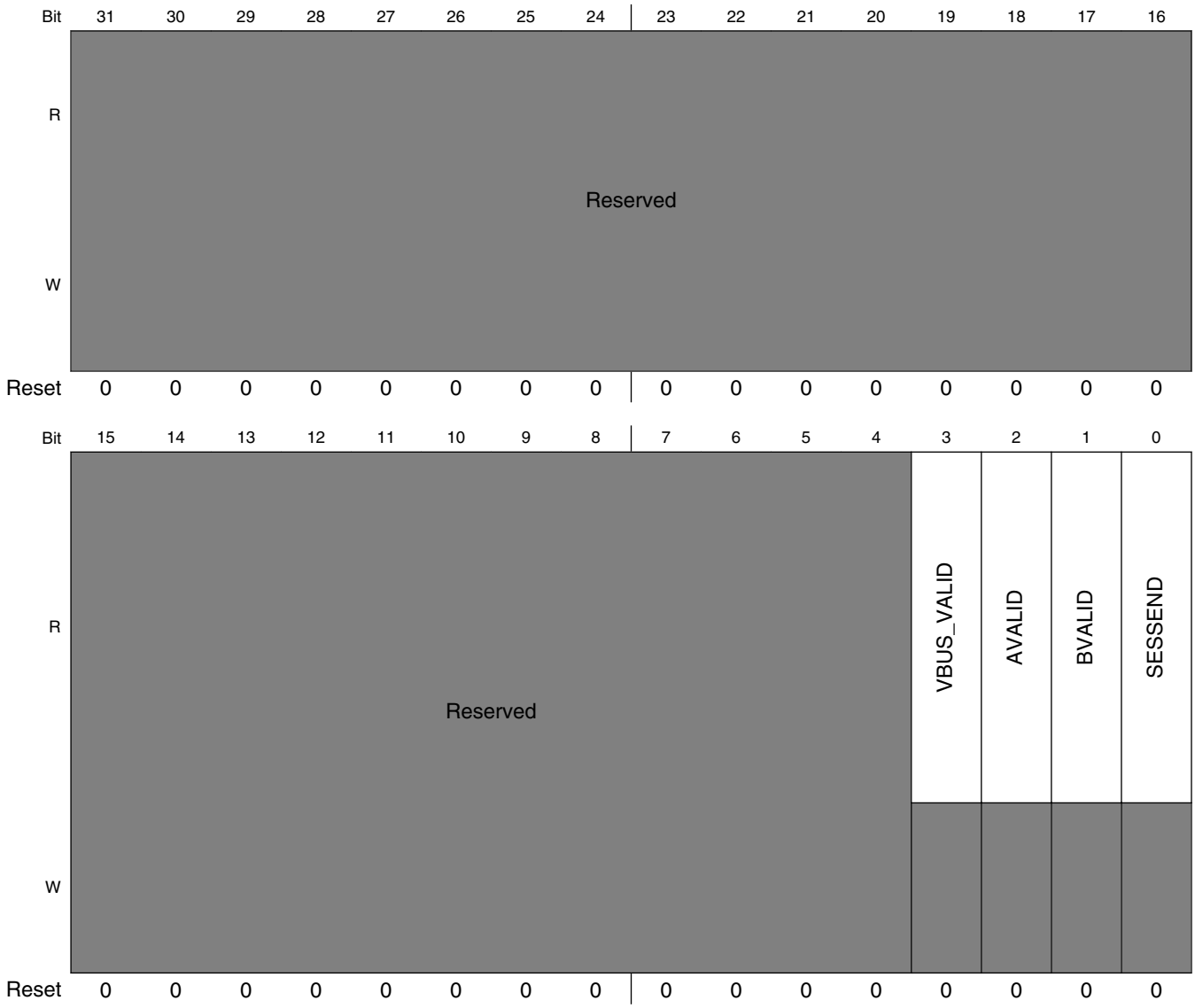
USB_ANALOG_USB1_CHRG_DETECT*n* field descriptions (continued)

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.

36.5.3 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...

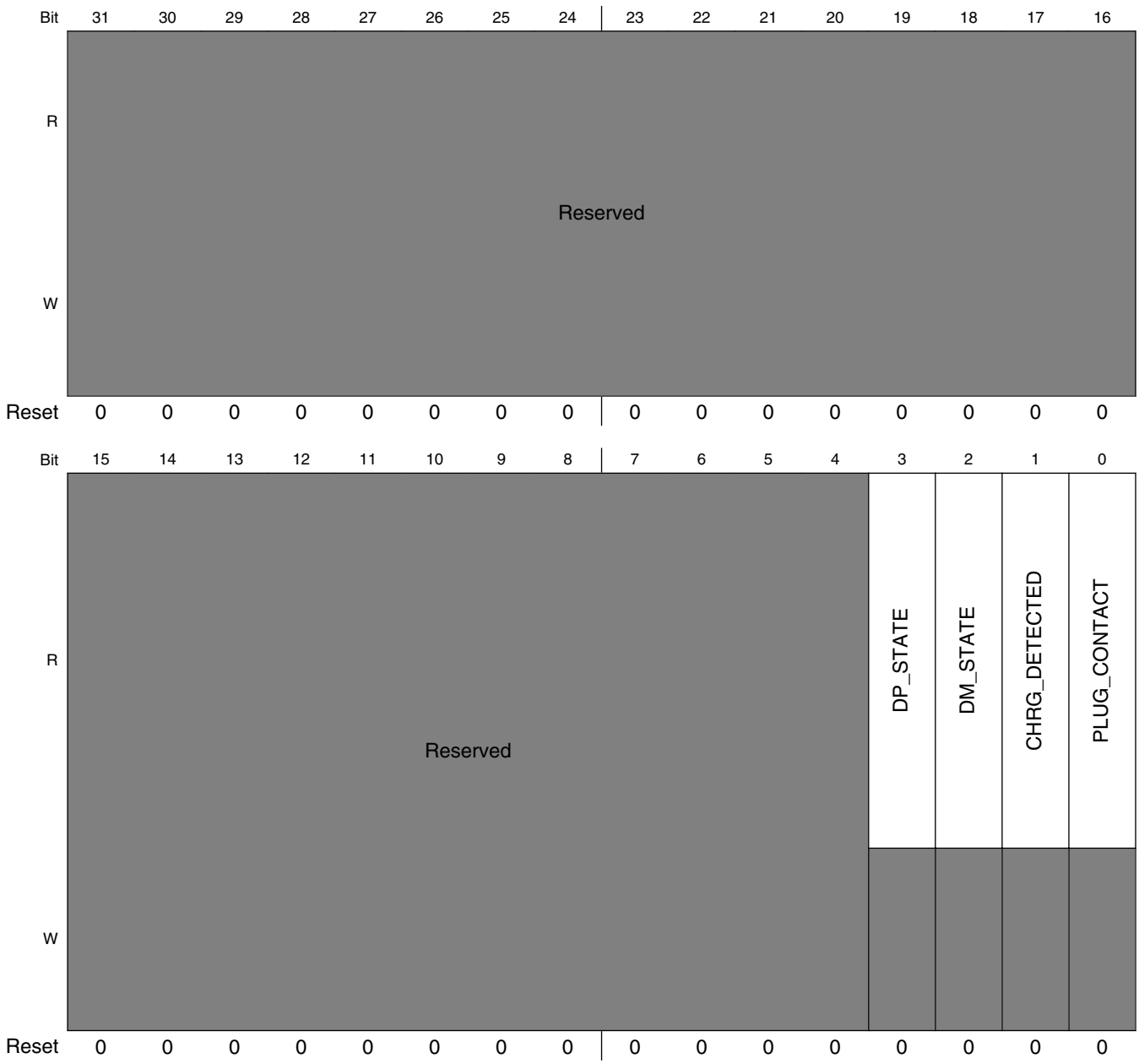
USB_ANALOG_USB1_VBUS_DETECT_STAT field descriptions (continued)

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.

36.5.4 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



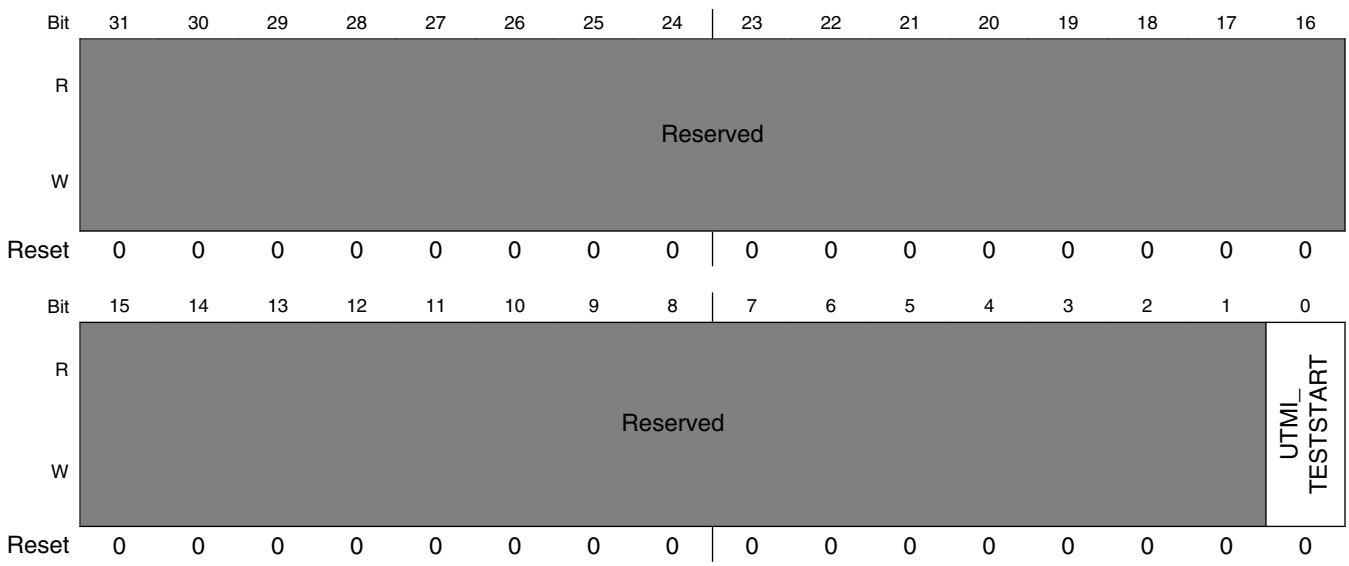
USB_ANALOG_USB1_CHRG_DETECT_STAT field descriptions

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.

36.5.5 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.

36.5.6 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

Bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved	EN_CLK_UTMI	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														EN_DEGLITCH	HS_USE_EXTERNAL_R
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

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.

36.5.7 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	1	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 0x006D0000 Silicon revision 1.0

Chapter 37
Keypad Port (KPP)

37.1 Chip-specific KPP information

Table 37-1. 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

This device only supports 4 x 4 external key pad matrix.

37.2 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.

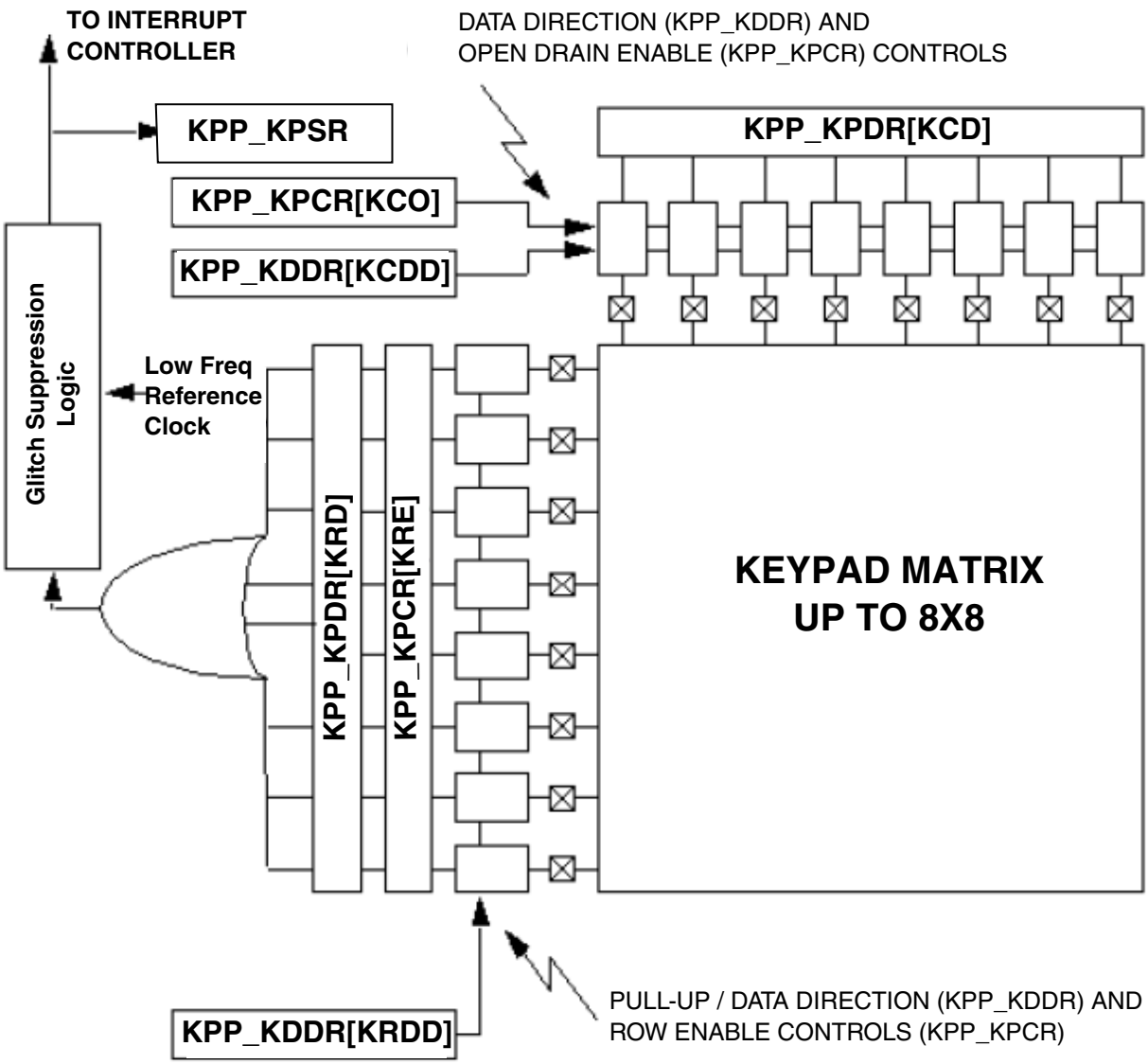


Figure 37-1. KPP Peripheral Block Diagram

37.2.1 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

37.2.2 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).

37.3 Clocks

The table found here describes the clock sources for KPP.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 37-2. KPP Clocks

Clock name	Clock Root	Description
ipg_clk_32k	ckil_sync_clk_root	Low-frequency reference clock (32 kHz)
ipg_clk_s	ipg_clk_root	Peripheral access clock

37.4 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.

See the table below for the list of external signals.

Table 37-3. KPP External Signals

Signal	Description	Pad	Mode	Direction
KPP_ROW0	Row input or output pin, from chip	GPIO_AD_B1_14	ALT7	IO
KPP_COL0	Column input or output pin, from chip	GPIO_AD_B1_15	ALT7	IO
KPP_ROW1	Row input or output pin, from chip	GPIO_AD_B1_12	ALT7	IO
KPP_COL1	Column input or output pin, from chip	GPIO_AD_B1_13	ALT7	IO
KPP_ROW2	Row input or output pin, from chip	GPIO_AD_B1_10	ALT7	IO
KPP_COL2	Column input or output pin, from chip	GPIO_AD_B1_11	ALT7	IO
KPP_ROW3	Row input or output pin, from chip	GPIO_AD_B1_08	ALT7	IO
KPP_COL3	Column input or output pin, from chip	GPIO_AD_B1_09	ALT7	IO

37.4.1 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.

37.4.2 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 37-4. 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 capability should be provided for column pins. 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.

37.4.3 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.

37.5 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.

37.5.1 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.

37.5.2 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.

37.5.3 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 (3 or 4) 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 higher software layer.

The basic debouncing period, which must be defined in the software routine, may be controlled with an internal timer. 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.

37.5.4 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 will 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.

37.5.5 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.

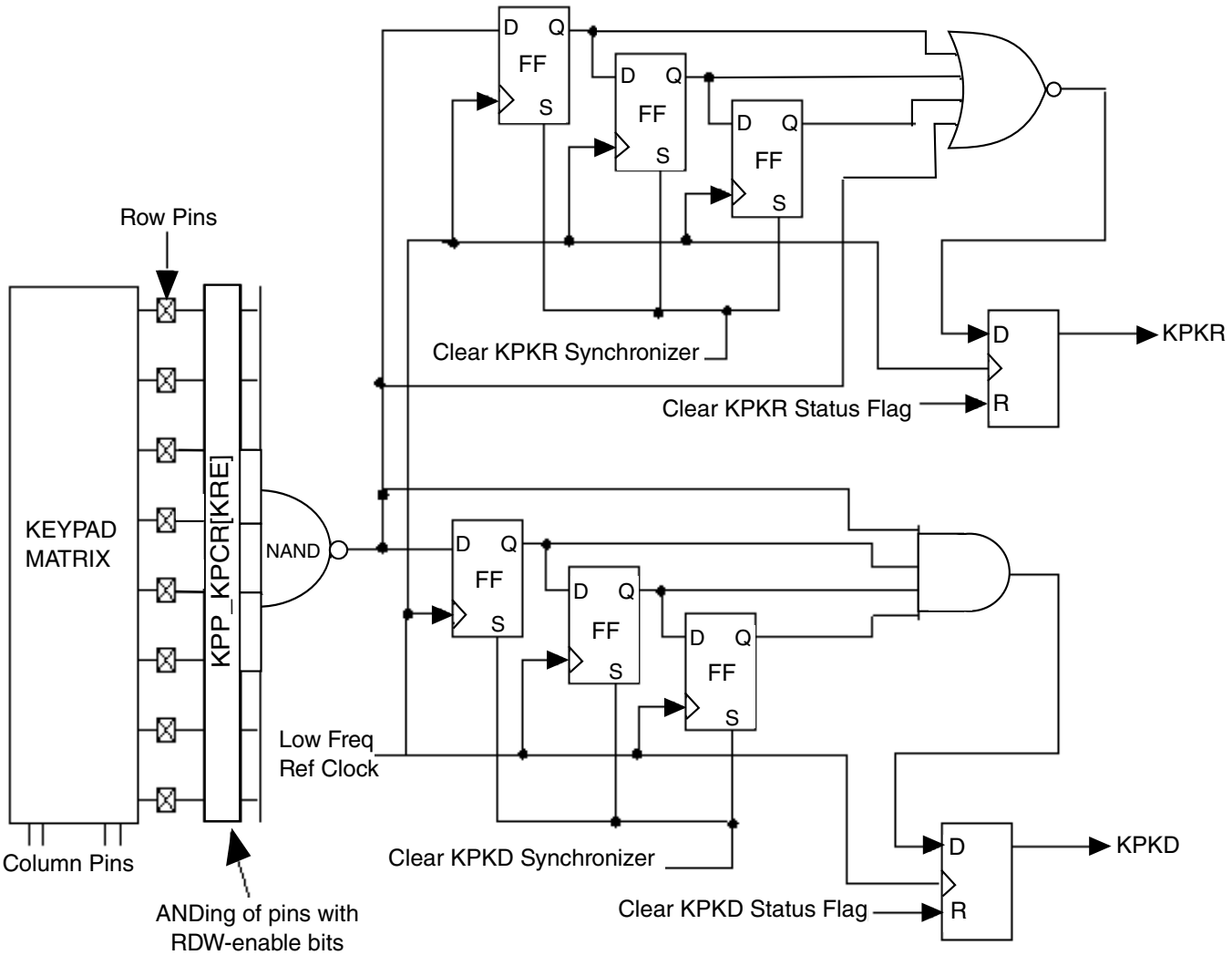


Figure 37-2. Keypad Synchronizer Functional Diagram

37.5.6 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 row 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.

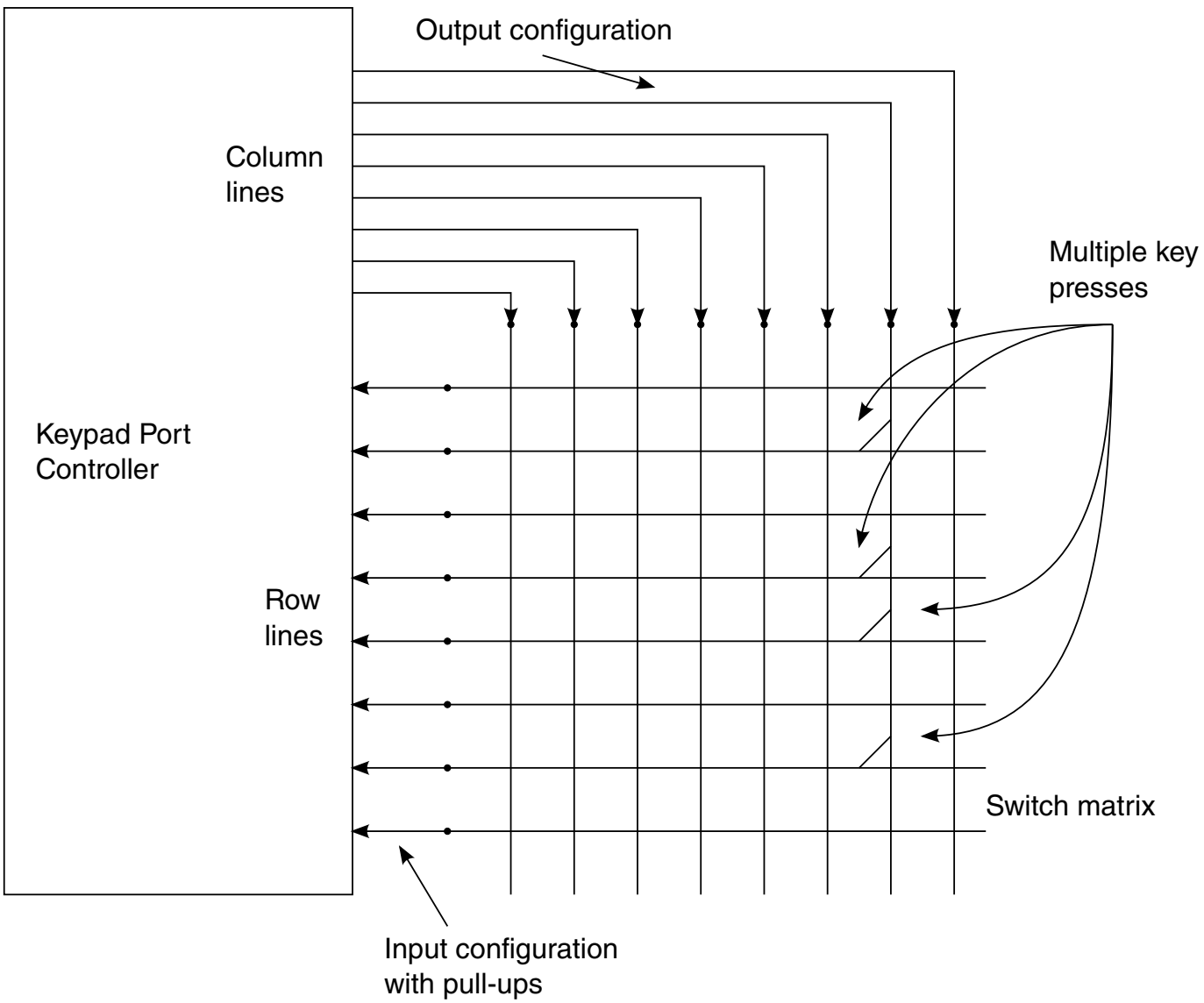


Figure 37-3. Multiple Key Presses on Same Column Line (Simplified View)

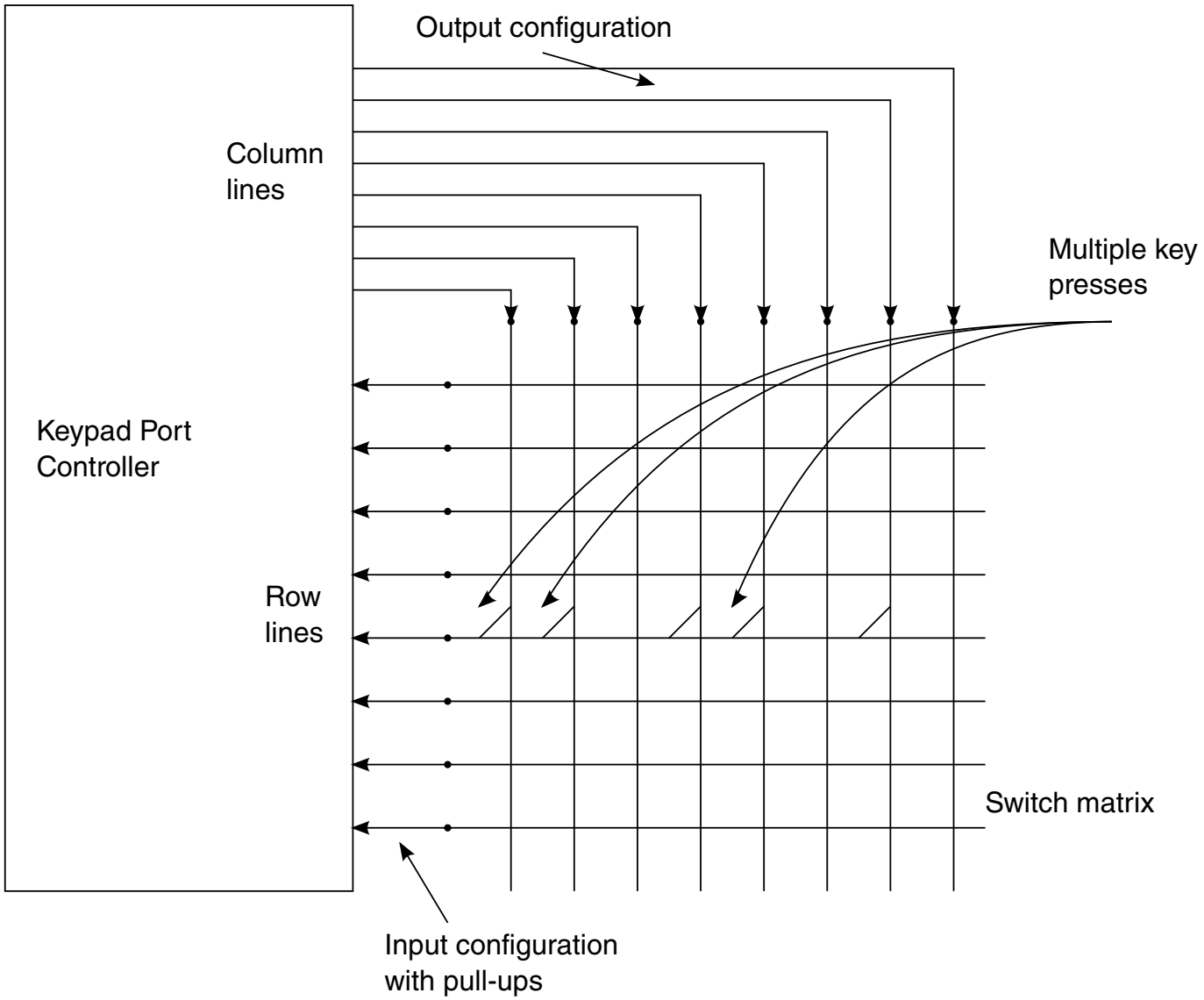


Figure 37-4. Multiple Key Presses on Same Row Line (Simplified View)

NOTE

An n key rollover is a technique with which the system can recognize the order in which keys are pressed.

37.5.6.1 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 37-5, 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 37-6).

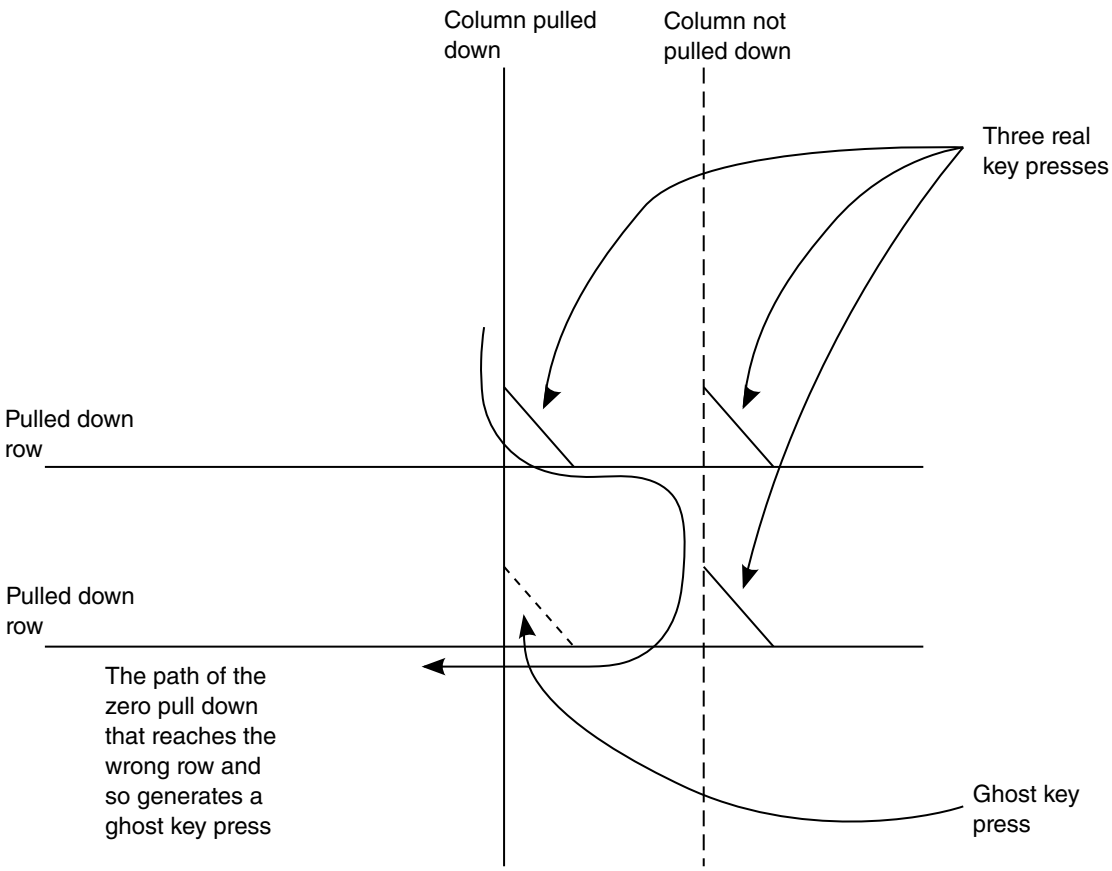


Figure 37-5. Decoding Wrong Three- Key-Presses

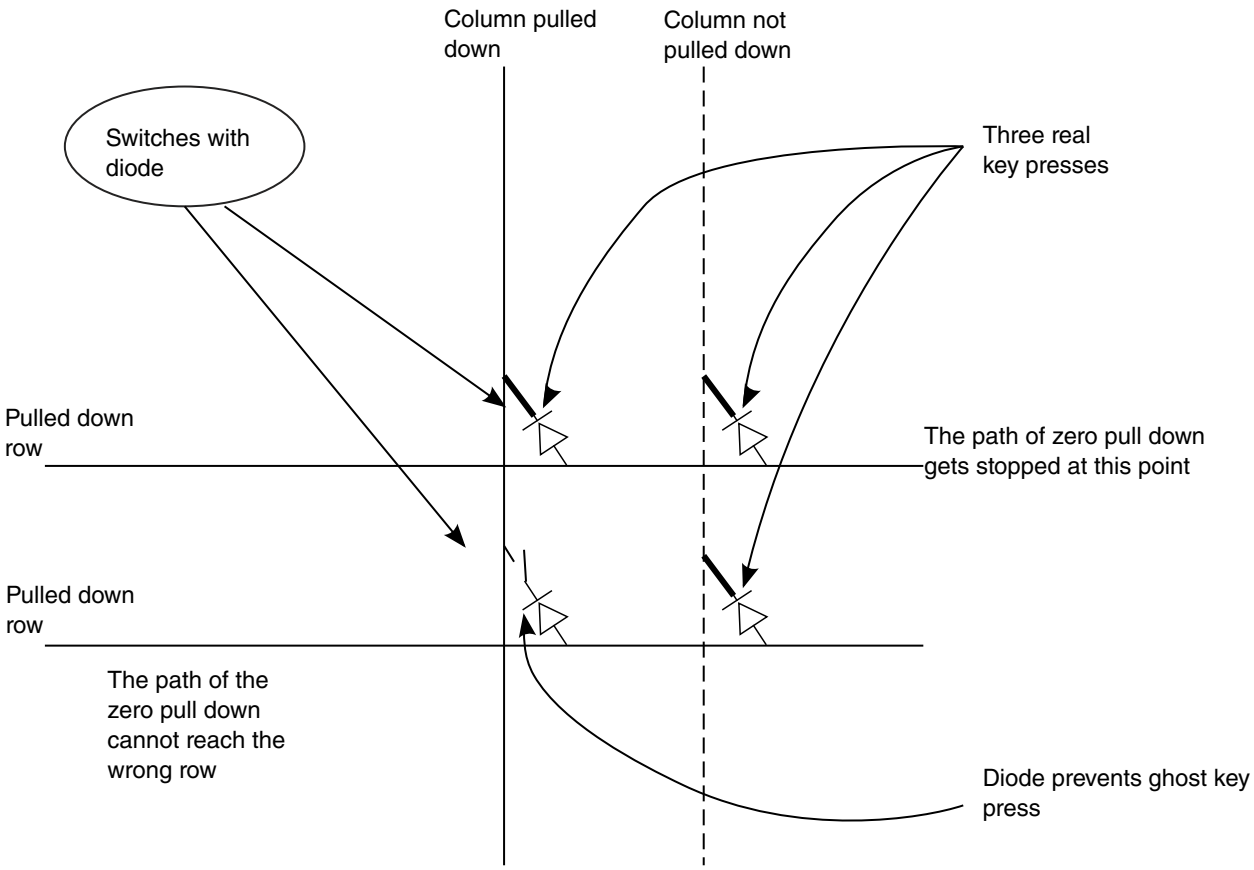


Figure 37-6. Matrix with "Ghost" Key Protections

37.5.7 3-Point Contact Keys Support

The KPP supports interfacing to a matrix consisting of 3-point contact keys. As shown in Figure 37-7, 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.

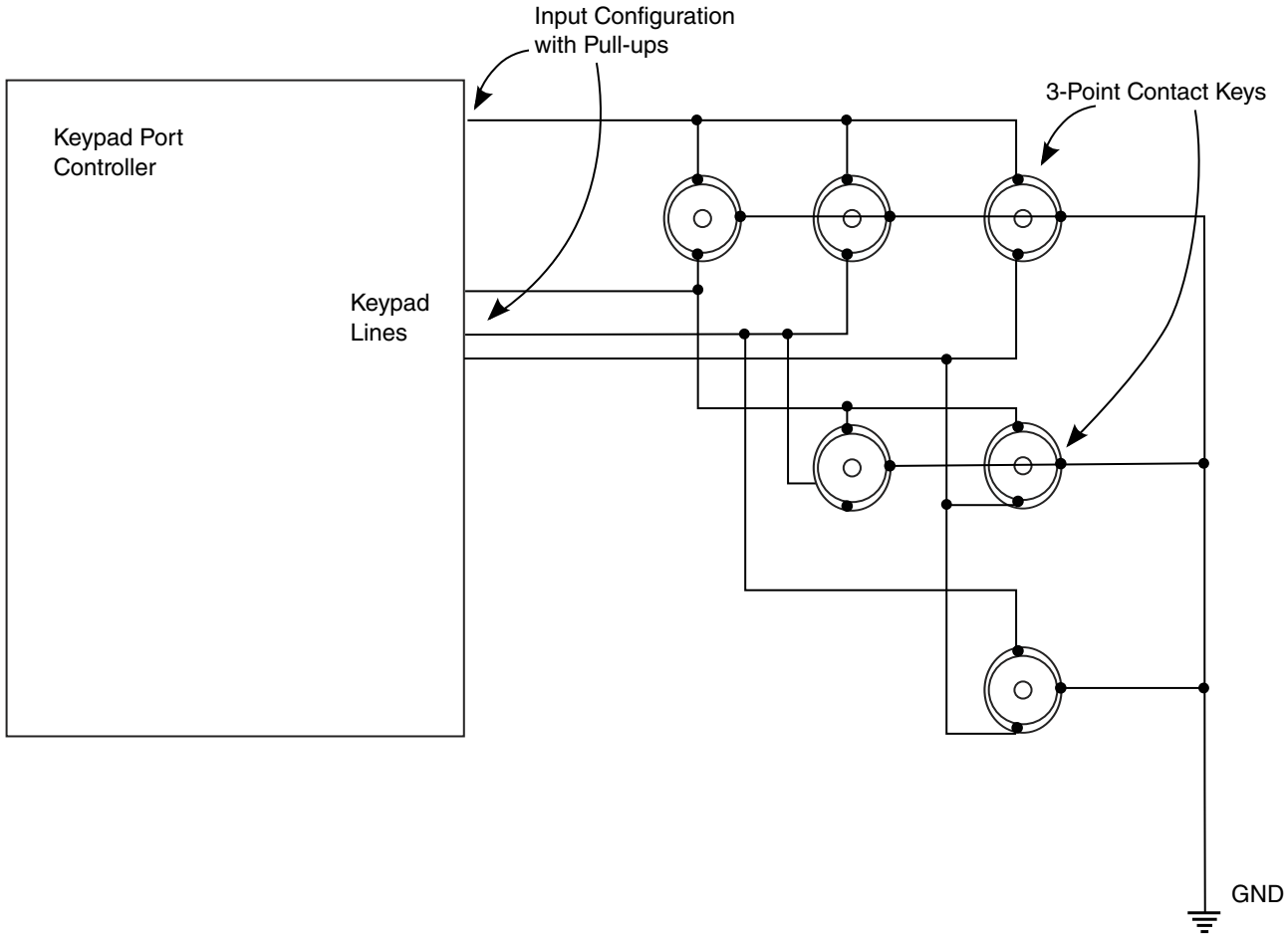


Figure 37-7. KPP Interface with 3-point Contact Key Matrix (Simplified View)

37.6 Initialization/Application Information

37.6.1 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.)

37.6.2 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.

37.6.3 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.

37.7 KPP Memory Map/Register Definition

The KPP contains four registers.

KPP memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
401F_C000	Keypad Control Register (KPP_KPCR)	16	R/W	0000h	37.7.1/1429
401F_C002	Keypad Status Register (KPP_KPSR)	16	R/W	0400h	37.7.2/1430
401F_C004	Keypad Data Direction Register (KPP_KDDR)	16	R/W	0000h	37.7.3/1431
401F_C006	Keypad Data Register (KPP_KPDR)	16	R/W	0000h	37.7.4/1432

37.7.1 Keypad Control Register (KPP_KPCR)

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.

Address: 401F_C000h base + 0h offset = 401F_C000h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	KCO								KRE							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

KPP_KPCR field descriptions

Field	Description
15–8 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. **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. 0 **TOTEM_POLE** — Column strobe output is totem pole drive. 1 **OPEN_DRAIN** — Column strobe output is open drain.
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. 0 Row is not included in the keypad key press detect. 1 Row is included in the keypad key press detect.

37.7.2 Keypad Status Register (KPP_KPSR)

The Keypad Status Register reflects the state of the key press detect circuit. The KPP_KPSR register is byte- or half-word-addressable.

Address: 401F_C000h base + 2h offset = 401F_C002h

Bit	15	14	13	12	11	10	9	8
Read	0						KRIE	KDIE
Write								
Reset	0	0	0	0	0	1	0	0

Bit	7	6	5	4	3	2	1	0
Read	0				0	0	KPKR	KPKD
Write					KRSS	KDSC	w1c	w1c
Reset	0	0	0	0	0	0	0	0

KPP_KPSR field descriptions

Field	Description
15–10 Reserved	This read-only field is reserved and always has the value 0.
9 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. 0 No interrupt request is generated when KPKR is set. 1 An interrupt request is generated when KPKR is set.
8 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. 0 No interrupt request is generated when KPKD is set. 1 An interrupt request is generated when KPKD is set.
7–4 Reserved	This read-only field is reserved and always has the value 0.
3 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". 0 No effect 1 Set bits which sets keypad release synchronizer chain

Table continues on the next page...

KPP_KPSR field descriptions (continued)

Field	Description
2 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". 0 No effect 1 Set bits that clear the keypad depress synchronizer chain
1 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". 0 No key release detected 1 All keys have been released
0 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. 0 No key presses detected 1 A key has been depressed

37.7.3 Keypad Data Direction Register (KPP_KDDR)

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.

Address: 401F_C000h base + 4h offset = 401F_C004h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	KCDD								KRDD							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

KPP_KDDR field descriptions

Field	Description
15–8 KCDD	Keypad Column Data Direction Register. Setting a bit configures the corresponding COL*n* pin as an output (where *n* = 7 through 0). 0 **INPUT** — COLn pin is configured as an input. 1 **OUTPUT** — COLn pin is configured as an output.
KRDD	Keypad Row Data Direction. Setting a bit configures the corresponding ROW*n* pin as an output (where *n* = 7 through 0). 0 **INPUT** — ROWn pin configured as an input. 1 **OUTPUT** — ROWn pin configured as an output.

37.7.4 Keypad Data Register (KPP_KPDR)

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.

Address: 401F_C000h base + 6h offset = 401F_C006h

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	KCD								KRD							
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

KPP_KPDR field descriptions

Field	Description
15–8 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
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

KPP_KPDR field descriptions (continued)

Field	Description

Chapter 38
Low Power Inter-Integrated Circuit (LPI2C)

38.1 Chip-specific LPI2C information

Table 38-1. 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.

38.2 Introduction

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.

- The LPI2C implements logic support for standard-mode, fast-mode, fast-mode plus and ultra-fast modes of operation.
- The LPI2C is designed to use little CPU overhead, with DMA offloading of FIFO register accesses.
- The LPI2C can continue operating in stop modes if an appropriate clock is available.

The LPI2C module also complies with the System Management Bus (SMBus) Specification, version 2. 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.

38.2.1 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 for multiple laves.
  - 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 4words.
- Receive FIFO of 4words.
- 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

38.2.2 Block Diagram

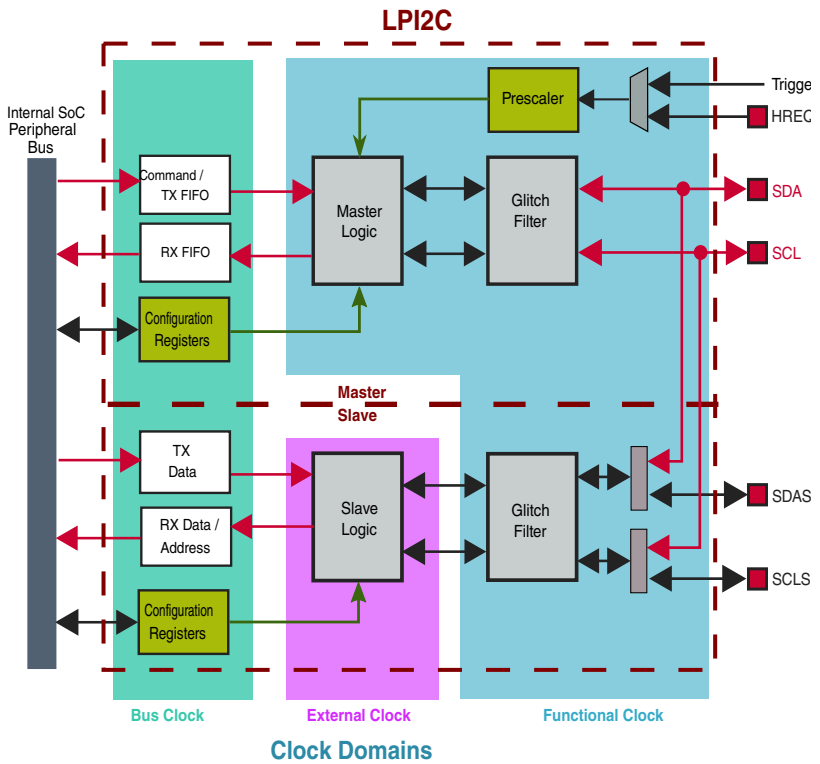


Figure 38-1. LPI2C block diagram

38.2.3 Modes of operation

Table 38-2. Chip modes supported by the LPI2C module

Chip mode	LPI2C Operation
Run	Normal operations
Stop	Can continue operating in stop mode if the Doze Enable bit (MCR[DOZEN]) is clear and the LPI2C is using an external or internal clock source that remains operating during stop mode.
Debug	Can continue operating in debug mode if the Debug Enable bit (MCR[DBGE]) is set.

38.2.4 Signal Descriptions

Table 38-3. 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
HREQ	Host request	If host request is asserted and the I2C bus is idle, then it will initiate an LPI2C master transfer.		I
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 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 the LPI2C slave SDA pin.	I/O

38.3 Functional description

38.3.1 Clocking and Resets

For device-specific clocking information, refer to the Clocking chapter.

Table 38-4. 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.

Table 38-5. 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] will reset all master logic and registers to their default state, except for the MCR itself. • The LPI2C slave implements a software reset bit in its Control Register. The SCR[RST] will reset all slave logic and registers to their default state, except for the SCR 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.

38.3.2 Master Mode

The LPI2C master logic operates independently from the slave logic to perform all master mode transfers on the I2C bus.

38.3.2.1 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 will automatically transmit 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.

38.3.2.2 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 will initiate 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 the Master Clock Configuration Register 0 (MCCR0); if a high speed mode transfer is configured, then the timing configuration from Master Clock Configuration Register 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 will continue 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 will generate a STOP condition automatically.

The LPI2C master can stall the I2C bus under certain conditions; this will result 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.

38.3.2.3 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 will delay 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.

38.3.2.4 Timing Parameters

The following timing parameters can be configured by the LPI2C master. Parameters are configured separately for high speed mode (MCCR1) and other modes (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 38-6. 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)

Table continues on the next page...

Table 38-6. Timing Parameters (continued)

I2C Specification Timing Parameter	I2C Specification Timing Symbol	LPI2C Timing Parameter (LPI2C functional clock cycles)
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)
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 risetime will increase the number of cycles that the signal takes to propagate through the synchronizer (and glitch filter), which increases the latency.

Table 38-7. 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 38-8. 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	-	
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)

Table continues on the next page...

Table 38-8. LPI2C Timing Parameter Restrictions (continued)

Timing Parameter	Minimum	Maximum	Comment
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 will depend 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 38-9. 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

38.3.2.5 Error Conditions

The LPI2C master will monitor for errors while it is active, the following conditions will generate 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[PTLF] flag to terminate the existing command either cleanly (by clearing MCR[MEN]), or abruptly (by setting MCR[SWRST]).

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.

38.3.2.6 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 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 will configure 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.

38.3.3 Slave Mode

To perform all slave mode transfers on the I2C bus, the LPI2C slave logic operates independently from the LPI2C master logic.

38.3.3.1 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 will automatically perform slave-transmit or slave-receive transfers until:

- a NACK is detected (unless 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

38.3.3.2 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 RXCFG=1).
- The Address Status register should only be read when the Address Valid flag is set.

38.3.3.3 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 will extend for one peripheral bus clock cycle after SDA updates when clock stretching is enabled.

38.3.3.4 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.

38.3.3.5 Error Conditions

The LPI2C slave can detect the following error conditions:

- Bit error flag will set when the LPI2C slave is driving SDA, but samples a different value than what is expected.
- FIFO error flag will set 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 will also set due to an address overrun when 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.

38.3.4 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

38.3.4.1 Master mode

The next table lists the master mode sources that can generate LPI2C master interrupts and LPI2C master transmit/receive DMA requests.

Table 38-10. Master Interrupts and DMA Requests

Master Status Register (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	Y	N	Y

Table continues on the next page...

Table 38-10. Master Interrupts and DMA Requests (continued)

Master Status Register (MSR)		Description	Can generate		
Flag	Name		Interrupt?	DMA Request?	Low Power Wakeup?
		• During an address byte, the master expected a NACK but detected an ACK • During a master-transmitter data byte, the master detected a NACK			
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 I2C bus, but a STOP condition has not been detected and a bus idle timeout (if enabled) has not occurred.	N	N	N

38.3.4.2 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 38-11. Slave Interrupts and DMA Requests

Slave Status Register (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 Register (STDR)	Y	TX	Y
RDF	Receive Data Flag	Data can be read from the Slave Receive Data Register (SRDR)	Y	RX	Y
AVF	Address Valid Flag	Address can be read from the Slave Address Status Register (SASR)	Y	RX	Y

Table continues on the next page...

Table 38-11. Slave Interrupts and DMA Requests (continued)

Slave Status Register (SSR)		Description	Can generate		
Flag	Name		Interrupt?	DMA Request?	Low Power Wakeup?
TAF	Transmit ACK Flag	ACK/NACK can be written to the Slave Transmit ACK Register (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

38.3.5 Peripheral Triggers

The connection of the LPI2C peripheral triggers to other peripherals depend upon the specific device being used.

Table 38-12. 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.

38.4 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.

38.4.1 LPI2C register descriptions

38.4.1.1 LPI2C Memory map

LPI2C1 base address: 401A_4000h

LPI2C2 base address: 401A_8000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Version ID Register (VERID)	32	RO	0100_0003h
4h	Parameter Register (PARAM)	32	RO	0000_0202h
10h	Master Control Register (MCR)	32	RW	0000_0000h
14h	Master Status Register (MSR)	32	W1C	0000_0001h
18h	Master Interrupt Enable Register (MIER)	32	RW	0000_0000h
1Ch	Master DMA Enable Register (MDER)	32	RW	0000_0000h
20h	Master Configuration Register 0 (MCFGR0)	32	RW	0000_0000h
24h	Master Configuration Register 1 (MCFGR1)	32	RW	0000_0000h
28h	Master Configuration Register 2 (MCFGR2)	32	RW	0000_0000h
2Ch	Master Configuration Register 3 (MCFGR3)	32	RW	0000_0000h
40h	Master Data Match Register (MDMR)	32	RW	0000_0000h
48h	Master Clock Configuration Register 0 (MCCR0)	32	RW	0000_0000h
50h	Master Clock Configuration Register 1 (MCCR1)	32	RW	0000_0000h
58h	Master FIFO Control Register (MFCR)	32	RW	0000_0000h
5Ch	Master FIFO Status Register (MFSR)	32	RO	0000_0000h
60h	Master Transmit Data Register (MTDR)	32	WO	0000_0000h
70h	Master Receive Data Register (MRDR)	32	RO	0000_4000h
110h	Slave Control Register (SCR)	32	RW	0000_0000h
114h	Slave Status Register (SSR)	32	W1C	0000_0000h
118h	Slave Interrupt Enable Register (SIER)	32	RW	0000_0000h
11Ch	Slave DMA Enable Register (SDER)	32	RW	0000_0000h
124h	Slave Configuration Register 1 (SCFGR1)	32	RW	0000_0000h
128h	Slave Configuration Register 2 (SCFGR2)	32	RW	0000_0000h
140h	Slave Address Match Register (SAMR)	32	RW	0000_0000h
150h	Slave Address Status Register (SASR)	32	RO	0000_4000h
154h	Slave Transmit ACK Register (STAR)	32	RW	0000_0000h
160h	Slave Transmit Data Register (STDR)	32	WO	0000_0000h
170h	Slave Receive Data Register (SRDR)	32	RO	0000_4000h

38.4.1.2 Version ID Register (VERID)

38.4.1.2.1 Offset

Register	Offset
VERID	0h

38.4.1.2.2 Function

Contains version numbers for the module design and feature set.

38.4.1.2.3 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

38.4.1.2.4 Fields

Field	Function
31-24	Major Version Number
MAJOR	Returns the major version number for the module design specification. Read-only field.
23-16	Minor Version Number
MINOR	Returns the minor version number for the module design specification. Read-only field.
15-0	Feature Specification Number
FEATURE	Returns the feature set number. Read-only field. 0000000000000010b - Master only, with standard feature set 0000000000000011b - Master and slave, with standard feature set

38.4.1.3 Parameter Register (PARAM)

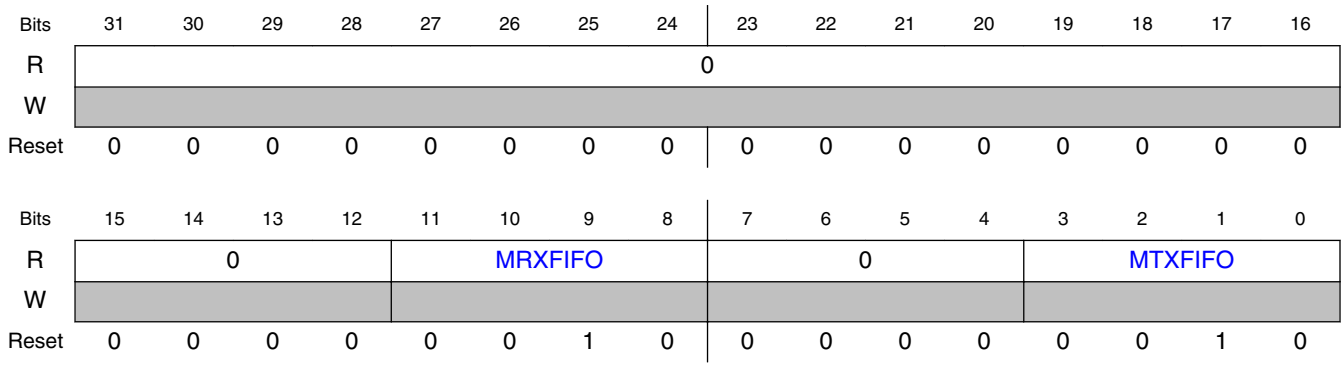
38.4.1.3.1 Offset

Register	Offset
PARAM	4h

38.4.1.3.2 Function

Contains parameter values that were implemented in the module.

38.4.1.3.3 Diagram



38.4.1.3.4 Fields

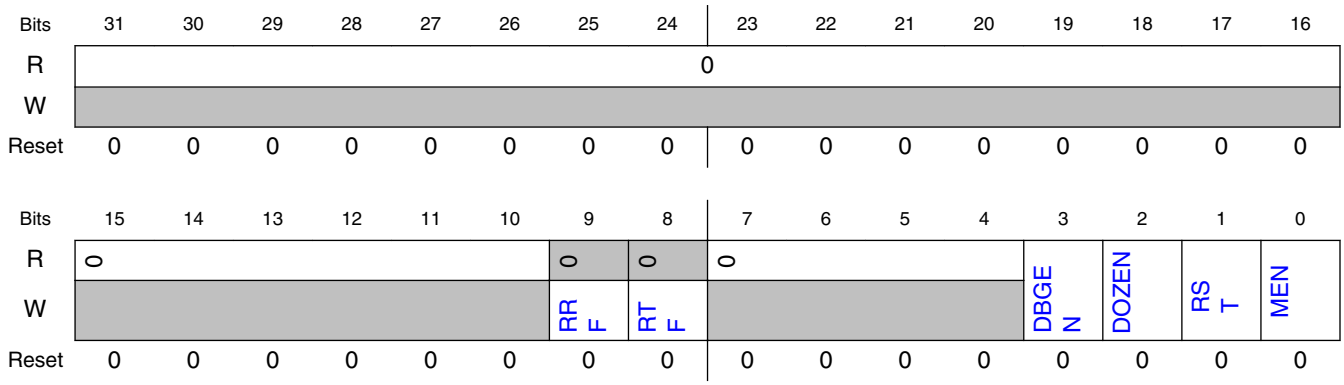
Field	Function
31-16 —	Reserved
15-12 —	Reserved
11-8 MRXFIFO	Master Receive FIFO Size Configures the number of words in the master receive FIFO to $2^{MRXFIFO}$
7-4 —	Reserved
3-0 MTXFIFO	Master Transmit FIFO Size Configures the number of words in the master transmit FIFO to $2^{MTXFIFO}$

38.4.1.4 Master Control Register (MCR)

38.4.1.4.1 Offset

Register	Offset
MCR	10h

38.4.1.4.2 Diagram



38.4.1.4.3 Fields

Field	Function
31-10 —	Reserved
9 RRF	Reset Receive FIFO 0b - No effect 1b - Receive FIFO is reset
8 RTF	Reset Transmit FIFO 0b - No effect 1b - Transmit FIFO is reset
7-4 —	Reserved
3 DBGEN	Debug Enable 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 Reset all internal master logic and registers, except the Master Control Register. RST remains set until cleared by software. 0b - Master logic is not reset 1b - Master logic is reset
0 MEN	Master Enable 0b - Master logic is disabled 1b - Master logic is enabled

38.4.1.5 Master Status Register (MSR)

38.4.1.5.1 Offset

Register	Offset
MSR	14h

38.4.1.5.2 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

38.4.1.5.3 Fields

Field	Function
31-26 —	Reserved
25 BBF	Bus Busy Flag 0b - I2C Bus is idle 1b - I2C Bus is busy
24 MBF	Master Busy Flag 0b - I2C Master is idle 1b - I2C Master is busy
23-15 —	Reserved
14 DMF	Data Match Flag Indicates that the received data has matched the MATCH0 and/or MATCH1 fields (as configured by MCFGR1[MATCFG], Match Configuration). Received data *that is discarded due to CMD field* does not cause Data Match Flag to set. 0b - Have not received matching data 1b - Have received matching data
13	Pin Low Timeout Flag

Table continues on the next page...

Field	Function
PLTF	Will set 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. • Pin Low Timeout Flag cannot be cleared as long as the pin low timeout continues. • Before the LPI2C can initiate a START condition, the Pin Low Timeout Flag must be cleared. 0b - Pin low timeout has not occurred or is disabled 1b - Pin low timeout has occurred
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 the AUTOSTOP bit is set. When FIFO Error Flag is set, the LPI2C master will send a STOP condition (if busy), and will not initiate a new START condition until FIFO Error Flag has been cleared. 0b - No error 1b - Master sending or receiving data without a START condition
11 ALF	Arbitration Lost Flag Set: • if the LPI2C master transmits a logic one and detects a logic zero on the I2C bus • or if the LPI2C master detects a START or STOP condition while the LPI2C master is transmitting data When the Arbitration Lost Flag sets, the LPI2C master will release the I2C bus (go idle), and the LPI2C master will not initiate a new START condition until the Arbitration Lost Flag 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 when transmitting an address or data. When set, the master will transmit a STOP condition and will not initiate a new START condition until NACK Detect Flag has been cleared. If a NACK is expected for a given address (as configured by the command word), then the NACK Detect Flag will set if a NACK is not generated. 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 The Receive Data Flag is set whenever the number of words in the receive FIFO is greater than RXWATER. 0b - Receive Data is not ready 1b - Receive data is ready
0 TDF	Transmit Data Flag

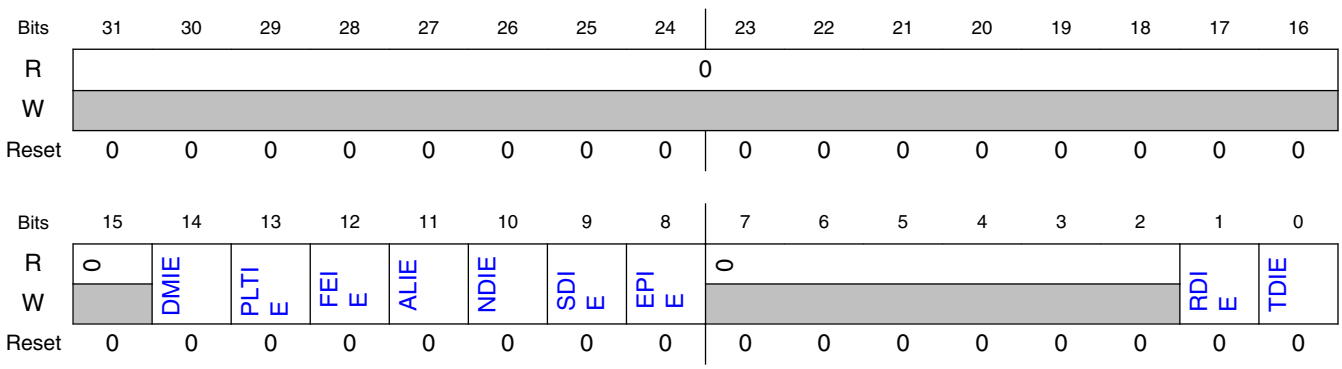
Field	Function
	The Transmit Data Flag is set whenever the number of words in the transmit FIFO is equal or less than TXWATER. 0b - Transmit data is not requested 1b - Transmit data is requested

38.4.1.6 Master Interrupt Enable Register (MIER)

38.4.1.6.1 Offset

Register	Offset
MIER	18h

38.4.1.6.2 Diagram



38.4.1.6.3 Fields

Field	Function
31-15 —	Reserved
14 DMIE	Data Match Interrupt Enable 0b - Disabled 1b - Enabled
13 PLTIE	Pin Low Timeout Interrupt Enable 0b - Disabled 1b - Enabled
12 FEIE	FIFO Error Interrupt Enable 0b - Enabled 1b - Disabled

Table continues on the next page...

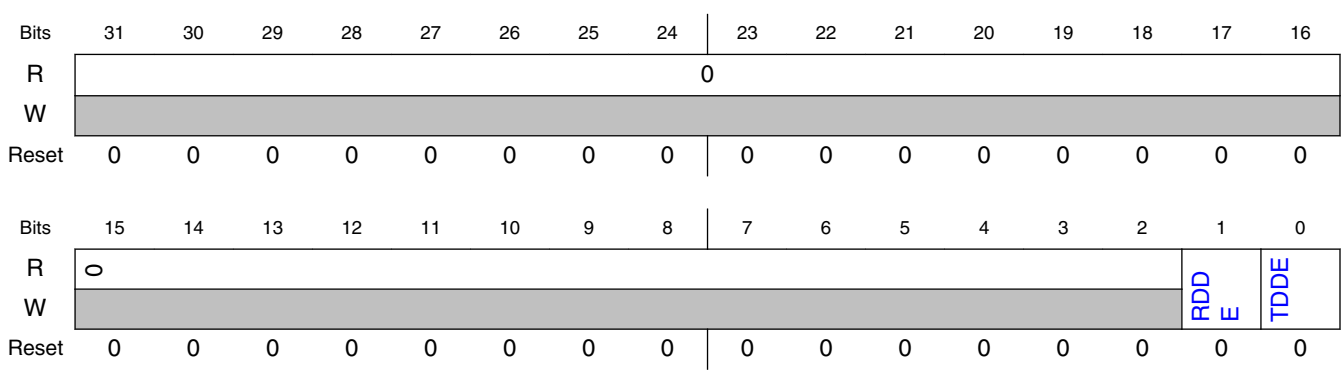
Field	Function
11 ALIE	Arbitration Lost Interrupt Enable 0b - Disabled 1b - Enabled
10 NDIE	NACK Detect Interrupt Enable 0b - Disabled 1b - Enabled
9 SDIE	STOP Detect Interrupt Enable 0b - Disabled 1b - Enabled
8 EPIE	End Packet Interrupt Enable 0b - Disabled 1b - Enabled
7-2 —	Reserved
1 RDIE	Receive Data Interrupt Enable 0b - Disabled 1b - Enabled
0 TDIE	Transmit Data Interrupt Enable 0b - Disabled 1b - Enabled

38.4.1.7 Master DMA Enable Register (MDER)

38.4.1.7.1 Offset

Register	Offset
MDER	1Ch

38.4.1.7.2 Diagram



38.4.1.7.3 Fields

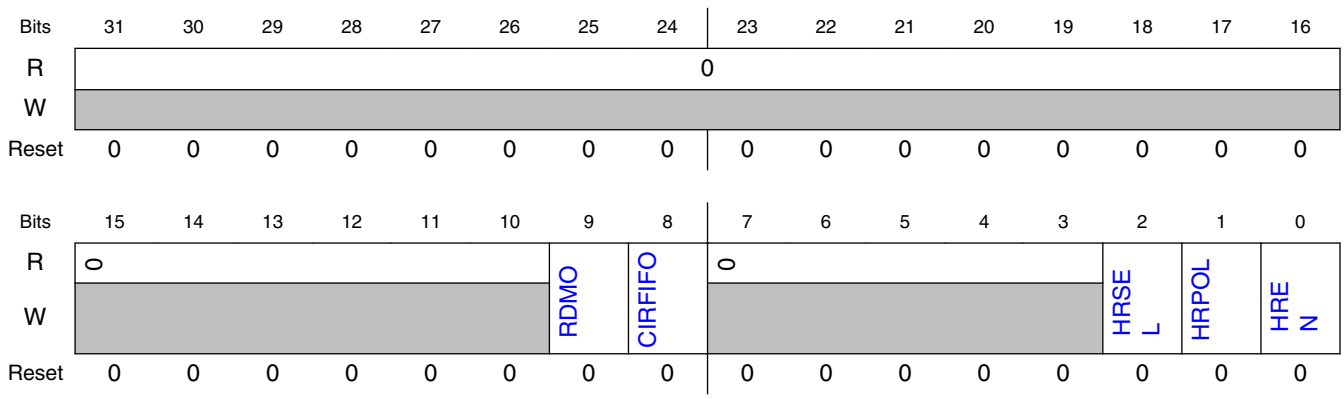
Field	Function
31-2 —	Reserved
1 RDDE	Receive Data DMA Enable 0b - DMA request is disabled 1b - DMA request is enabled
0 TDDE	Transmit Data DMA Enable 0b - DMA request is disabled 1b - DMA request is enabled

38.4.1.8 Master Configuration Register 0 (MCFGR0)

38.4.1.8.1 Offset

Register	Offset
MCFGR0	20h

38.4.1.8.2 Diagram



38.4.1.8.3 Fields

Field	Function
31-10 —	Reserved
9	Receive Data Match Only

Table continues on the next page...

Field	Function
RDMO	When enabled, all received data that does not cause the Data Match Flag (MSR[DMF]) to set is discarded. After the Data Match Flag is set, the RDMO configuration is ignored. When disabling RDMO, clear RDMO before clearing the Data Match Flag, 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 will be emptied 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 will cause the contents of the transmit FIFO to be cycled through repeatedly. If AUTOSTOP is set, then a STOP condition will be sent whenever the transmit FIFO is empty and the read pointer is restored. 0b - Circular FIFO is disabled 1b - Circular FIFO is enabled
7-3 —	Reserved
2 HRSEL	Host Request Select Selects the source of the host request input. When host request input is enabled, the Host Request Select field should remain static (the Host Request Select should not change). 0b - Host request input is pin HREQ 1b - Host request input is input trigger
1 HRPOL	Host Request Polarity Configures the polarity of the host request input pin. When host request input is enabled, the Host Request Polarity field 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 will only initiate 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

38.4.1.9 Master Configuration Register 1 (MCFGR1)

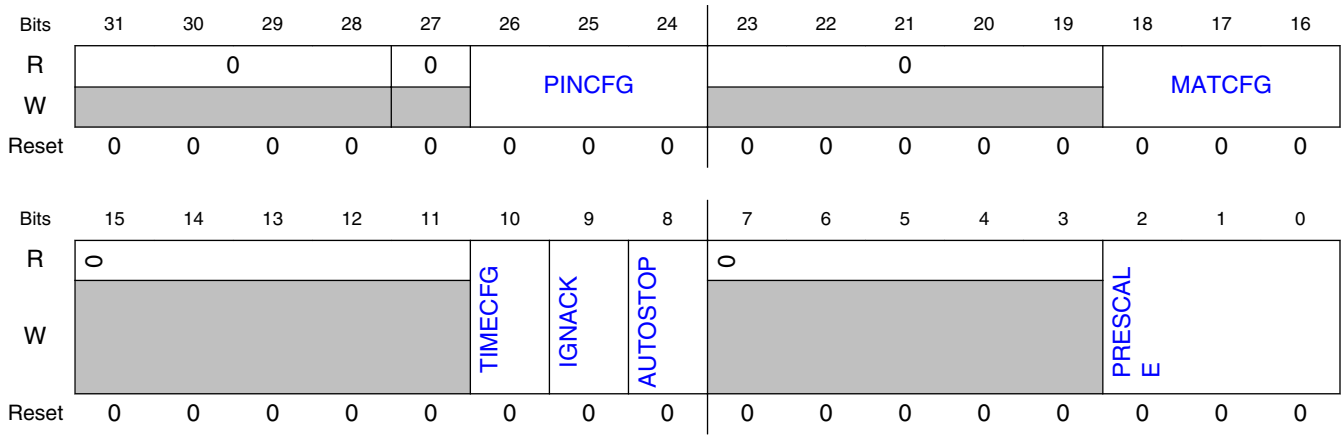
38.4.1.9.1 Offset

Register	Offset
MCFGR1	24h

38.4.1.9.2 Function

The MCFGR1 should only be written when the I2C Master is disabled.

38.4.1.9.3 Diagram



38.4.1.9.4 Fields

Field	Function
31-28 —	Reserved
27 —	Reserved
26-24 PINCFG	Pin Configuration Configures the pin mode for LPI2C. **Table 38-13. 2-pin / 4-pin pin configurations for masters and slaves** 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

PINCFG	SCL / SDA pins	SCLS / SDAS pins
000	Bi-directional open drain for master and slave	Not used
001	Output-only (ultra-fast mode) open drain for master and slave	Not used
010	Bi-directional push-pull for master and slave	Not used
011	Input only for master and slave	Output-only push-pull for master and slave
100	Bi-directional open drain for master	Bi-directional open drain for slave
101	Output-only (ultra-fast mode) open drain for master	Output-only open drain for slave
110	Bi-directional push-pull for master	Bi-directional push-pull for slave
111	Input only for master and slave	Inverted output-only push-pull for master and slave

Table continues on the next page...

Field	Function
	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)
23-19 —	Reserved
18-16 MATCFG	Match Configuration Configures the condition that will cause the DMF to set. 000b - Match is disabled 001b - Reserved 010b - Match is enabled (1st data word equals MATCH0 OR MATCH1) 011b - Match is enabled (any data word equals MATCH0 OR MATCH1) 100b - Match is enabled (1st data word equals MATCH0 AND 2nd data word equals MATCH1) 101b - Match is enabled (any data word equals MATCH0 AND next data word equals MATCH1) 110b - Match is enabled (1st data word AND MATCH1 equals MATCH0 AND MATCH1) 111b - Match is enabled (any data word AND MATCH1 equals MATCH0 AND MATCH1)
15-11 —	Reserved
10 TIMECFG	Timeout Configuration 0b - Pin Low Timeout Flag will set if SCL is low for longer than the configured timeout 1b - Pin Low Timeout Flag will set 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 will receive ACK and NACK normally 1b - LPI2C Master will treat 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

38.4.1.10 Master Configuration Register 2 (MCFGR2)

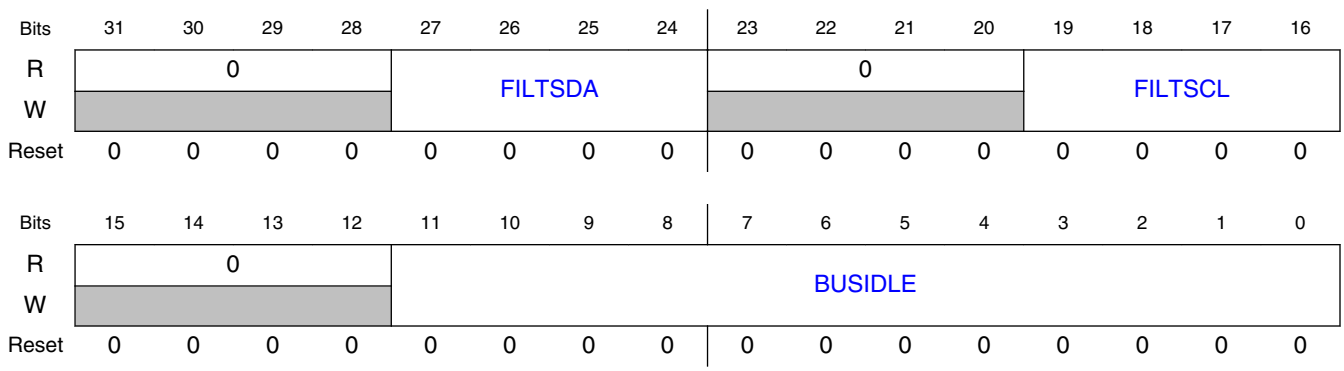
38.4.1.10.1 Offset

Register	Offset
MCFGR2	28h

38.4.1.10.2 Function

The Master Configuration Register 2 should only be written when the I2C Master is disabled.

38.4.1.10.3 Diagram



38.4.1.10.4 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 will disable the glitch filter • Glitches equal to or less than FILTSDA cycles long will be 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.

Table continues on the next page...

Field	Function
	• A configuration of 0 will disable the glitch filter • Glitches equal to or less than FILTSCL cycles long will be filtered out and ignored • 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. • 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

38.4.1.11 Master Configuration Register 3 (MCFGR3)

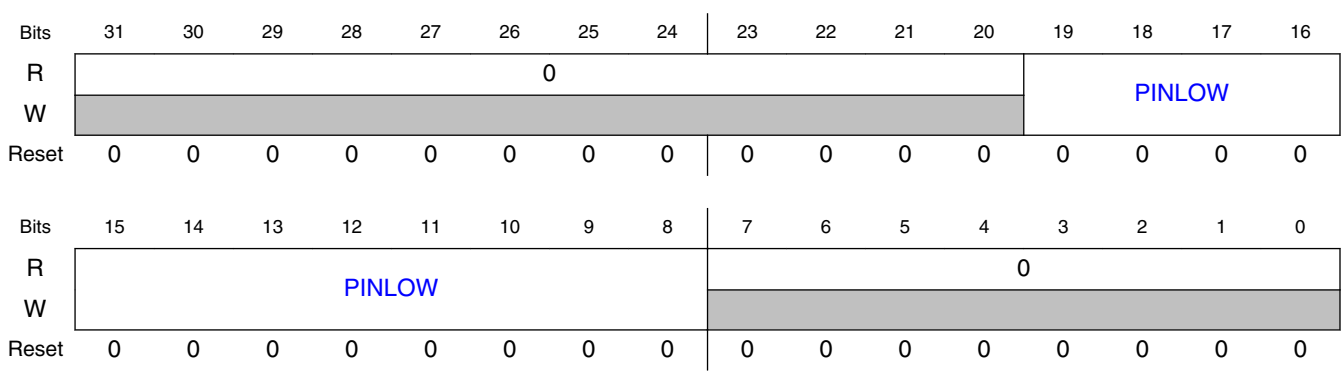
38.4.1.11.1 Offset

Register	Offset
MCFGR3	2Ch

38.4.1.11.2 Function

The MCFGR3 should only be written when the I2C Master is disabled.

38.4.1.11.3 Diagram



38.4.1.11.4 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 PLTF is set • When Pin Low Timeout is set to zero, the Pin Low Timeout feature is disabled
7-0 —	Reserved

38.4.1.12 Master Data Match Register (MDMR)

38.4.1.12.1 Offset

Register	Offset
MDMR	40h

38.4.1.12.2 Function

The MDMR should only be written when the I2C Master is disabled or idle.

38.4.1.12.3 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

38.4.1.12.4 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.

38.4.1.13 Master Clock Configuration Register 0 (MCCR0)

38.4.1.13.1 Offset

Register	Offset
MCCR0	48h

38.4.1.13.2 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.

38.4.1.13.3 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

38.4.1.13.4 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 • as the hold time for a START condition • as the setup and hold time for a repeated START condition • as the 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 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.

38.4.1.14 Master Clock Configuration Register 1 (MCCR1)

38.4.1.14.1 Offset

Register	Offset
MCCR1	50h

38.4.1.14.2 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 MCCR0), before switching to high speed mode (with timing configured by MCCR1).

38.4.1.14.3 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

38.4.1.14.4 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 Data Valid Delay 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 • as the hold time for a START condition • as the setup and hold time for a repeated START condition • as the 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.

Table continues on the next page...

Field	Function
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.

38.4.1.15 Master FIFO Control Register (MFCR)

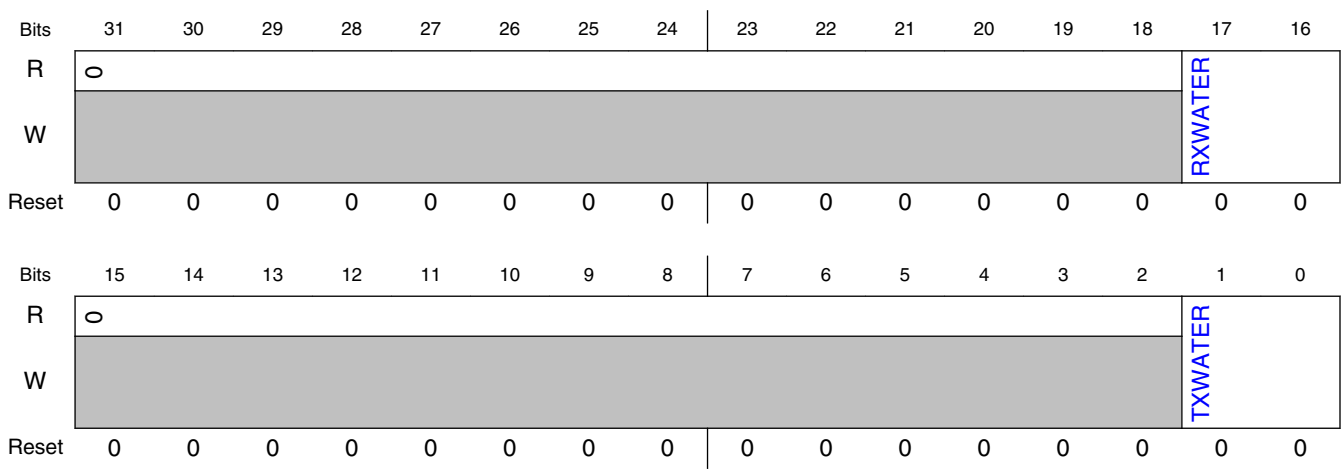
38.4.1.15.1 Offset

Register	Offset
MFCR	58h

38.4.1.15.2 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).

38.4.1.15.3 Diagram



38.4.1.15.4 Fields

Field	Function
31-18 —	Reserved
17-16 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 to or greater than the FIFO size will be truncated.
15-2 —	Reserved
1-0 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 to or greater than the FIFO size will be truncated.

38.4.1.16 Master FIFO Status Register (MFSR)

38.4.1.16.1 Offset

Register	Offset
MFSR	5Ch

38.4.1.16.2 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

38.4.1.16.3 Fields

Field	Function
31-19	Reserved

Table continues on the next page...

Field	Function
—	
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.

38.4.1.17 Master Transmit Data Register (MTDR)

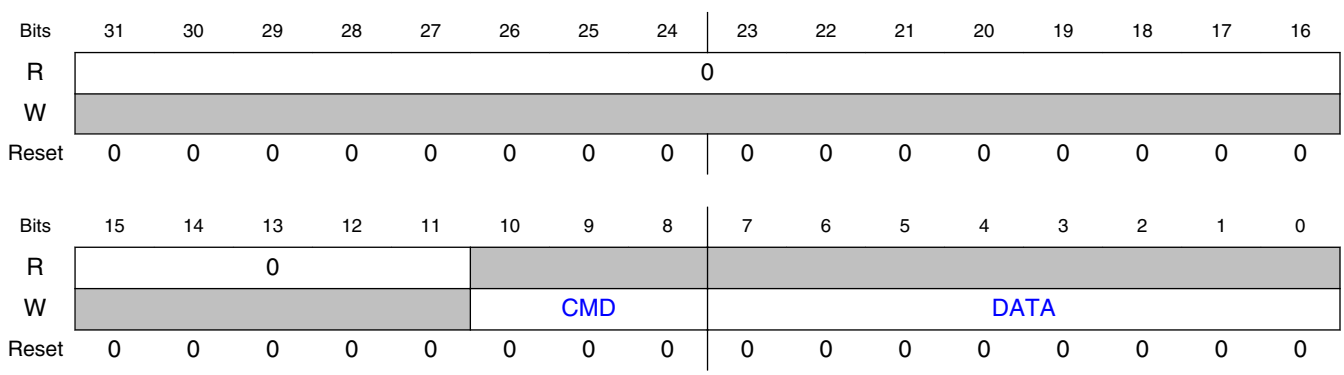
38.4.1.17.1 Offset

Register	Offset
MTDR	60h

38.4.1.17.2 Function

- An 8-bit write to the CMD field will store the data in the Command FIFO, but does not increment the FIFO write pointer.
- An 8-bit write to the DATA field will zero extend the CMD field, unless the CMD field has been written separately since the last FIFO write; it (the 8-bit write) also increments the FIFO write pointer.
- A 16-bit or 32-bit will write both the CMD and DATA fields and increment the FIFO.

38.4.1.17.3 Diagram



38.4.1.17.4 Fields

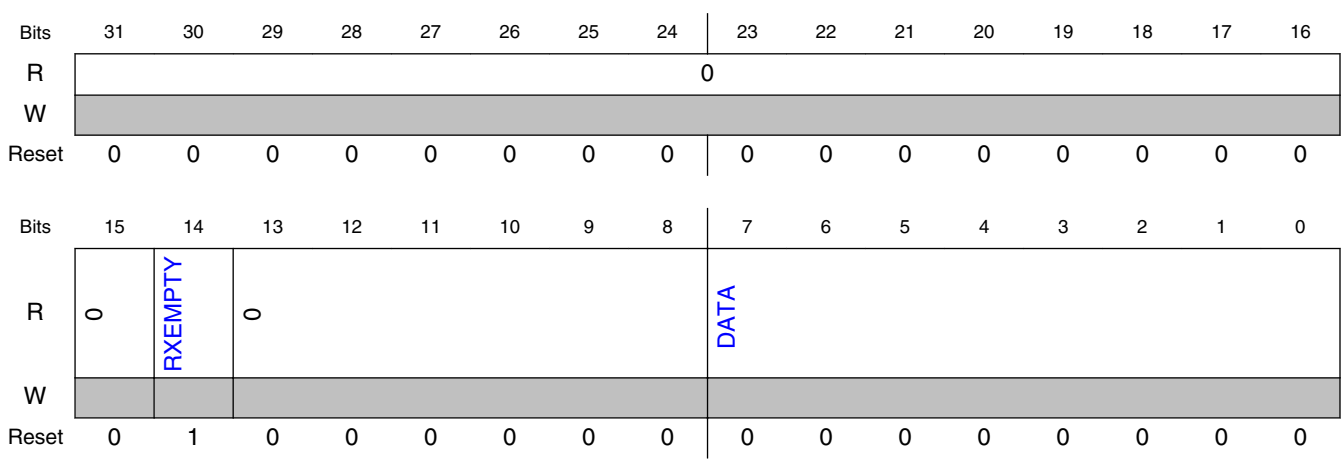
Field	Function
31-11 —	Reserved
10-8 CMD	Command Data 000b - Transmit DATA[7:0] 001b - Receive (DATA[7:0] + 1) bytes 010b - Generate STOP condition 011b - Receive and discard (DATA[7:0] + 1) bytes 100b - Generate (repeated) START and transmit address in DATA[7:0] 101b - Generate (repeated) START and 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 will zero extend the CMD field.

38.4.1.18 Master Receive Data Register (MRDR)

38.4.1.18.1 Offset

Register	Offset
MRDR	70h

38.4.1.18.2 Diagram



38.4.1.18.3 Fields

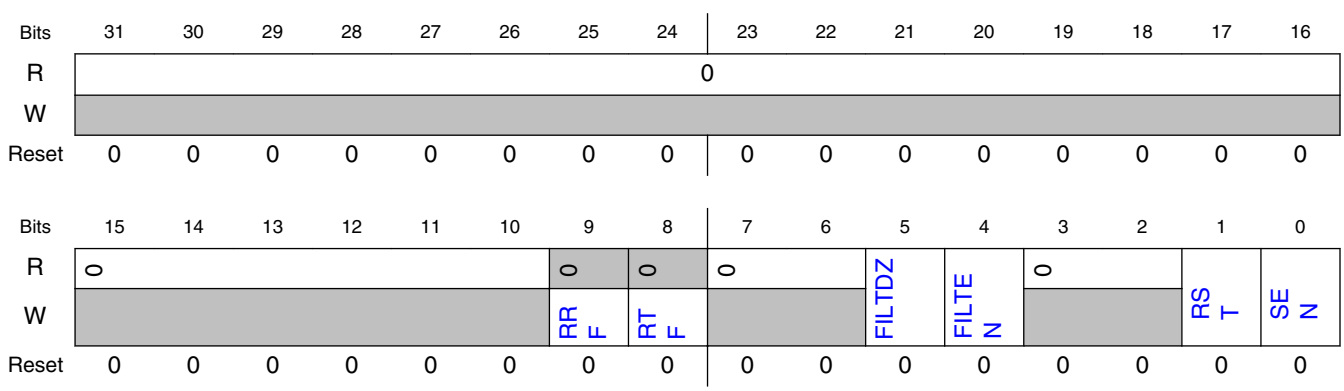
Field	Function
31-15 —	Reserved
14 RXEMPTY	RX Empty 0b - Receive FIFO is not empty 1b - Receive FIFO is empty
13-8 —	Reserved
7-0 DATA	Receive Data Reading the Receive Data register returns the data received by the I2C master that has not been discarded. Receive data can be discarded due to the CMD field, or the master can be configured to discard non-matching data.

38.4.1.19 Slave Control Register (SCR)

38.4.1.19.1 Offset

Register	Offset
SCR	110h

38.4.1.19.2 Diagram



38.4.1.19.3 Fields

Field	Function
31-10 —	Reserved
9 RRF	Reset Receive FIFO 0b - No effect 1b - Receive Data Register is now empty
8 RTF	Reset Transmit FIFO 0b - No effect 1b - Transmit Data Register is now empty
7-6 —	Reserved
5 FILTDZ	Filter Doze Enable Filter Doze Enable bit 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 Filter Enable bit 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 0b - Slave mode logic is not reset 1b - Slave mode logic is reset
0 SEN	Slave Enable 0b - I2C Slave mode is disabled 1b - I2C Slave mode is enabled

38.4.1.20 Slave Status Register (SSR)

38.4.1.20.1 Offset

Register	Offset
SSR	114h

38.4.1.20.2 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

38.4.1.20.3 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 • SMBus Alert Response Flag is cleared by reading the Address Status Register • SMBus Alert Response Flag 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. • General Call Flag is cleared by reading the Address Status 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 ADDRCFG.

Table continues on the next page...

Field	Function
	• Address Match 1 Flag is cleared by reading the Address Status 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 ADDRCFG. • Address Match 0 Flag is cleared by reading the Address Status Register • Address Match 0 Flag 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 FIFO error flag 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 Bit Error Flag will set if the LPI2C slave transmits a logic one and detects a logic zero on the I2C bus. The slave will ignore 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 STOP Detect Flag will set 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 Repeated Start Flag will set when the LPI2C slave detects a repeated START condition and if the LPI2C slave matched the last address byte. The Repeated Start Flag 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 Transmit ACK Flag is cleared by writing the Transmit ACK register. 0b - Transmit ACK/NACK is not required 1b - Transmit ACK/NACK is required
2 AVF	Address Valid Flag Address Valid Flag is cleared by reading the address status register. When Receive Data Configuration (SCFGR1[RXCFG]) is set, the Address Valid Flag is also cleared by reading the Receive Data register. 0b - Address Status Register is not valid 1b - Address Status Register is valid
1 RDF	Receive Data Flag Receive Data Flag is cleared by reading the receive data register. When Receive Data Configuration (SCFGR1[RXCFG]) is set, the Receive Data Flag is not cleared when reading the Receive Data register and if AVF is set. 0b - Receive data is not ready 1b - Receive data is ready

Table continues on the next page...

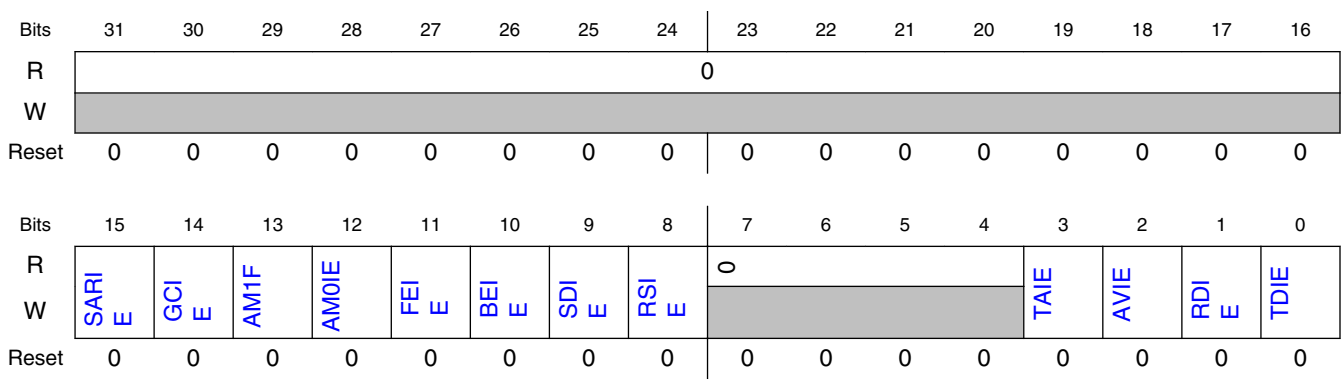
Field	Function
0 TDF	Transmit Data Flag Transmit Data Flag is cleared by writing the Transmit Data register. When Transmit Flag Configuration (TXCFG) is clear, and if a NACK or Repeated START or STOP condition is detected, then Transmit Data Flag is also cleared. 0b - Transmit data not requested 1b - Transmit data is requested

38.4.1.21 Slave Interrupt Enable Register (SIER)

38.4.1.21.1 Offset

Register	Offset
SIER	118h

38.4.1.21.2 Diagram



38.4.1.21.3 Fields

Field	Function
31-16 —	Reserved
15 SARIE	SMBus Alert Response Interrupt Enable 0b - Disabled 1b - Enabled
14 GCIE	General Call Interrupt Enable 0b - Disabled 1b - Enabled

Table continues on the next page...

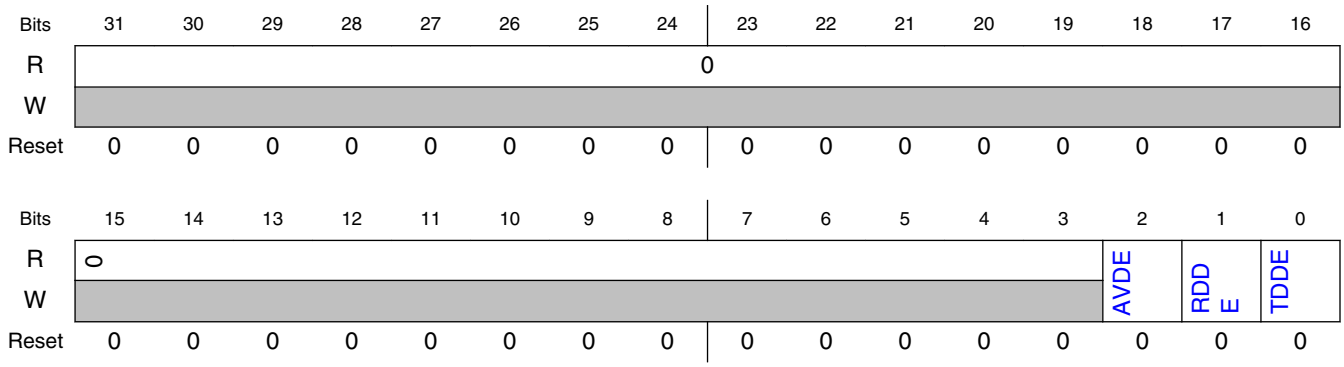
Field	Function
13 AM1F	Address Match 1 Interrupt Enable 0b - Disabled 1b - Enabled
12 AM0IE	Address Match 0 Interrupt Enable 0b - Enabled 1b - Disabled
11 FEIE	FIFO Error Interrupt Enable 0b - Disabled 1b - Enabled
10 BEIE	Bit Error Interrupt Enable 0b - Disabled 1b - Enabled
9 SDIE	STOP Detect Interrupt Enable 0b - Disabled 1b - Enabled
8 RSIE	Repeated Start Interrupt Enable 0b - Disabled 1b - Enabled
7-4 —	Reserved
3 TAIE	Transmit ACK Interrupt Enable 0b - Disabled 1b - Enabled
2 AVIE	Address Valid Interrupt Enable 0b - Disabled 1b - Enabled
1 RDIE	Receive Data Interrupt Enable 0b - Disabled 1b - Enabled
0 TDIE	Transmit Data Interrupt Enable 0b - Disabled 1b - Enabled

38.4.1.22 Slave DMA Enable Register (SDER)

38.4.1.22.1 Offset

Register	Offset
SDER	11Ch

38.4.1.22.2 Diagram



38.4.1.22.3 Fields

Field	Function
31-3 —	Reserved
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 Receive Data Configuration (SCFGR1[RXCFG]), to allow the DMA to read the address from the Receive Data Register. 0b - DMA request is disabled 1b - DMA request is enabled
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

38.4.1.23 Slave Configuration Register 1 (SCFGR1)

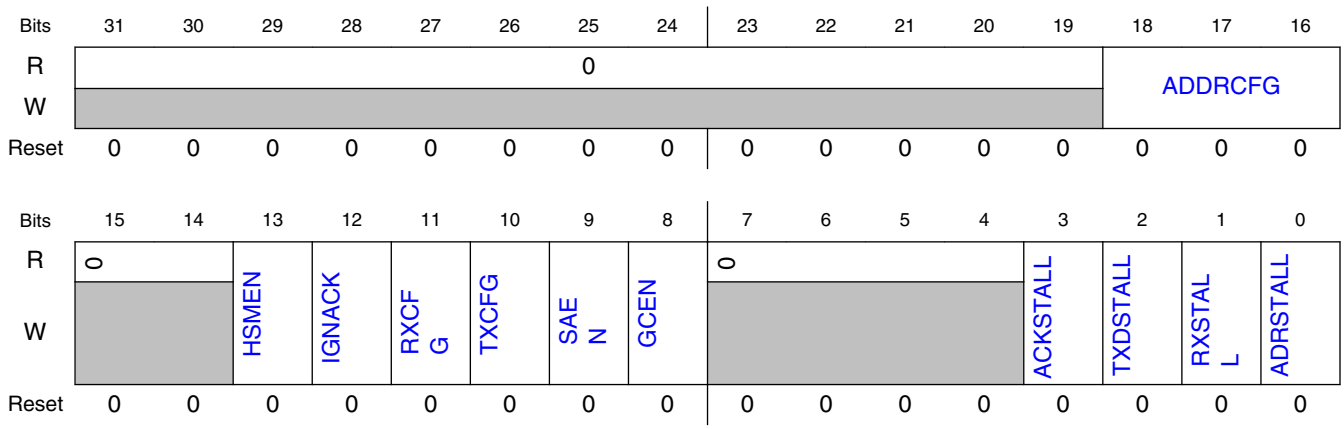
38.4.1.23.1 Offset

Register	Offset
SCFGR1	124h

38.4.1.23.2 Function

The Slave Configuration Register 1 should only be written when the I2C Slave is disabled.

38.4.1.23.3 Diagram



38.4.1.23.4 Fields

Field	Function
31-19 —	Reserved
18-16 ADDRCFG	Address Configuration Configures the condition that will cause 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, the FILTEN and 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 will continue transfers after a NACK is detected. Ignore NACK bit is required to be set in Ultra-Fast Mode. 0b - Slave will end transfer when NACK is detected

Table continues on the next page...

Field	Function
	1b - Slave will not end transfer when NACK detected
11 RXCFG	Receive Data Configuration 0b - Reading the Receive Data register will return received data and clear the Receive Data flag (MSR[RDF]). 1b - Reading the Receive Data register when the Address Valid flag (SSR[AVF])is set, will return the Address Status register and clear the Address Valid flag. Reading the Receive Data register when the Address Valid flag is clear, will return received data and clear the Receive Data flag (MSR[RDF]).
10 TXCFG	Transmit Flag Configuration The transmit data flag will always assert 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 will assert whenver the Transmit Data register is empty, and the Transmit Data flag will negate 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. 0b - Transmit Data Flag will only assert during a slave-transmit transfer when the Transmit Data register is empty 1b - Transmit Data Flag will assert whenever the Transmit Data register is empty
9 SAEN	SMBus Alert Enable 0b - Disables match on SMBus Alert 1b - Enables match on SMBus Alert
8 GCEN	General Call Enable 0b - General Call address is disabled 1b - General Call address is enabled
7-4 —	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 Transmit ACK 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 Transmit Data flag (SSR[TDF]) is set 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 the Receive Data flag (SSR[RDF]) is set 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

Table continues on the next page...

Field	Function
0 ADRSTALL	Address SCL Stall Enables SCL clock stretching when the Address Valid Flag (SSR[AVF]) is asserted. 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

38.4.1.24 Slave Configuration Register 2 (SCFGR2)

38.4.1.24.1 Offset

Register	Offset
SCFGR2	128h

38.4.1.24.2 Function

The Slave Configuration Register 2 should only be written when the I2C Slave is disabled.

38.4.1.24.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0				FILTSDA				0				FILTSCL			
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		DATAVD						0				CLKHOLD			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

38.4.1.24.4 Fields

Field	Function
31-28 —	Reserved
27-24	Glitch Filter SDA

Table continues on the next page...

Field	Function
FILTSDA	Configures the I2C slave digital glitch filters for SDA input. • A configuration of 0 will disable the glitch filter • Glitches equal to or less than FILTSDA cycles long will be 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 will disable the glitch filter • Glitches equal to or less than FILTSCL cycles long will be filtered out and ignored • 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

38.4.1.25 Slave Address Match Register (SAMR)

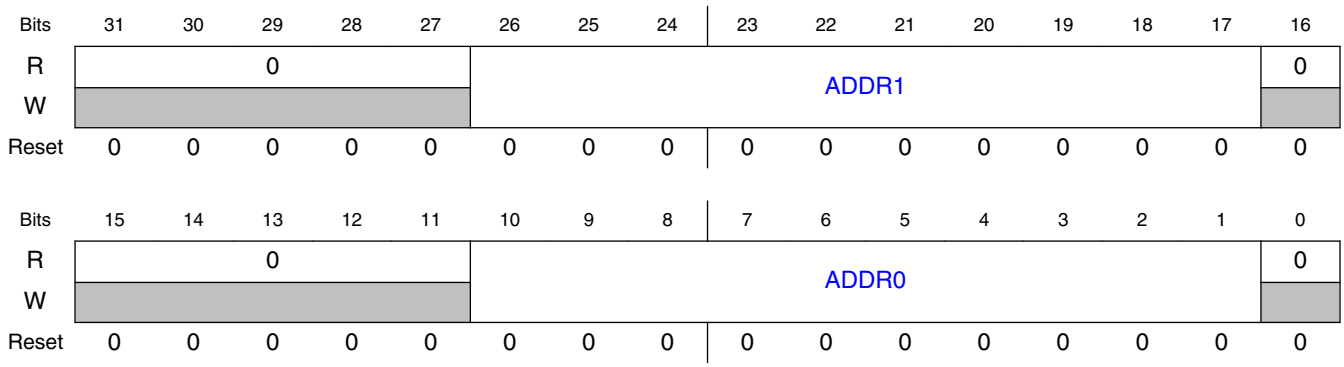
38.4.1.25.1 Offset

Register	Offset
SAMR	140h

38.4.1.25.2 Function

The SAMR should only be written when the I2C Slave is disabled.

38.4.1.25.3 Diagram



38.4.1.25.4 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

38.4.1.26 Slave Address Status Register (SASR)

38.4.1.26.1 Offset

Register	Offset
SASR	150h

38.4.1.26.2 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	ANV	0			RADDR										
W																
Reset	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

38.4.1.26.3 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 store 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]

38.4.1.27 Slave Transmit ACK Register (STAR)

38.4.1.27.1 Offset

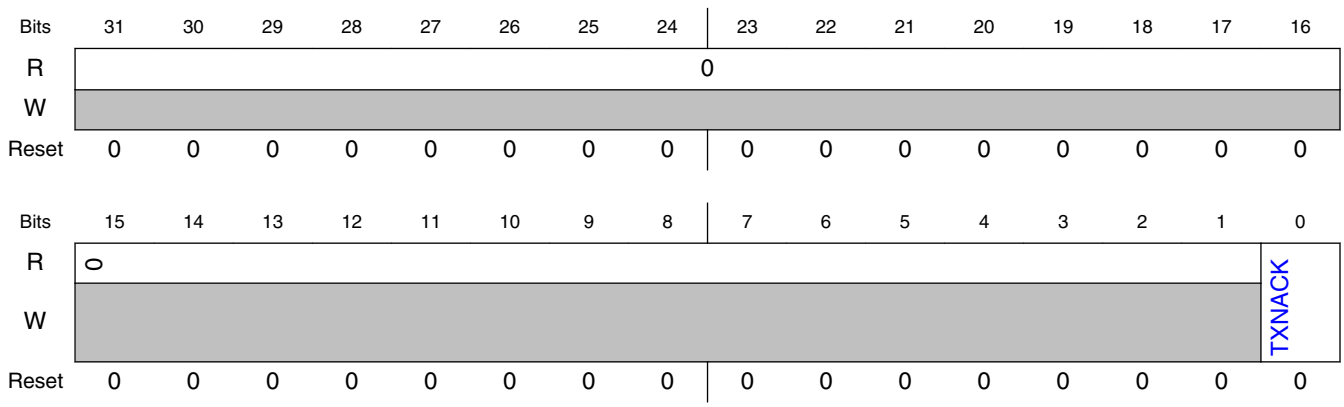
Register	Offset
STAR	154h

38.4.1.27.2 Function

The Slave Transmit ACK Register can only be written when the ACK SCL Stall bit is set in Slave Configuration Register 1 (SCFGR1[ACKSTALL].

- The ACKSTALL bit will enable 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], Slave Configuration Register 2).

38.4.1.27.3 Diagram



38.4.1.27.4 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 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 will stall 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

38.4.1.28 Slave Transmit Data Register (STDR)

38.4.1.28.1 Offset

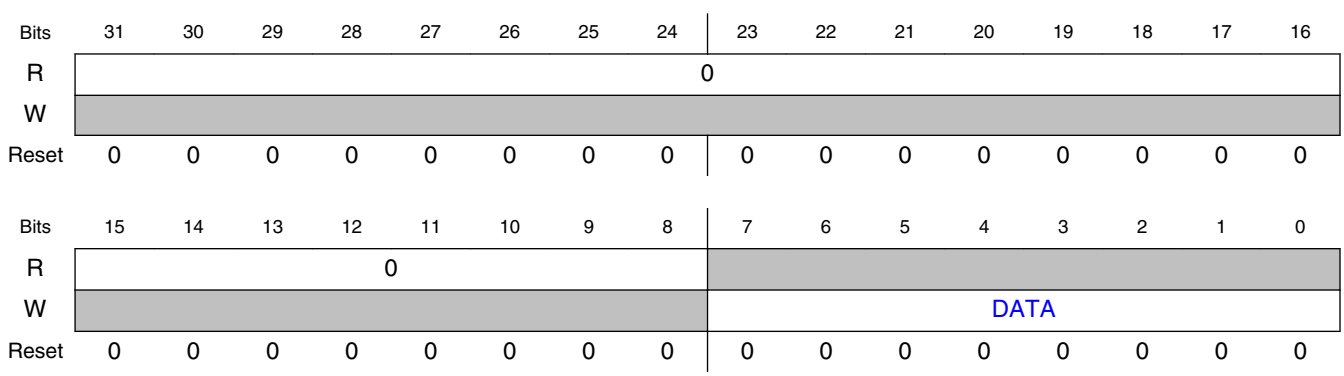
Register	Offset
STDR	160h

38.4.1.28.2 Function

Clock stretching (enabled or disabled) affects when the transmit data is transferred. The TXDSTALL bit will enable clock stretching during the 1st data bit of a slave-transmit transfer.

- **If clock stretching is enabled (TXDSTALL=1)**, then the transmit data transfer is stalled until the Slave Transmit register (STDR) is updated. Clock stretching is extended by at least 1 bus clock cycle after the Slave Transmit Data Register (STDR) is updated, and clock stretching can be delayed even more using the Clock Hold Time field (SCFGR2[CLKHOLD], Slave Configuration Register 2).
- **If clock stretching is disabled (TXDSTALL=0)**, then the transmit data should be written before the start of the slave-transmit transfer; otherwise (i.e., if the transmit data is not written before the start of the slave-transmit transfer), the FIFO Error Flag (SSR[FEF]) will be set.

38.4.1.28.3 Diagram



38.4.1.28.4 Fields

Field	Function
31-8 —	Reserved

Table continues on the next page...

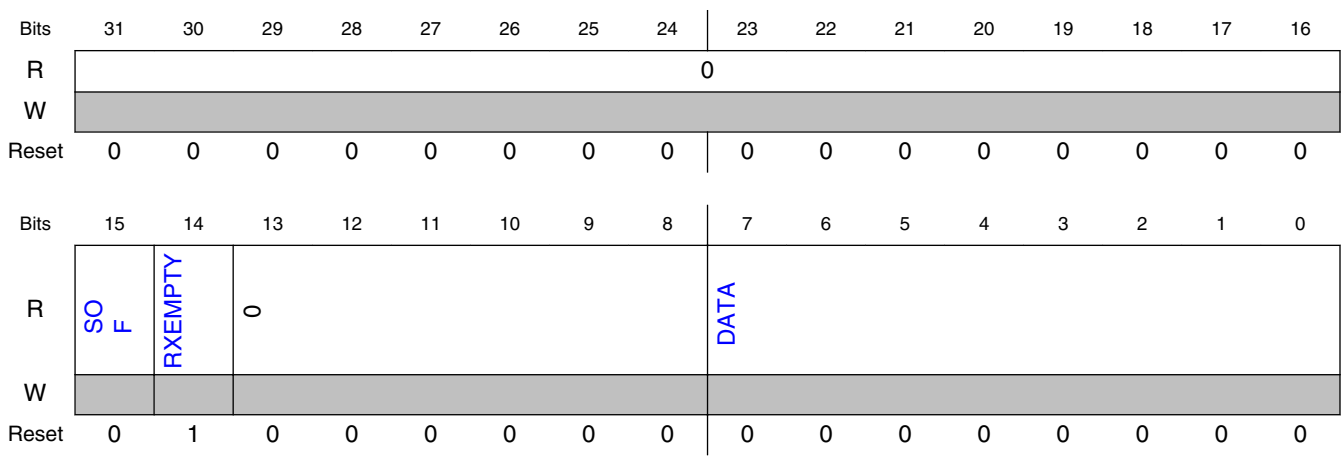
Field	Function
7-0 DATA	Transmit Data Writing to the Slave Transmit Data Register (STDR) will store I2C slave transmit data *in* the Slave Transmit Data Register

38.4.1.29 Slave Receive Data Register (SRDR)

38.4.1.29.1 Offset

Register	Offset
SRDR	170h

38.4.1.29.2 Diagram



38.4.1.29.3 Fields

Field	Function
31-16 —	Reserved
15 SOF	Start Of Frame 0b - Indicates this is not the first data word since a (repeated) START or STOP condition 1b - Indicates this is the first data word since a (repeated) START or STOP condition
14 RXEMPTY	RX Empty 0b - The Receive Data Register is not empty 1b - The Receive Data Register is empty

Table continues on the next page...

Field	Function
13-8 —	Reserved
7-0 DATA	Receive Data Reading the Slave Receive Data Register returns the data received by the I2C slave

Chapter 39
Low Power Serial Peripheral Interface (LPSPI)

39.1 Chip-specific LPSPI information

NOTE

The "Output Triggers" section is not applicable for this device.

Table 39-1. 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

39.2 Introduction

The 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 LPSPI is designed to use little CPU overhead, with DMA offloading of FIFO register accesses.
- The LPSPI can continue operating in stop modes, if an appropriate clock is available.
- The LPSPI supports DMA accesses and generates a DMA request.

The Serial Peripheral Interface bus (SPI) 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.

- SPI devices communicate in full duplex mode using a master-slave scheme with a single master. This enables communication in both directions to happen simultaneously.
- Multiple slave devices are selected using individual slave select (SS) lines.
- The master device originates the frame for reading and writing.
- SPI accesses are always synchronous to the external clock.
- Each SPI can only assert one chip select at a time, but technically there can be multiple SPI slaves sharing the same chip select (in the form of a larger shift register). This requires:
  - the SDO of the master to connect to the SDI of the first slave,
  - the SDO of first slave to connect to the SDI of the next slave,
  - and the SDO of last slave to connect to the SDI of the LPSPI master.

  Note that some SPI modules use MISO/MOSI for the input/output data.

39.2.1 Features

The LPSPI supports:

- Word size = 32 bits
- Configurable clock polarity and clock phase
- Master operation supporting multiple peripheral chip selects (see chip-specific information)
- 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
- Support for full duplex transfers supporting 1-bit transmit and receive on each clock edge
- Support for half duplex transfers supporting 1-bit transmit or receive on each clock edge
- Support for half duplex transfers supporting 2-bit or 4-bit transmit or receive on each clock edge (master only)
- Host request input can be used to control the start time of an SPI bus transfer (master only)
- Receive data match logic supporting wakeup on data match

39.2.2 Block Diagram

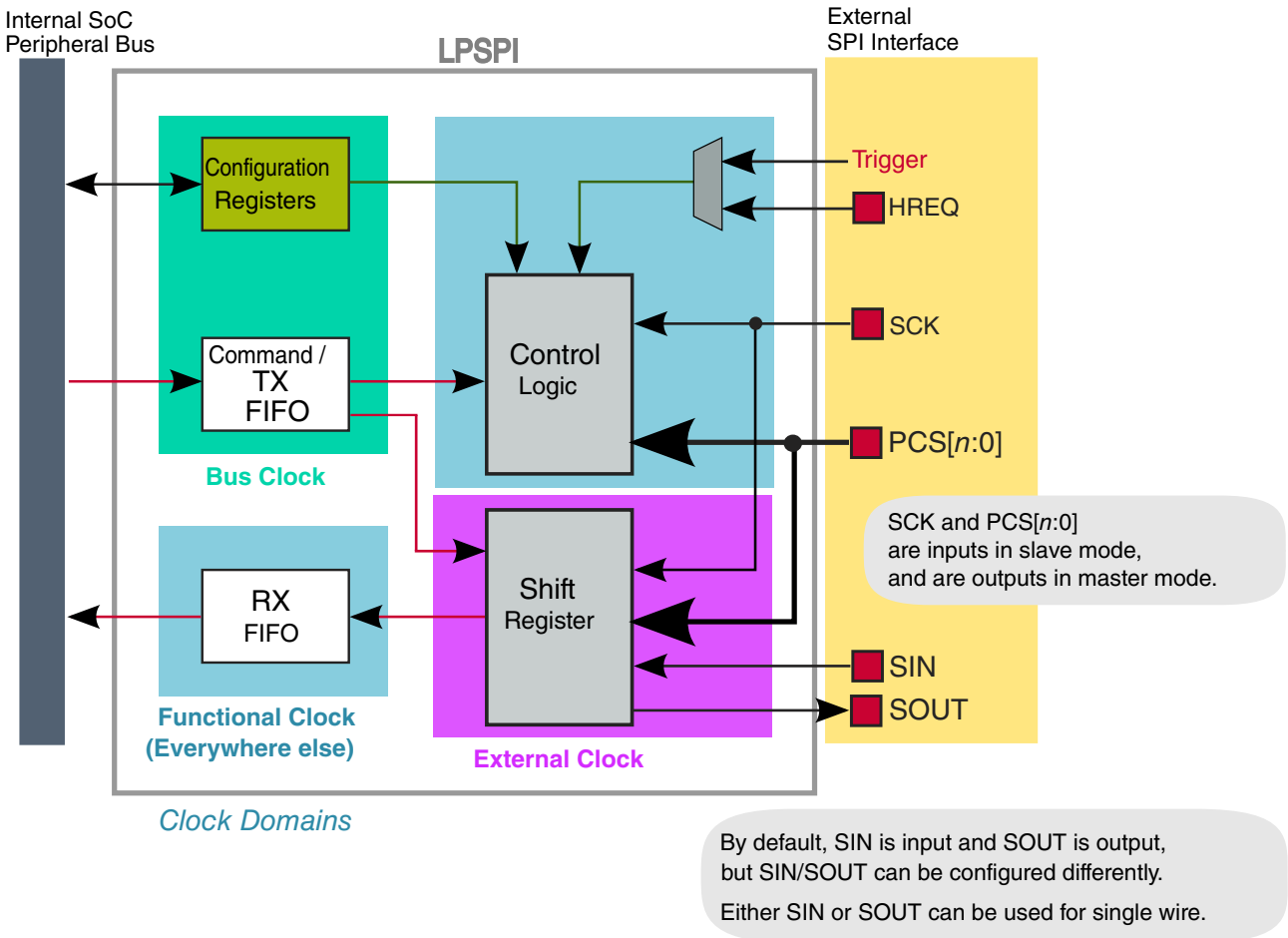


Figure 39-1. Block Diagram

39.2.3 Modes of operation

Table 39-2. Chip modes supported by the LPSPI module

Chip mode	LPSPI Operation
Run	Normal operation
Stop	Can continue operating in stop mode if the Doze Enable bit (CR[DOZEN]) is clear and the LPSPI is using an external or internal clock source that remains operating during stop mode.
Debug (the core is in Debug/Halted mode)	Can continue operating in debug mode, if the Debug Enable bit (CR[DBGEN]) is set.

39.2.4 Signal Descriptions

Table 39-3. 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 HREN=1 and HRSEL=0 • Input in either slave mode or when used as Host Request • Output in master mode	I/O
PCS[2] / DATA[2]	Peripheral Chip Select or data pin 2 during quad-data transfers	• Input in slave mode • Output in master mode • Input in quad-data receive transfers • Output in quad-data transmit transfers	I/O
PCS[3] / DATA[3]	Peripheral Chip Select or data pin 3 during quad-data transfers	• Input in slave mode • Output in master mode • Input in quad-data receive transfers • Output in quad-data transmit transfers	I/O
SOUT / DATA[0]	Serial Data Output	Can be configured as serial data input signal • Used as data pin 0 in quad-data transfers • Used as data pin 0 in dual-data transfers	I/O
SIN / DATA[1]	Serial Data Input	Can be configured as serial data output signal • Used as data pin 1 in quad-data transfers • Used as data pin 1 in dual-data transfers	I/O

39.3 Functional description

39.3.1 Clocking and resets

For device-specific clocking information, refer to the Clocking chapter.

Table 39-4. 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 (LPSPI_SCK).
External clock	• The LPSPI shift register is clocked directly by the LPSPI_SCK clock. • How the LPSPI_SCK clock is generated or supplied depends upon the mode (master or slave):

Table continues on the next page...

Table 39-4. Clocks (continued)

	• in master mode: the LPSPI_SCK clock is generated internally. • in slave mode: the LPSPI_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.

Table 39-5. 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. • Control Register CR[RTF](Reset Transmit FIFO) and CR[RRF] (Reset Receive FIFO) are write-only bits. • After being reset,a FIFO is empty.

NOTE

The entire PCSPOL field is not supported in every LPSPI module instance. Refer to the chip-specific information for LPSPI.

Field supported in	Field not supported in
LPSPI0_CFGR1	—
LPSPI1_CFGR1[13–8]	LPSPI1_CFGR1[15–14]

39.3.2 Master Mode

39.3.2.1 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 Register (TDR).
- Command words are stored to the transmit/command FIFO, by writing the Transmit Command Register (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, the Continuous Transfer bit (TCR[CONT]), and the Continuing Command bit (TCR[CONTC]).

Table 39-6. Possible actions when a command word is at the top of the transmit/command FIFO

Condition	Action
If the 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 Continuous Transfer bit (TCR[CONT]) is set or cleared and the Continuing Command bit (TCR[CONTC]) is cleared	then the SPI frame will complete 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 CONTC=0 will always terminate (stop) the existing transfer regardless of the previous CONT value.
If the LPSPI is busy and the existing Continuous Transfer bit (TCR[CONT]) is set or cleared and the new Continuing Command bit (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 LPSPI_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). When the Continuing Command bit (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.

NOTE

About the Continuous Transfer bit (CONT) and Continuing Command bit (CONTC):

- Continuous Transfer bit (CONT)=1 is used to keep PCS asserted at end of frame, allowing the transfer to continue.
- Continuing Command bit (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.

The Continuing Command bit (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 the Continuous Transfer bit (CONT)=1.

The current state of the existing command word can be read by reading the Transmit Command Register (TCR). It requires at least 3 LPSPI functional clock cycles for the transmit command register to update after the transmit command register is written (assuming an empty FIFO) and the LPSPI must be enabled (Module Enable CR[MEN] bit is set).

Writing the transmit command register does not initiate a SPI bus transfer, unless the Transmit Data Mask (TCR[TXMSK]) bit is set. When the Transmit Data Mask bit is set, a new command word will not be loaded until the end of the existing frame (based on FRAMESZ configuration); at the end of the transfer, the TXMSK bit will be cleared.

In master mode, the LPSPI command word in the Transmit Command Register (TCR) controls SPI attributes (using bits and fields in registers).

Table 39-7. LPSPI Command Word in Master Mode

Transmit Command Register (TCR)		Description	Can this bit/ field be modified during a data transfer?
Bit/Field	Name		
CPOL	Clock Polarity	Configures the polarity of the LPSPI_SCK pin. Any change of CPOL value will cause a transition on the LPSPI_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
PCS	Peripheral Chip Select	Configures which LPSPI_PCS asserts for the transfer; the polarity of LPSPI_PCS is static and configured by PCSPOL. If PCSCFG is set, 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 bounday.	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 Output Config CFGR1[OUTCFG]). Useful for half-duplex transfers.	Y
WIDTH	Transfer Width	Configures the number of bits shifted on each LPSPI_SCK pulse. • 1-bit transfers support traditional SPI bus transfers in either half-duplex or full-duplex data formats. • 2-bit and 4-bit transfers are useful for interfacing to QuadSPI memory devices, and only support half-duplex data formats, and at least one bit (Transmit Data Mask (TCR[TXMSK] or Receive Data Mask TCR[RXMSK]) must also be set.	Y

Table continues on the next page...

Table 39-7. LPSPI Command Word in Master Mode (continued)

Transmit Command Register (TCR)		Description	Can this bit/ field be modified during a data transfer?
Bit/Field	Name		
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:

- The LPSPI initiates a SPI bus transfer when:
  - data is written to the transmit FIFO
  - and the HREQ pin is asserted (or disabled)
  - and the LPSPI is enabled
- To perform the SPI bus transfer, LPSPI uses the attributes configured in the Transmit Command Register (TCR) and uses the timing parameters in the Clock Configuration Register (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 the LPSPI goes idle (the LPSPI completes the current transfer and the Transmit Command Register (TCR) is empty).

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 can fit within the transmit/command FIFO, without requiring additional FIFO accesses. When the circular FIFO is enabled, 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 the 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.

39.3.2.2 Receive FIFO and Data Match

The receive FIFO is used to store receive data during SPI bus transfers. When the Receive Data Mask TCR[RXMSK] bit is set, 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 the Transmit Command Register (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 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 Receive Data Mask TCR[RXMSK] bit cannot cause the data match to set, and will delay the receive data match on the first received data word, until all discarded data is received.
- The receiver data match function can also be configured to discard all received data until a data match is detected, using the Receive Data Match Only CFGR0[RDMO] bit.
- After a receive data match, to allow all subsequent data to be received, first clear the Receive Data Match Only CFGR0[RDMO] bit, then clear the Data Match Flag SR[DMF] bit.

39.3.2.3 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 Register (CCR) cannot be changed when the LPSPI module is busy, to support interfacing to different slave devices at different frequencies, the Prescaler Value TCR[PRESCALE] configuration can be changed between SPI bus transfers using the Transmit Command Register (TCR).

Table 39-8. LPSPI Timing Parameters

Clock Configuration Register (CCR)		Description	Min	Max
Bit/Field	Name			
SCKDIV	SCK Divider	Configures the LPSPI_SCK clock period to (SCKDIV +2) cycles. When SCK Divider is configured to an odd number of cycles, the first half of the LPSPI_SCK cycle is one cycle longer than the second half of the LPSPI_SCK cycle.	0 (2 cycles)	255 (257 cycles)
DBT	Delay Between Transfers	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 LPSPI_PCS pins.	0 (2 cycles)	255 (257 cycles)
DBT	Delay Between Transfers	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)

39.3.2.4 Pin Configuration

- To swap directions or support half-duplex transfers on the same pin, the LPSPI_SIN and LPSPI_SOUT pins can be configured using the Pin Configuration CFGR1[PINCFG].
- To determine if an output data pin (like LPSPI_SOUT) will tristate when the LPSPI_PCS is negated, or if the output data pin will simply retain the last value, use the Output Config CFGR1[OUTCFG].
- When configuring for half-duplex transfers using the same data pin in single-bit transfer mode, or when configuring any transfer in 2-bit and 4-bit transfer modes, the output data pins must be configured to tristate when LPSPI_PCS is negated; use the Output Config CFGR1[OUTCFG].
- When performing quad-data transfers, the Peripheral Chip Select Configuration CFGR1[PCSCFG] must be enabled. The Peripheral Chip Select Configuration CFGR1[PCSCFG] is also used to disable LPSPI_PCS[3:2] functions.

39.3.2.5 Clock Loopback

The LPSPI master can be configured to use 1 of 2 clocks to sample the input data (for example, LPSPI_SIN):

- either the LPSPI_SCK output clock directly
- or a delayed version of the LPSPI_SCK output clock

The delayed version of the LPSPI_SCK is delayed by the LPSPI_SCK pin output delay, plus the LPSPI_SCK pin input delay, and is configured by setting CFGR1[SAMPLE]. Enabling the loopback version of the LPSPI_SCK can improve the setup time of the input data from the slave.

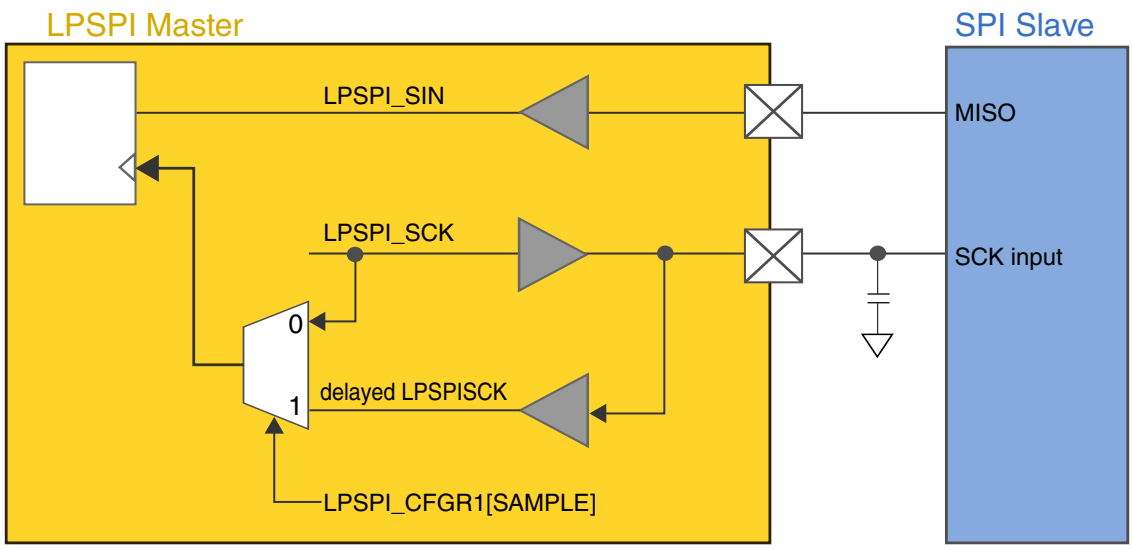


Figure 39-2. Clock Loopback

See the device datasheet for the specific input setup time in master loopback mode.

39.3.3 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 Register (TCR) remain static (unchanging)

39.3.3.1 Transmit and Command FIFO commands

Before enabling the LPSPI in slave mode, the Transmit Command Register (TCR) should be initialized, although the Transmit Command Register register will not update until after the LPSPI is enabled. After being enabled, the Transmit Command Register should only be changed if the LPSPI is idle. In slave mode, the LPSPI command word in the Transmit Command Register (TCR) controls SPI attributes (using bits and fields in registers). Before the LPSPI_PCS input asserts, the transmit FIFO must be filled with transmit data, or the transmit error flag will set.

Table 39-9. LPSPI Command Word in Slave Mode

Transmit Command Register (TCR)		Description
Bit/Field	Name	
CPOL	Clock Polarity	Configures the polarity of the external LPSPI_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 LPSPI_PCS is used, the polarity of LPSPI_PCS is static and configured by PCSPOL. If PCSCFG is set, then PCS[3:2] 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, only the first FRAMSZ bits will be transmitted/ received by the LPSPI module.
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 Output Config CFGR1[OUTCFG]). Useful for half-duplex transfers.
WIDTH	Transfer Width	Configures the number of bits shifted on each LPSPI_SCK pulse. • 1-bit transfers support traditional SPI bus transfers in either half-duplex or full-duplex data formats. • 2-bit and 4-bit transfers are useful for interfacing to QuadSPI memory devices, and only support half-duplex data formats, and at least one bit (Transmit Data Mask TCR[TXMSK] or Receive Data Mask (TCR[RXMSK]) must also be set.
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.

39.3.3.2 Receive FIFO and Data Match

The receive FIFO is used to store received data during SPI bus transfers. When the Receive Data Mask TCR[RXMSK] is set, 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 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 the Receive Data Mask TCR[RXMSK] is set) cannot cause the data match to set, and will delay 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 Receive Data Match Only CFGR0[RDMO] bit.
- After a receive data match, to allow all subsequent data to be received, first clear the Receive Data Match Only CFGR0[RDMO] bit, then clear the Data Match Flag SR[DMF] bit.

39.3.3.3 Clocked Interface

The LPSPI module supports interfacing to external masters that provide only clock and data pins (LPSPI_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 LPSPI_PCS input to be always asserted (configure the Peripheral Chip Select Polarity CFGR1[PCSPOL[n]] = 1). For example, to configure PCS[0] to be always asserted, set PCSPOL[0] = 1, and don’t configure PCS[0] in the pin muxing.
- setting the Automatic PCS CFGR1[AUTOPCS] bit = 1 (Automatic PCS generation is enabled). When Automatic PCS generation (AUTOPCS) is set, a minimum of 4 LPSPI functional clock cycles (divided by PRESCALE configuration) is required between the last LPSPI_SCK edge of one word and the first LPSPI_SCK edge of the next word.

39.3.4 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 39-10. LPSPI Interrupts and DMA Requests

Status Register (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 the No Stall CFGR1[NOSTALL] bit is clear (0, transfers will stall when transmit FIFO is empty or receive FIFO is full), the Transmit Error Flag bit cannot set.	Y	N	Y
REF	Receive Error Flag	Receive error flag, indicates a receive FIFO overflow. In master mode when the No Stall CFGR1[NOSTALL] bit is clear (0, transfers will stall when transmit FIFO is empty or receive FIFO is full), the Receive Error Flag bit cannot set.	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

39.3.5 Peripheral Triggers

The connection of the LPSPI peripheral triggers to other peripherals depend upon the specific device being used.

Table 39-11. LPSPI Triggers

Trigger	Description	
Frame Output Trigger	The frame output trigger: • asserts at the end of each frame (when PCS negates) • remains asserted until PCS next asserts	The LPSPI generates 2 output triggers that can be connected to other peripherals on the device.
Word Output Trigger	The word output trigger: • asserts at the end of each received word • remains asserted for 1 LPSPI_SCK period	
Input Trigger	To control the start of a LPSPI bus transfer, the LPSPI input trigger can be selected instead of the LPSPI_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 LPSPI_HREQ input (and therefore the LPSPI input trigger) is ignored .	

39.4 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.

39.4.1 LPSPI register descriptions

39.4.1.1 LPSPI memory map

LPSPI1 base address: 4019_4000h

LPSPI2 base address: 4019_8000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Version ID Register (VERID)	32	RO	0102_0004h
4h	Parameter Register (PARAM)	32	RO	0004_0404h
10h	Control Register (CR)	32	RW	0000_0000h
14h	Status Register (SR)	32	W1C	0000_0001h
18h	Interrupt Enable Register (IER)	32	RW	0000_0000h
1Ch	DMA Enable Register (DER)	32	RW	0000_0000h
20h	Configuration Register 0 (CFGR0)	32	RW	0000_0000h
24h	Configuration Register 1 (CFGR1)	32	RW	0000_0000h
30h	Data Match Register 0 (DMR0)	32	RW	0000_0000h
34h	Data Match Register 1 (DMR1)	32	RW	0000_0000h
40h	Clock Configuration Register (CCR)	32	RW	0000_0000h
58h	FIFO Control Register (FCR)	32	RW	0000_0000h
5Ch	FIFO Status Register (FSR)	32	RO	0000_0000h
60h	Transmit Command Register (TCR)	32	RW	0000_001Fh
64h	Transmit Data Register (TDR)	32	WO	0000_0000h
70h	Receive Status Register (RSR)	32	RO	0000_0002h
74h	Receive Data Register (RDR)	32	RO	0000_0000h

39.4.1.2 Version ID Register (VERID)

39.4.1.2.1 Offset

Register	Offset
VERID	0h

39.4.1.2.2 Function

Contains version numbers for the module design and feature set.

39.4.1.2.3 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

39.4.1.2.4 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.

39.4.1.3 Parameter Register (PARAM)

39.4.1.3.1 Offset

Register	Offset
PARAM	4h

39.4.1.3.2 Function

Contains parameter values that were implemented in the module.

39.4.1.3.3 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

39.4.1.3.4 Fields

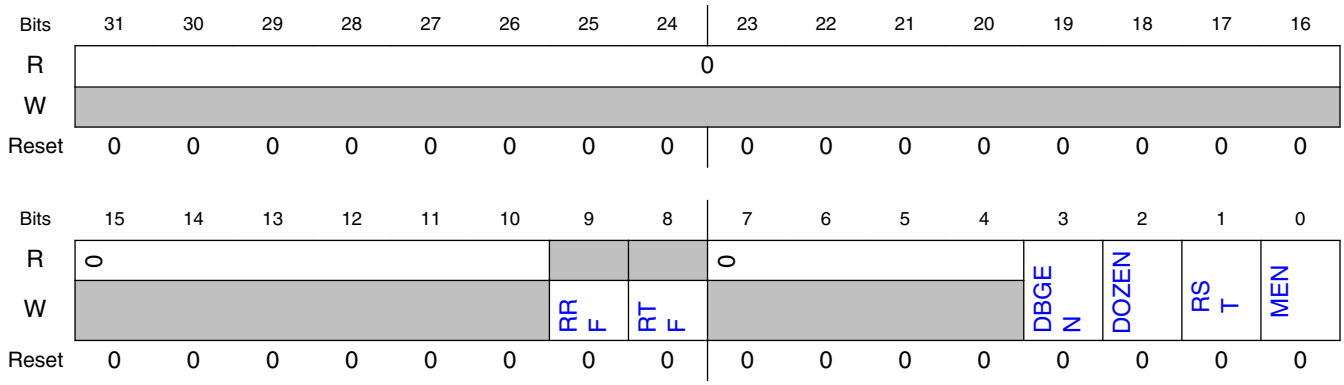
Field	Function
31-24 —	Reserved
23-16 PCSNUM	PCS Number Sets the number of PCS pins supported by the peripheral.
15-8 RXFIFO	Receive FIFO Size Sets the maximum number of words in the receive FIFO, which is 2^{RXFIFO}
7-0 TXFIFO	Transmit FIFO Size Sets the maximum number of words in the transmit FIFO, which is 2^{TXFIFO}

39.4.1.4 Control Register (CR)

39.4.1.4.1 Offset

Register	Offset
CR	10h

39.4.1.4.2 Diagram



39.4.1.4.3 Fields

Field	Function
31-10 —	Reserved
9 RRF	Reset Receive FIFO 0b - No effect 1b - Receive FIFO is reset
8 RTF	Reset Transmit FIFO 0b - No effect 1b - Transmit FIFO is reset
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. 0b - Module is not reset 1b - Module is reset
0 MEN	Module Enable 0b - Module is disabled 1b - Module is enabled

39.4.1.5 Status Register (SR)

39.4.1.5.1 Offset

Register	Offset
SR	14h

39.4.1.5.2 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

39.4.1.5.3 Fields

Field	Function
31-25 —	Reserved
24 MBF	Module Busy Flag 0b - LPSPI is idle 1b - LPSPI is busy
23-14 —	Reserved
13 DMF	Data Match Flag Indicates that the received data has matched the MATCH0 and/or MATCH1 fields (as configured by CFGR1[MATCFG], Configuration Register 1). 0b - Have not received matching data 1b - Have received matching data
12 REF	Receive Error Flag The Receive Error Flag will set 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

Table continues on the next page...

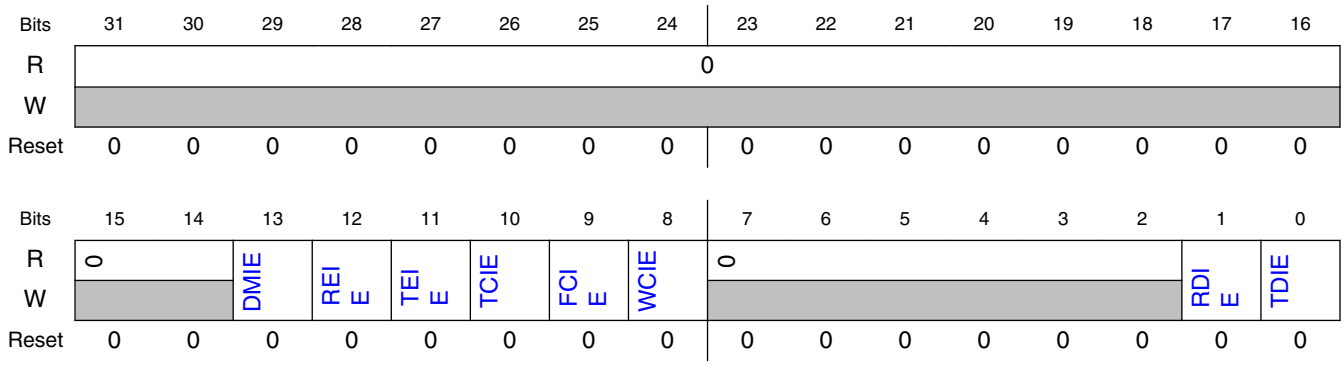
Field	Function
	1b - Receive FIFO has overflowed
11 TEF	Transmit Error Flag The Transmit Error Flag will set when the Transmit FIFO underruns. When the Transmit Error Flag is set, it is recommended to first end the transfer, clear the Transmit Error Flag and then restart the transfer from the beginning. 0b - Transmit FIFO underrun has not occurred 1b - Transmit FIFO underrun has occurred
10 TCF	Transfer Complete Flag In master mode when the LPSPI returns to idle state with the transmit FIFO empty, the Transfer Complete Flag will set. 0b - All transfers have not completed 1b - All transfers have completed
9 FCF	Frame Complete Flag The Frame Complete Flag will set 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 will set 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 FCR[RXWATER] (FIFO Control Register) 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 FCR[TXWATER] (FIFO Control Register) 0b - Transmit data not requested 1b - Transmit data is requested

39.4.1.6 Interrupt Enable Register (IER)

39.4.1.6.1 Offset

Register	Offset
IER	18h

39.4.1.6.2 Diagram



39.4.1.6.3 Fields

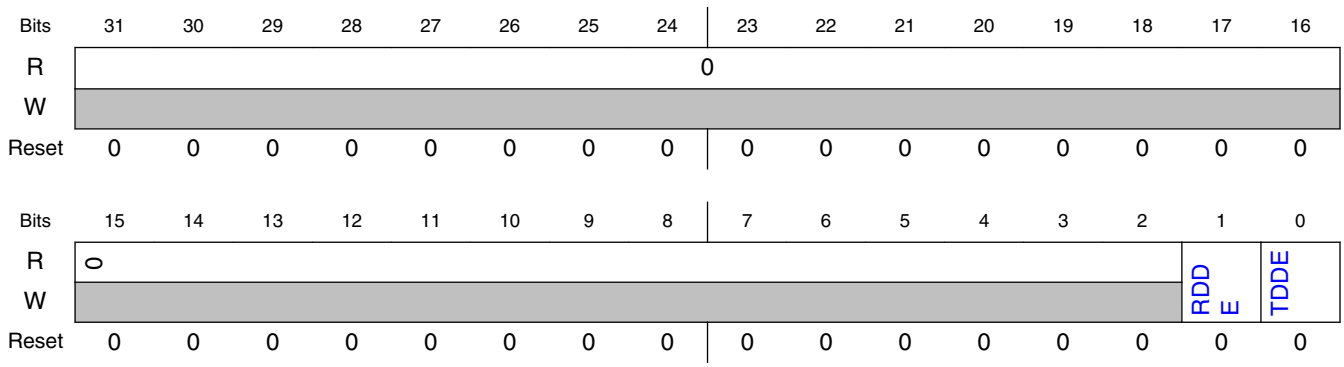
Field	Function
31-14 —	Reserved
13 DMIE	Data Match Interrupt Enable 0b - Disabled 1b - Enabled
12 REIE	Receive Error Interrupt Enable 0b - Disabled 1b - Enabled
11 TEIE	Transmit Error Interrupt Enable 0b - Disabled 1b - Enabled
10 TCIE	Transfer Complete Interrupt Enable 0b - Disabled 1b - Enabled
9 FCIE	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

39.4.1.7 DMA Enable Register (DER)

39.4.1.7.1 Offset

Register	Offset
DER	1Ch

39.4.1.7.2 Diagram



39.4.1.7.3 Fields

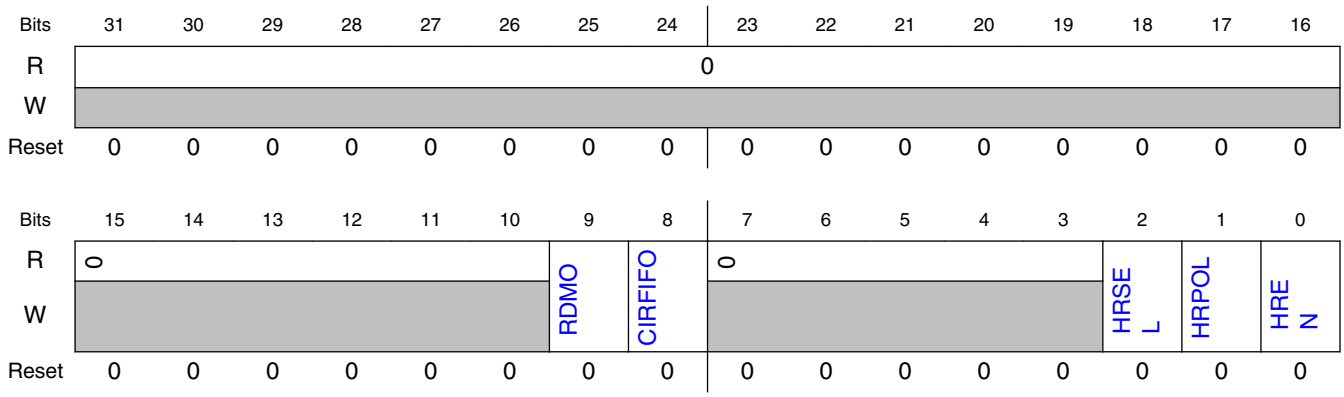
Field	Function
31-2 —	Reserved
1 RDDE	Receive Data DMA Enable 0b - DMA request is disabled 1b - DMA request is enabled
0 TDDE	Transmit Data DMA Enable 0b - DMA request is disabled 1b - DMA request is enabled

39.4.1.8 Configuration Register 0 (CFGR0)

39.4.1.8.1 Offset

Register	Offset
CFGR0	20h

39.4.1.8.2 Diagram



39.4.1.8.3 Fields

Field	Function
31-10 —	Reserved
9 RDMO	Receive Data Match Only When Receive Data Match Only is enabled, all received data that does not cause the Data Match Flag (SR[DMF]) to set is discarded. • Receive Data Match Only bit should be set when the LPSPI is idle and the Data Match Flag is clear • After the Data Match Flag (DMF) is set, the Receive Data Match Only bit configuration is ignored • When disabling RDMO and to ensure that no receive data is lost, before clearing the Data Match Flag, first clear Receive Data Match Only (RDMO) 0b - Received data is stored in the receive FIFO as in normal operations 1b - Received data is discarded unless the Data Match Flag (DMF) is set
8 CIRFIFO	Circular FIFO Enable When enabled, the transmit FIFO read pointer is saved to a temporary register. The transmit FIFO will be emptied as it normally is, but after the LPSPI is idle and the transmit FIFO is empty, then the read pointer value will be restored from the temporary register. This restoring of the read pointer will cause the contents of the transmit FIFO to be cycled through repeatedly. 0b - Circular FIFO is disabled 1b - Circular FIFO is enabled
7-3 —	Reserved
2 HRSEL	Host Request Select Selects the source of the host request input. When the host request function is enabled with the LPSPI_HREQ pin, the LPSPI_PCS[1] function is disabled. 0b - Host request input is the LPSPI_HREQ pin 1b - Host request input is the input trigger
1 HRPOL	Host Request Polarity Configures the polarity of the host request pin. 0b - Active low

Table continues on the next page...

Field	Function
	1b - Active high
0 HREN	Host Request Enable When enabled in master mode, the LPSPI will only start a new SPI bus transfer if the host request input is asserted. When the LPSPI is busy, the host request input is ignored. 0b - Host request is disabled 1b - Host request is enabled

39.4.1.9 Configuration Register 1 (CFGR1)

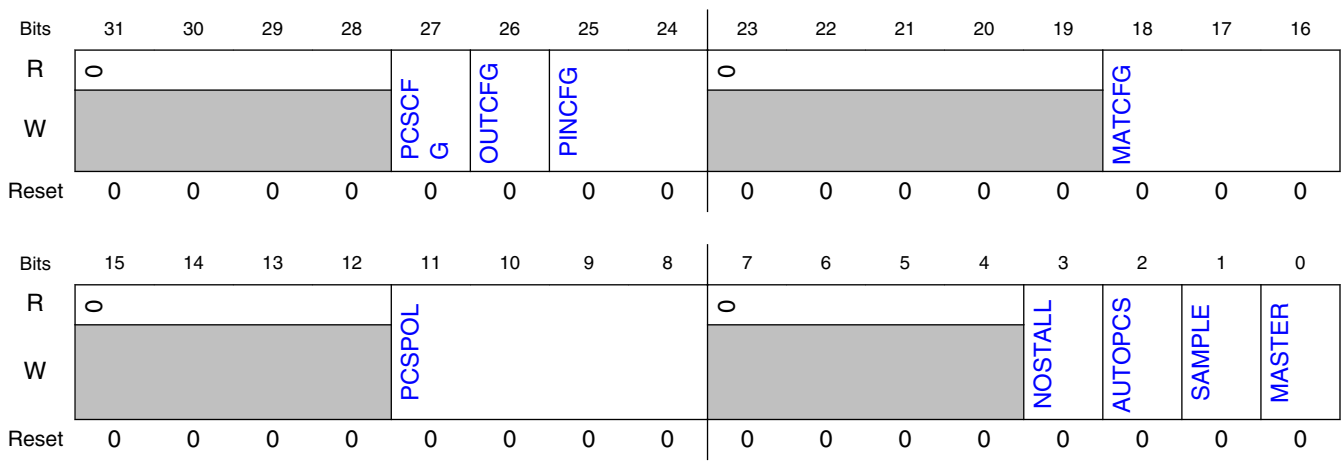
39.4.1.9.1 Offset

Register	Offset
CFGR1	24h

39.4.1.9.2 Function

The CFGR1 should only be written when the LPSPI is disabled.

39.4.1.9.3 Diagram



39.4.1.9.4 Fields

Field	Function
31-28	Reserved

Table continues on the next page...

Field	Function
—	
27 PCSCFG	Peripheral Chip Select Configuration If performing 4-bit transfers, the Peripheral Chip Select Configuration bit must be set. 0b - PCS[3:2] are enabled 1b - PCS[3:2] are disabled
26 OUTCFG	Output Configuration Configures if the output data is tristated between accesses (LPSPI_PCS is negated). If performing 2-bit or 4-bit 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 1-bit transfers. If performing 2-bit or 4-bit 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 10b - SOUT is used for both input and output data 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 will cause the DMF to set. **NOTE:** *Syntax:* * is boolean AND, + is boolean OR 000b - Match is disabled 001b - Reserved 010b - 010b - Match is enabled, if 1st data word equals MATCH0 OR MATCH1, i.e., (1st data word = MATCH0 + MATCH1) 011b - 011b - Match is enabled, if any data word equals MATCH0 OR MATCH1, i.e., (any data word = MATCH0 + MATCH1) 100b - 100b - Match is enabled, if 1st data word equals MATCH0 AND 2nd data word equals MATCH1, i.e., [(1st data word = MATCH0) * (2nd data word = MATCH1)] 101b - 101b - Match is enabled, if any data word equals MATCH0 AND the next data word equals MATCH1, i.e., [(any data word = MATCH0) * (next data word = MATCH1)] 110b - 110b - Match is enabled, if (1st data word AND MATCH1) equals (MATCH0 AND MATCH1), i.e., [(1st data word * MATCH1) = (MATCH0 * MATCH1)] 111b - 111b - Match is enabled, if (any data word AND MATCH1) equals (MATCH0 AND MATCH1), i.e., [(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.
7-4 —	Reserved

Table continues on the next page...

Field	Function
3 NOSTALL	No Stall In master mode, the LPSPI will stall transfers when the transmit FIFO is empty or when the receive FIFO is full, ensuring that no transmit FIFO underrun or receive FIFO overrun can occur. Setting the No Stall bit will disable this functionality. 0b - Transfers will stall when the transmit FIFO is empty or the receive FIFO is full 1b - Transfers will not stall, allowing transmit FIFO underruns or receive FIFO overruns to occur
2 AUTOPCS	Automatic PCS For correct operations, the LPSPI slave normally requires the PCS to negate between frames. Setting the Automatic PCS bit will cause the 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 will sample the input data on a delayed LPSPI_SCK edge, which improves the setup time when sampling data. • The input data setup time in master mode with delayed LPSPI_SCK edge is equal to the input data setup time in slave mode • In slave mode, the SAMPLE bit is ignored 0b - Input data is sampled on SCK edge 1b - Input data is sampled on delayed SCK edge
0 MASTER	Master Mode Configures the LPSPI in master or slave mode. The Master Mode bit directly controls the direction of the LPSPI_SCK and LPCPI_PCS pins. 0b - Slave mode 1b - Master mode

39.4.1.10 Data Match Register 0 (DMR0)

39.4.1.10.1 Offset

Register	Offset
DMR0	30h

39.4.1.10.2 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

39.4.1.10.3 Fields

Field	Function
31-0 MATCH0	Match 0 Value When Receive Data Match Only (CFGR0[RDMO]) is enabled, the Match 0 Value is compared against the received data.

39.4.1.11 Data Match Register 1 (DMR1)

39.4.1.11.1 Offset

Register	Offset
DMR1	34h

39.4.1.11.2 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

39.4.1.11.3 Fields

Field	Function
31-0 MATCH1	Match 1 Value When Receive Data Match Only (CFGR0[RDMO]) is enabled, the Match 1 Value is compared against the received data.

39.4.1.12 Clock Configuration Register (CCR)

39.4.1.12.1 Offset

Register	Offset
CCR	40h

39.4.1.12.2 Function

The Clock Configuration Register is only used in master mode, and the Clock Configuration Register cannot be changed when the LPSPI is enabled.

39.4.1.12.3 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

39.4.1.12.4 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.

Table continues on the next page...

Field	Function
	• 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.
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.

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.
	  • 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.

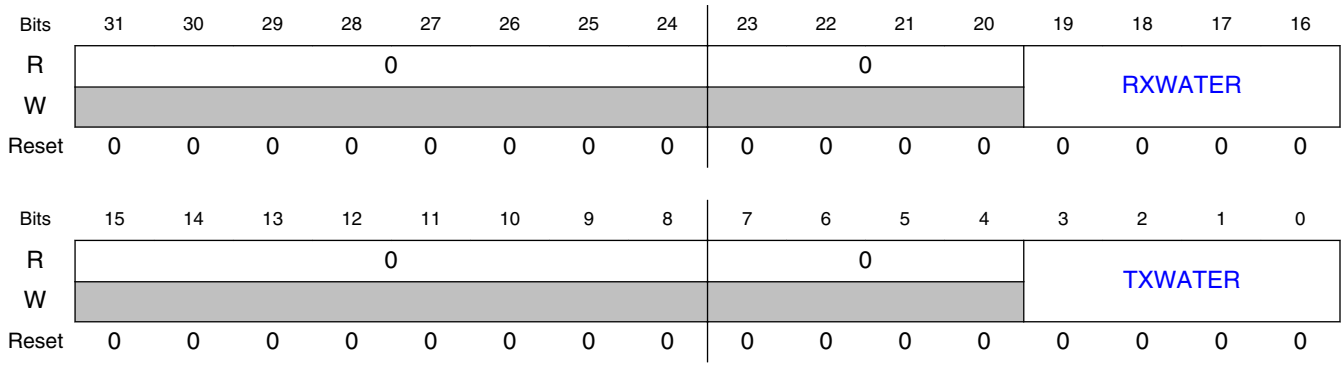
7-0 SCKDIV	SCK Divider In master mode, the SCK Divider configures the divide ratio of the SCK pin. • The SCK period is equal to (SCKDIV+2) cycles of the LPSPI functional clock divided by the Prescaler Value TCR[PRESCALE] configuration. • The minimum SCK period is 2 cycles. • If the SCK period is an odd number of cycles, then the 1st half of the SCK period will be 1 cycle longer than the 2nd half of the SCK period.

39.4.1.13 FIFO Control Register (FCR)

39.4.1.13.1 Offset

Register	Offset
FCR	58h

39.4.1.13.2 Diagram



39.4.1.13.3 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 will be truncated.
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 will be truncated.

39.4.1.14 FIFO Status Register (FSR)

39.4.1.14.1 Offset

Register	Offset
FSR	5Ch

39.4.1.14.2 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

39.4.1.14.3 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.

39.4.1.15 Transmit Command Register (TCR)

39.4.1.15.1 Offset

Register	Offset
TCR	60h

39.4.1.15.2 Function

Writes to either the Transmit Command Register or Transmit Data Register will push 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 will be 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. Changing the command word will cause 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 will clear TXMSK when the LPSPI_PCS negates.
- **In master mode,** if the command word is changed before an existing frame has completed, then the existing frame will terminate and the command word will then update. 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 the LPSPI is idle and there is no SPI bus transfer.

Avoid register reading problems: Reading the Transmit Command Register will return 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:

- to either read the Transmit Command Register when the transmit FIFO is empty,
- or to read the Transmit Command Register more than once and then compare the returned values.

39.4.1.15.3 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 W	Reserved				FRAMESZ											
Reset	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

39.4.1.15.4 Fields

Field	Function
31 CPOL	Clock Polarity The Clock Polarity field is only updated between frames. 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 between frames. 0b - Data is captured on the leading edge of SCK and changed on the following edge of SCK 1b - 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 between frames. 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 between frames. **NOTE:** The entire PCS field is not fully supported in every LPSPI module instance. Refer to the chip-specific information for LPSPI. 00b - Transfer using LPSPI_PCS[0] 01b - Transfer using LPSPI_PCS[1] 10b - Transfer using LPSPI_PCS[2] 11b - Transfer using LPSPI_PCS[3]
23 LSBF	LSB First 0b - Data is transferred MSB first 1b - Data is transferred LSB first
22 BYSW	Byte Swap Byte swap will swap 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 CONT	Continuous Transfer • In master mode, continuous transfer will keep 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 continuous transfer is enabled, the LPSPI will only transmit the first FRAMESZ bits; after which the LPSPI will transmit received data (assuming a 32-bit shift register). 0b - Continuous transfer is disabled 1b - Continuous transfer is enabled

Table continues on the next page...

Field	Function
20 CONTC	Continuing Command In master mode, the Continuing Command 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 Transmit Data Mask bit will initiate a new transfer which cannot be aborted by another command word; the Transmit Data Mask bit will be cleared by hardware at the end of the transfer. 0b - Normal transfer 1b - Mask transmit data
17-16 WIDTH	Transfer Width For 2-bit or 4-bit 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 Software should only write zero to this field.
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 (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 bits, and the 3rd word is 8 bits.

39.4.1.16 Transmit Data Register (TDR)

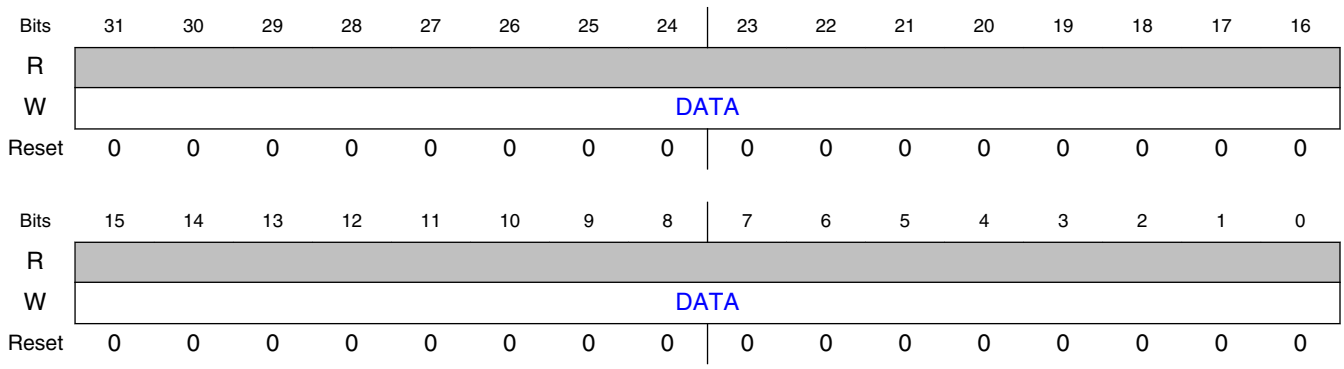
39.4.1.16.1 Offset

Register	Offset
TDR	64h

39.4.1.16.2 Function

Writes to either the Transmit Command Register or Transmit Data Register will push the data into the transmit FIFO, in the order that it (the data) was written. The Data Register can be written using 32-bit, 16-bit or 8-bit writes, but each write will push data into the FIFO with zero pushed in unwritten bytes.

39.4.1.16.3 Diagram



39.4.1.16.4 Fields

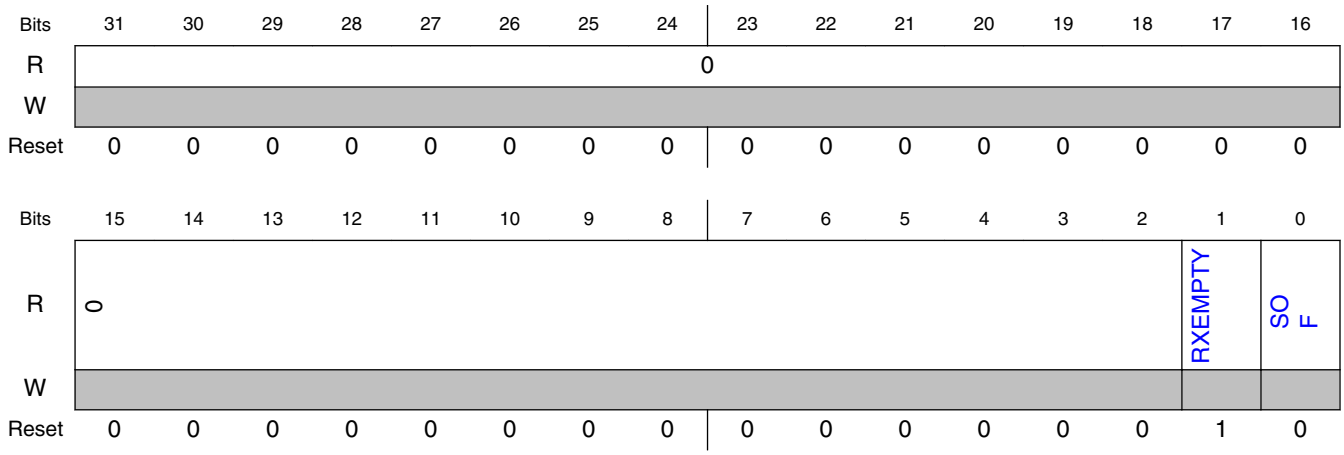
Field	Function
31-0 DATA	Transmit Data Both 8-bit and 16-bit writes of transmit data will zero-extend the data written and push the data into the transmit FIFO. To zero-extend 8-bit and 16-bit writes (to 32 bits), means that the higher order (most significant) empty parts of the 8-bit and 16-bit writes are filled with zeroes.

39.4.1.17 Receive Status Register (RSR)

39.4.1.17.1 Offset

Register	Offset
RSR	70h

39.4.1.17.2 Diagram



39.4.1.17.3 Fields

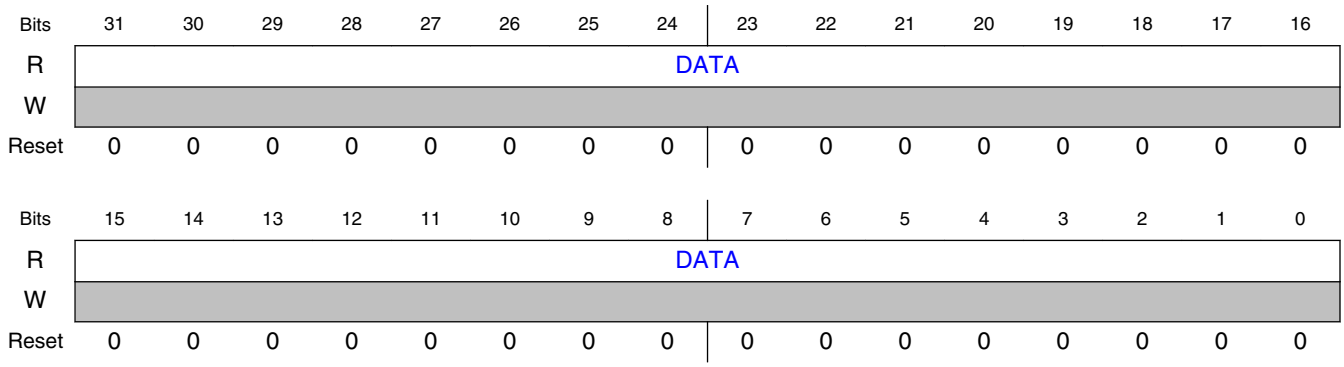
Field	Function
31-2 —	Reserved
1 RXEMPTY	RX FIFO Empty 0b - RX FIFO is not empty 1b - RX FIFO is empty
0 SOF	Start Of Frame Indicates that this is the first data word received after LPSPI_PCS assertion. 0b - Subsequent data word received after LPSPI_PCS assertion 1b - First data word received after LPSPI_PCS assertion

39.4.1.18 Receive Data Register (RDR)

39.4.1.18.1 Offset

Register	Offset
RDR	74h

39.4.1.18.2 Diagram



39.4.1.18.3 Fields

Field	Function
31-0 DATA	Receive Data

Chapter 40
Low Power Universal Asynchronous Receiver/ Transmitter (LPUART)

40.1 Chip-specific LPUART information

Table 40-1. 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

40.2 Introduction

40.2.1 Features

Features of the LPUART module include:

- Full-duplex, standard non-return-to-zero (NRZ) format
- Programmable baud rates (13-bit modulo divider) with configurable oversampling ratio from 4x to 32x
- Transmit and receive baud rate can operate asynchronous to the bus clock:
  - Baud rate can be configured independently of the bus clock frequency
  - Supports operation in Stop 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
- 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 break character generation / 11-bit break character detection
- 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

40.2.2 Modes of operation

40.2.2.1 Stop mode

The LPUART will remain functional during Stop mode, provided the CTRL[DOZEEN] bit is clear and the asynchronous transmit and receive clock remain enabled. The LPUART can generate an interrupt or DMA request to cause a wakeup from Stop mode.

If the LPUART is disabled in Stop mode, then it can generate a wakeup via the STAT[RXEDGIF] flag if the receiver detects an active edge.

40.2.2.2 Wait mode

The LPUART can be configured to Stop in Wait modes, when the CTRL[DOZEEN] bit is set. The transmitter and receiver will finish transmitting/receiving the current word.

40.2.3 Signal Descriptions

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

40.2.4 Block diagram

The following figure shows the transmitter portion of the LPUART.

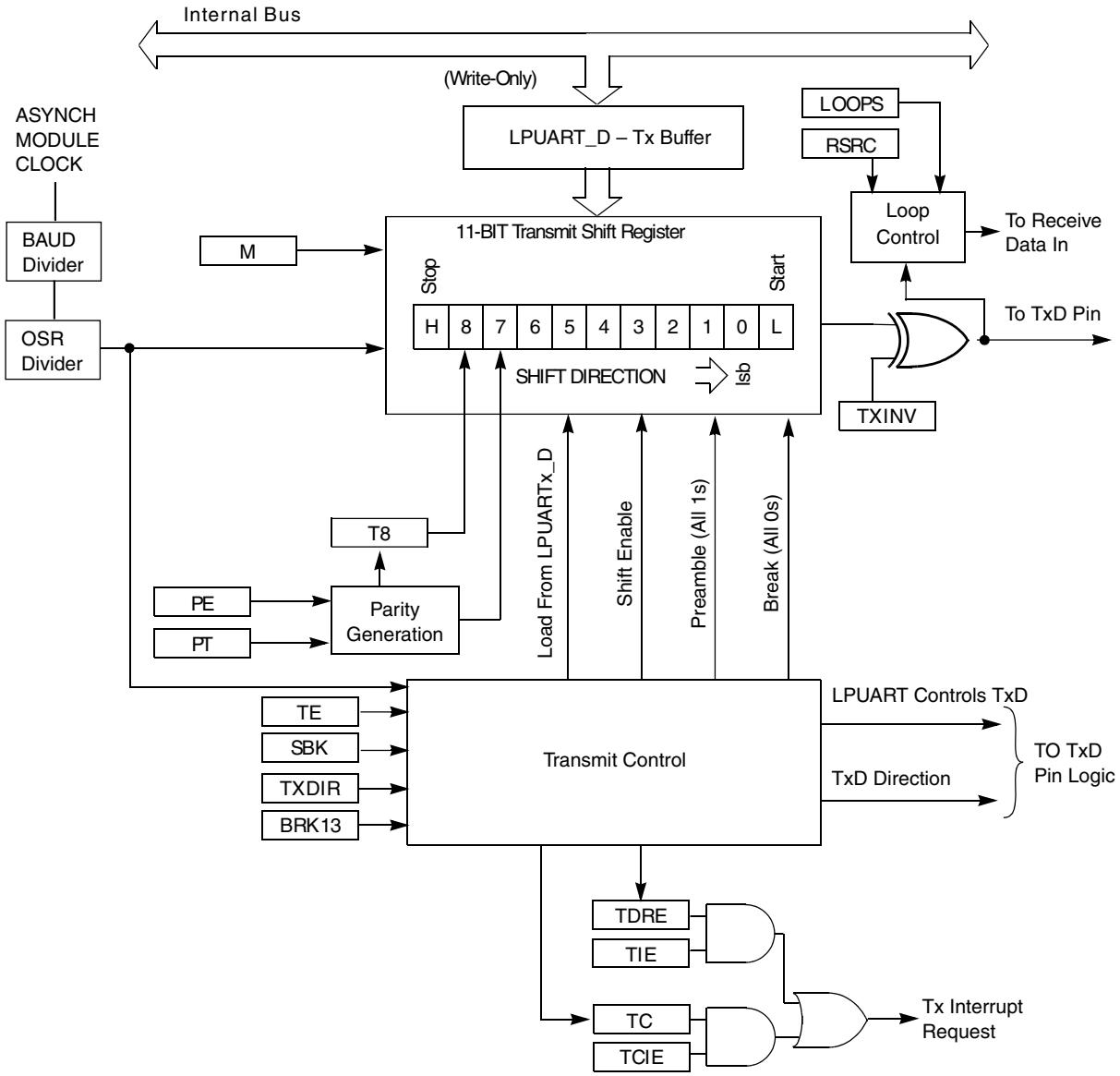


Figure 40-1. LPUART transmitter block diagram

The following figure shows the receiver portion of the LPUART.

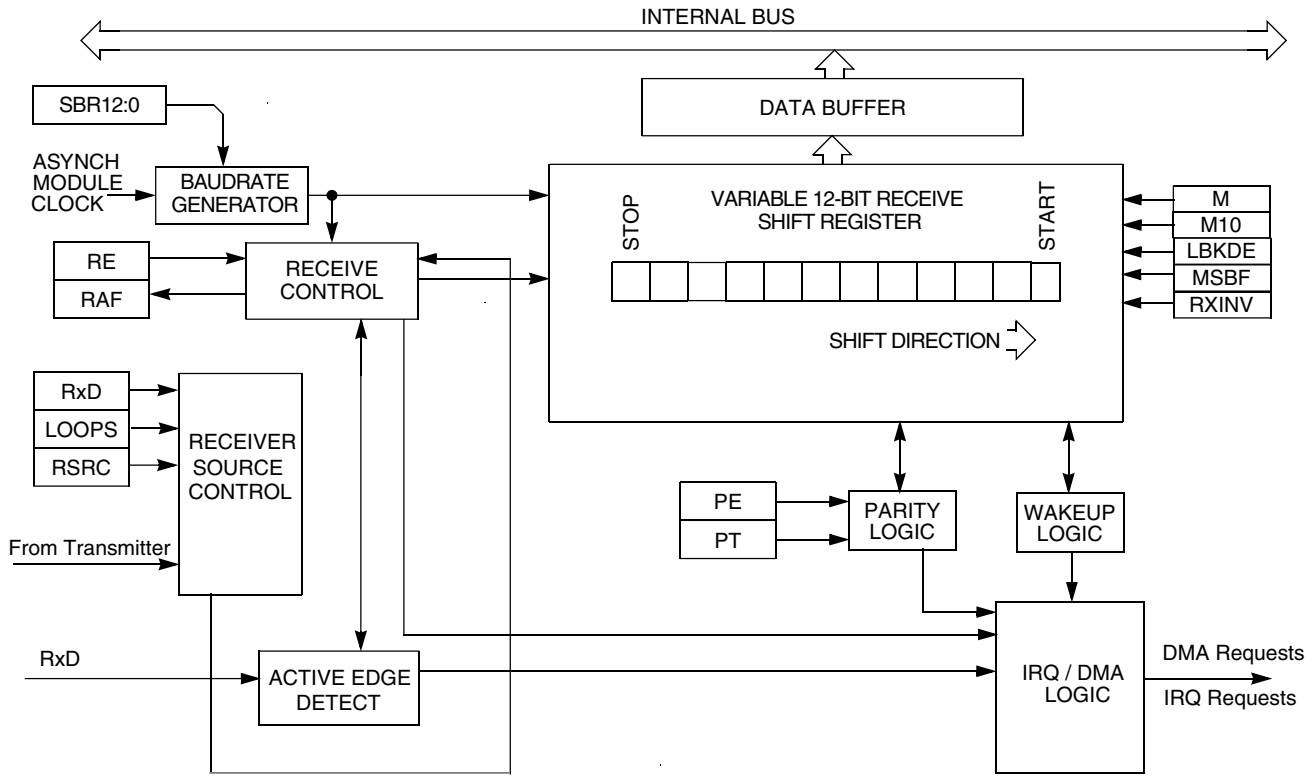


Figure 40-2. LPUART receiver block diagram

40.3 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.

40.3.1 Clocking and Resets

Table 40-2. 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 40-3. 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.

40.3.2 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. Depending on the over sampling ratio, the receiver has an acquisition rate of 4 to 32 samples per bit time.

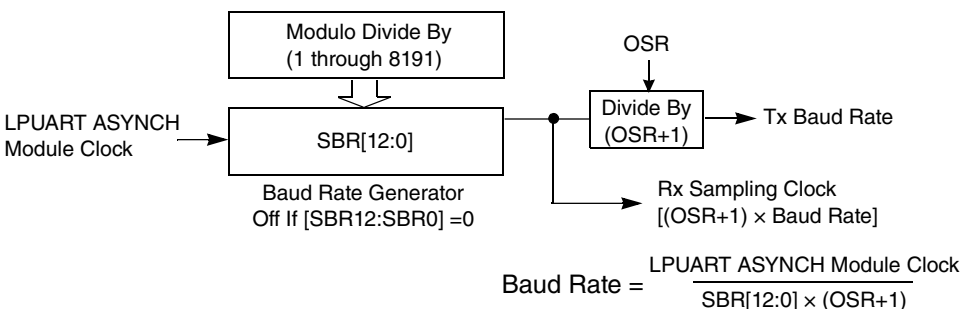


Figure 40-3. LPUART baud rate generation

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 will align to the next edge of the transmit bit clock.

40.3.3 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, CTRL[TXINV] is cleared following reset. The transmitter output is inverted by setting CTRL[TXINV]. 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 data buffer. Programs store data into the transmit data buffer by writing to the DATA register.

The central element of the LPUART transmitter is the transmit shift register that is 9-bit to 13 bits 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 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 data buffer at the DATA register.

If no new character is waiting in the transmit data buffer after a stop bit is shifted out 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 will not start transmitting another character.

40.3.3.1 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 would wait 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 bit 13 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 would wait for STAT[TDRE] to become set to indicate the last character of a message has moved to the transmit shifter, then 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 bit 13 set and the data bits also set. 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 40-4. 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
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

40.3.3.2 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 data buffer, the character in the shift register is sent and TXD remains in the mark 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 also be enabled even if the same LPUART receiver's RTS_B signal is disabled.

40.3.3.3 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 transmitter data buffer, RTS_B asserts one bit time before the start bit is transmitted. RTS_B remains asserted for the whole time that the transmitter data buffer has any characters. RTS_B deasserts one bit time after all characters in the transmitter data buffer and shift register are completely sent, including the last stop bit. Transmitting a break character also asserts RTS_B, with the same assertion and deassertion timing as having a character in the transmitter data buffer.

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 will remain asserted until the transfer is completed, even if the transmitter is disabled mid-way through a data transfer.

40.3.3.4 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.

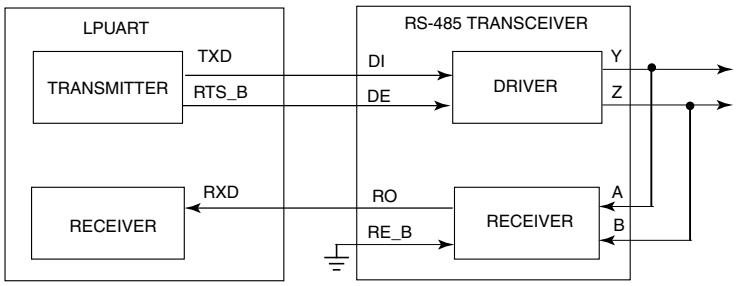


Figure 40-4. Transceiver driver enable using RTS_B

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.

40.3.4 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. Because the LPUART receiver is double-buffered, the program has one full character time after [RDRF] is set before the data in the receive data buffer must be read to avoid a receiver overrun.

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.

40.3.4.1 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 data buffer.

When the LPUART receiver is configured to sample on both edges of the baud rate clock, 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 falling edge detection logic continuously looks for falling edges. If an edge is detected, the sample clock is resynchronized to bit times (unless resynchronization has been disabled). This improves the reliability of the receiver in the presence of noise or mismatched baud rates. It does not improve worst case analysis because some characters do not have any extra falling edges anywhere in the character frame.

In the case of a framing error, provided the received character was not a break character, the sampling logic that searches for a falling edge is filled with three logic 1 samples so that a new start bit can be detected almost immediately.

40.3.4.2 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, or at the beginning of the next 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 will ignore all characters that do not meet the address match requirements.

Table 40-5. Receiver Wakeup Options

RWU	MA1 | MA2	MATCFG	WAKE:RWUID	Receiver Wakeup
0	0	X	X	Normal operation
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	Address match on and address match off, IDLE flag not set for discarded characters
0	1	10	X1	Address match on and address match off, IDLE flag set for discarded characters

40.3.4.2.1 Idle-line wakeup

When 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 bit selects 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 one 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.

40.3.4.2.2 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.

40.3.4.2.3 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.

40.3.4.2.4 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 data buffer. If no marked address match occurs then no transfer is made to the receive data buffer, 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 data buffer.

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 data buffer 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.

40.3.4.2.5 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 only transferred 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 data buffer until the next idle line condition is detected. If no address match occurs then no transfer is made to the receive data buffer, 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 data buffer.

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 data buffer 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.

40.3.4.2.6 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 data buffer, 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 data buffer.

NOTE

Match on, match off operation requires both BAUD[MAEN1] and BAUD[MAEN2] to be asserted.

40.3.4.3 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 will cause 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 will cause 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 receiver data buffer is overrun.
- If the receiver request-to-send functionality is disabled, the receiver RTS_B remains deasserted.

40.3.4.4 Infrared 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.

40.3.4.4.1 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.

40.3.4.4.2 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.

40.3.4.4.3 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.

40.3.4.4.4 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.

40.3.5 Additional LPUART functions

The following sections describe additional LPUART functions.

40.3.5.1 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. For the transmit data buffer, these bits are stored in CTRL[T8] and CTRL[T9]. For the receiver, these bits are held in CTRL[R8] and CTRL[R9]. They are also accessible via 16-bit or 32-bit accesses to the DATA register.

For coherent 8-bit writes to the transmit data buffer, write to CTRL[T8] and CTRL[T9] before writing to DATA[7:0]. For 16-bit and 32-bit writes to the DATA register, all 10 transmit bits are written to the transmit data buffer at the same time.

If the bit values to be transmitted as the ninth and tenth bit of a new character are the same as for the previous character, it is not necessary to write to CTRL[T8] and CTRL[T9] again. When data is transferred from the transmit data buffer to the transmit shifter, the value in CTRL[T8] and CTRL[T9] is copied at the same time data is transferred from DATA[7:0] to the shifter.

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.

40.3.5.2 Idle length

An idle character is a character where the start bit, all data bits and stop bits are in the mark position. 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 will cause any discarded characters to be treated as if they were idle characters.

40.3.5.3 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.

40.3.5.4 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.

40.3.6 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. This module covers data rates only between 2.4 kbits/s and 115.2 kbits/s.

The LPUART has an infrared transmit encoder and receive decoder. 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 and transformed to CMOS levels by the IR receive decoder, external from the LPUART. 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.

40.3.6.1 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.

40.3.6.2 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.

40.3.7 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 data buffer 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 was 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] was already set when a new character is ready to be transferred from the receive shifter to the receive data buffer, the overrun (STAT[OR]) flag is set instead of the data along with any associated NF, FE, or 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).

40.3.8 Peripheral Triggers

The connection of the LPUART peripheral triggers with other peripherals are device specific.

40.3.8.1 Output Triggers

The LPUART generates three 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 at when the idle flag would set, for one oversampling clock period.

40.3.8.2 Input Trigger

The LPUART supports one peripheral input trigger, that can be configured in one of the following ways.

- 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 generated from a PWM source with a period that is less than the bit clock frequency.
- 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.

40.4 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 will generate a bus error.

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.

40.4.1 LPUART register descriptions

40.4.1.1 LPUART Memory map

LPUART*n* base address: 4018_4000h + (*n*-1)×4000h, where *n* is from 1 to 4.

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
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

40.4.1.2 Version ID Register (VERID)

40.4.1.2.1 Offset

Register	Offset
VERID	0h

40.4.1.2.2 Function

The Version ID register indicates the version integrated for this instance on the device and also indicates inclusion/exclusion of several optional features.

40.4.1.2.3 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

40.4.1.2.4 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.

40.4.1.3 Parameter Register (PARAM)

40.4.1.3.1 Offset

Register	Offset
PARAM	4h

40.4.1.3.2 Function

The Parameter register indicates the parameter configuration for this instance on the device

40.4.1.3.3 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

40.4.1.3.4 Fields

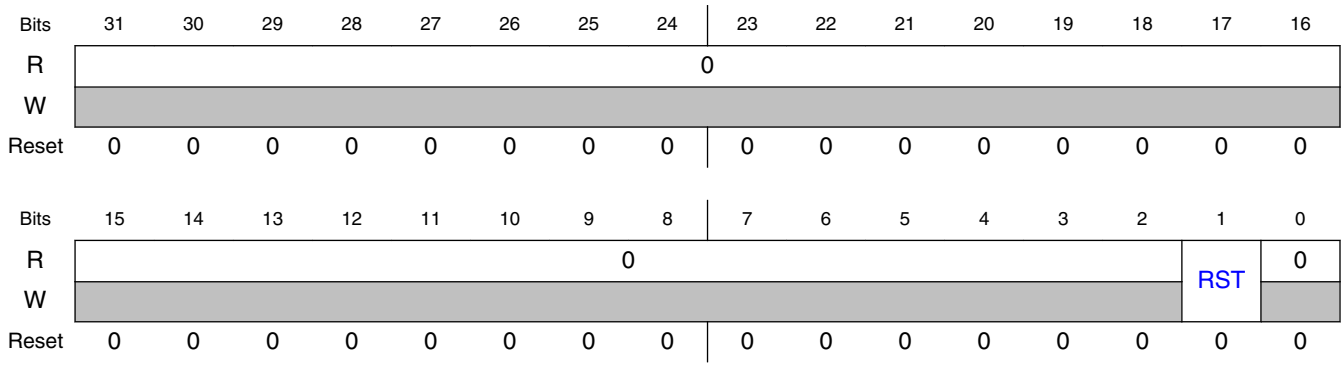
Field	Function
31-16 —	Reserved
15-8 RXFIFO	Receive FIFO Size The number of words in the receive FIFO is 2^RXFIFO.
7-0 TXFIFO	Transmit FIFO Size The number of words in the transmit FIFO is 2^TXFIFO.

40.4.1.4 LPUART Global Register (GLOBAL)

40.4.1.4.1 Offset

Register	Offset
GLOBAL	8h

40.4.1.4.2 Diagram



40.4.1.4.3 Fields

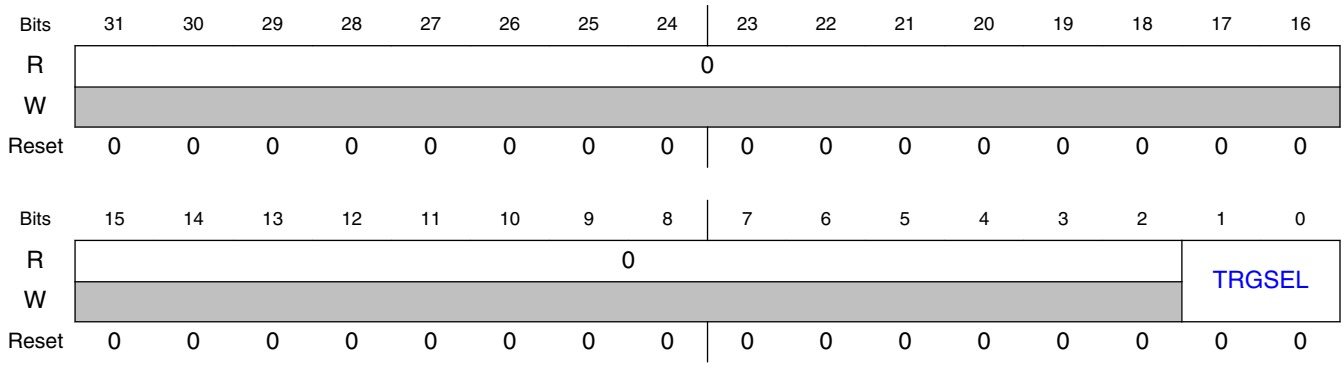
Field	Function
31-2 —	Reserved
1 RST	Software Reset Resets all internal logic and registers, except the Global Register. Remains set until cleared by software. 0b - Module is not reset. 1b - Module is reset.
0 —	Reserved

40.4.1.5 LPUART Pin Configuration Register (PINCFG)

40.4.1.5.1 Offset

Register	Offset
PINCFG	Ch

40.4.1.5.2 Diagram



40.4.1.5.3 Fields

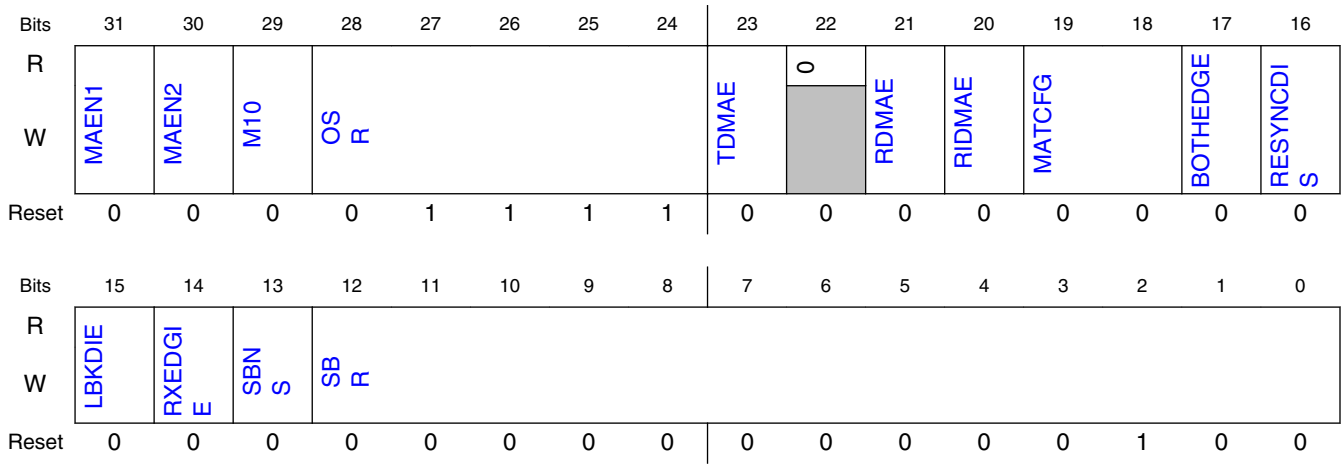
Field	Function
31-2 —	Reserved
1-0 TRGSEL	Trigger Select Configures the input trigger usage. This field should only be changed when the transmitter and receiver are both disabled. 00b - Input trigger is disabled. 01b - Input trigger is used instead of RXD pin input. 10b - Input trigger is used instead of CTS_B pin input. 11b - Input trigger is used to modulate the TXD pin output. The TXD pin output (after TXINV configuration) is ANDed with the input trigger.

40.4.1.6 LPUART Baud Rate Register (BAUD)

40.4.1.6.1 Offset

Register	Offset
BAUD	10h

40.4.1.6.2 Diagram



40.4.1.6.3 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 only be changed when the transmitter and receiver are both disabled. 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 only be changed when the transmitter and receiver are both disabled. 00000b - Writing 0 to this field will result 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.

Table continues on the next page...

Field	Function
	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 RIDMAE	Receiver Idle DMA Enable RIDMAE configures the receiver idle flag, STAT[IDLE], to generate a DMA request. When this bit is set, reading the DATA register when either DATA[RXEMPT] or DATA[IDLINE] bit is set, will generate an End Of Packet response until the completion of the existing DMA transfer. During an End of Packet response, reading the DATA register will return 0x0000_33FF and does not pull data from the FIFO. 0b - DMA request disabled. 1b - DMA request enabled.
19-18 MATCFG	Match Configuration Configures the match addressing mode used. This field should only be changed when the transmitter and receiver are both disabled. 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 only be changed 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, disables the resynchronization of the received data word when a data one followed by data zero transition is detected. This bit should only be changed when the receiver is disabled. 0b - Resynchronization during received data word is supported 1b - Resynchronization during received data word is disabled

Table continues on the next page...

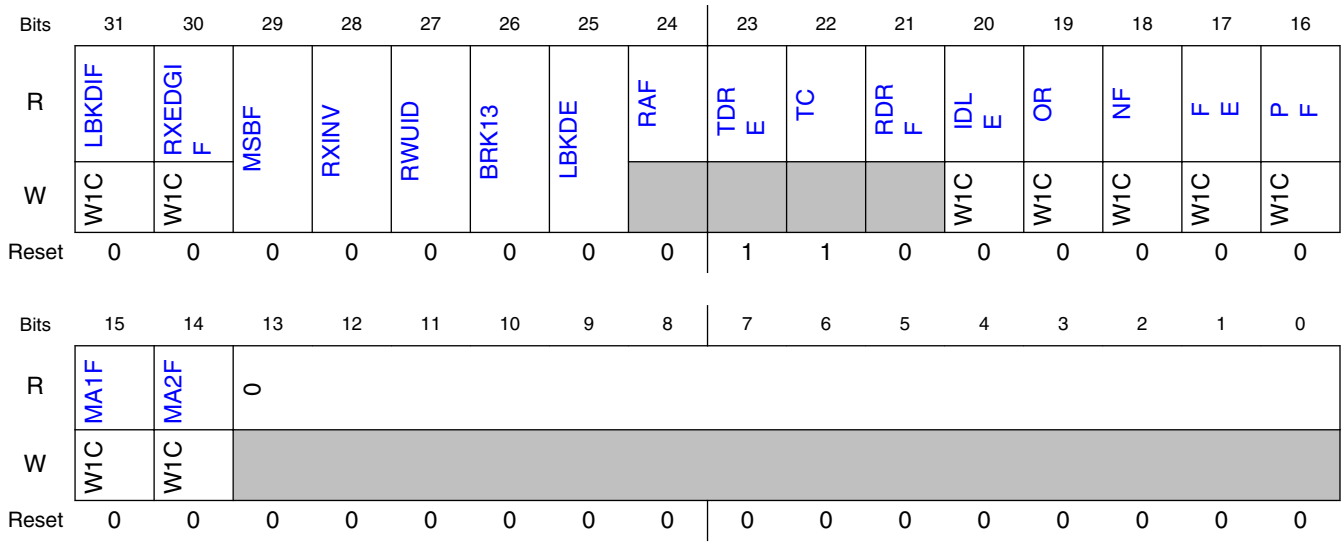
Field	Function
15 LBKDIE	LIN Break Detect Interrupt Enable LBKDIE enables the LIN break detect flag, STAT[LBKDIF], to generate interrupt requests. 0b - Hardware interrupts from STAT[LBKDIF] flag are disabled (use polling). 1b - Hardware interrupt requested when STAT[LBKDIF] flag is 1.
14 RXEDGIE	RX Input Active Edge Interrupt Enable Enables the receive input active edge, STAT[RXEDGIF], to generate interrupt requests. Changing CTRL[LOOP] 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 only be changed 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).

40.4.1.7 LPUART Status Register (STAT)

40.4.1.7.1 Offset

Register	Offset
STAT	14h

40.4.1.7.2 Diagram



40.4.1.7.3 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. LBKDIF is cleared by writing a 1 to it. 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. RXEDGIF is cleared by writing a 1 to it. 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 only be changed 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 (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]. Further, the first bit received after the start bit is identified as bit9, bit8, bit7 or bit6 depending on the setting of CTRL[M] and CTRL[PE].
28 RXINV	Receive Data Inversion Setting this bit reverses the polarity of the received data input. This bit should only be changed 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	Receive Wake Up Idle Detect

Table continues on the next page...

Field	Function
RWUID	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 only be changed 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. Detection of a framing error is not affected by the state of this bit. This bit should only be changed 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 data buffer. 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 will set when the number of datawords in the transmit FIFO (DATA register) 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 will 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 data buffer full. 1b - Transmit data buffer empty.
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 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 data buffer 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.

Table continues on the next page...

Field	Function
	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 will continue to be received until an overrun condition occurs once the entire character is received. 0b - Receive data buffer empty. 1b - Receive data buffer full.
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 has been 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 was 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 data buffer 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 was 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 was 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 was 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. 0b - No parity error. 1b - Parity error.
15 MA1F	Match 1 Flag

Table continues on the next page...

Field	Function
	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
13-0 —	Reserved

40.4.1.8 LPUART Control Register (CTRL)

40.4.1.8.1 Offset

Register	Offset
CTRL	18h

40.4.1.8.2 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.

40.4.1.8.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	R8T9	R9T8	TXDIR	TXINV	ORIE	NEIE	FEIE	PEIE	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
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	MA1IE	MA2IE	0		M7	IDLECFG			LOOPS	DOZEEN	RSRC	M	WAKE	ILT	PE	PT
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

40.4.1.8.4 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, they 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 will finish 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 ORIE	Overrun Interrupt Enable This bit enables the overrun flag (OR) to generate hardware interrupt requests. 0b - OR interrupts disabled; use polling. 1b - Hardware interrupt 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 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 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 requested when PF is set.
23	Transmit Interrupt Enable

Table continues on the next page...

Field	Function
TIE	Enables STAT[TDRE] to generate interrupt requests. 0b - Hardware interrupts from TDRE disabled; use polling. 1b - Hardware interrupt 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; use polling. 1b - Hardware interrupt requested when TC flag is 1.
21 RIE	Receiver Interrupt Enable Enables STAT[RDRF] to generate interrupt requests. 0b - Hardware interrupts from RDRF disabled; use polling. 1b - Hardware interrupt 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 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 will read 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 will read as 1 until the receiver finishes receiving the current character (if any). 0b - Receiver disabled. 1b - Receiver enabled.
17 RWU	Receiver Wakeup Control 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 with 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. 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.

Table continues on the next page...

Field	Function
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 only be changed 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. 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, or • An idle condition on the receive pin input signal. 0b - Configures RWU for idle-line wakeup. 1b - Configures RWU with address-mark wakeup.

Table continues on the next page...

Field	Function
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 1s 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.

40.4.1.9 LPUART Data Register (DATA)

40.4.1.9.1 Offset

Register	Offset
DATA	1Ch

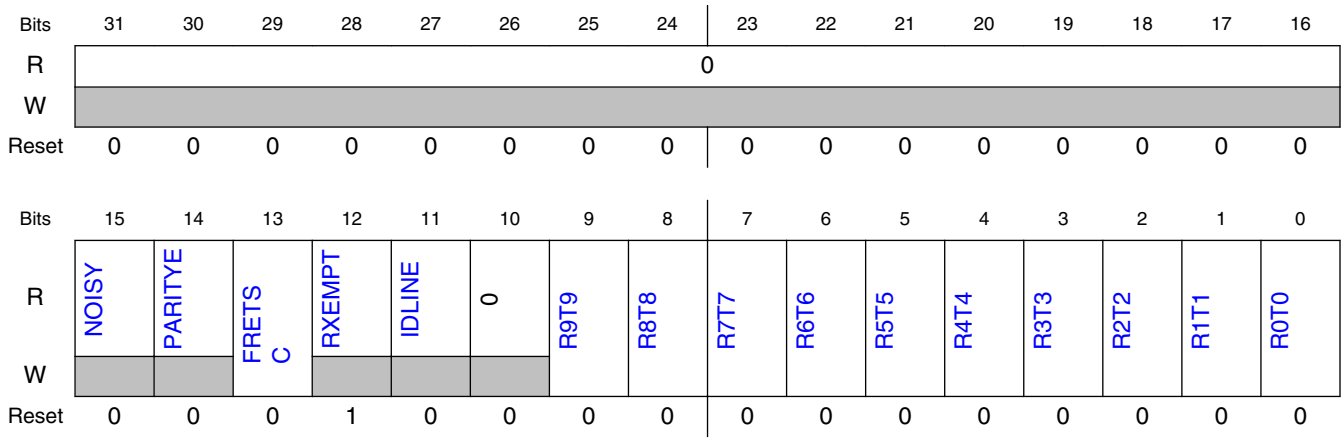
40.4.1.9.2 Function

NOTE

This register is two separate registers. Reads return the contents of the read-only receive data buffer and writes go to the write-only transmit data buffer.

Reads and writes of this register are also involved in the automatic flag clearing mechanisms for some of the LPUART status flags.

40.4.1.9.3 Diagram



40.4.1.9.4 Fields

Field	Function
31-16 —	Reserved
15 NOISY	NOISY The current received dataword contained in DATA[R9:R0] was received with noise. 0b - The dataword was received without noise. 1b - The data was received with noise.
14 PARITYE	PARITYE The current received dataword contained in DATA[R9:R0] was received with a parity error. 0b - The dataword was received without a parity error. 1b - The dataword was 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 was received without a frame error on read, or transmit a normal character on write. 1b - The dataword was 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.

Table continues on the next page...

Field	Function
10 —	Reserved
9 R9T9	R9T9 Read receive data buffer 9 or write transmit data buffer 9.
8 R8T8	R8T8 Read receive data buffer 8 or write transmit data buffer 8.
7 R7T7	R7T7 Read receive data buffer 7 or write transmit data buffer 7.
6 R6T6	R6T6 Read receive data buffer 6 or write transmit data buffer 6.
5 R5T5	R5T5 Read receive data buffer 5 or write transmit data buffer 5.
4 R4T4	R4T4 Read receive data buffer 4 or write transmit data buffer 4.
3 R3T3	R3T3 Read receive data buffer 3 or write transmit data buffer 3.
2 R2T2	R2T2 Read receive data buffer 2 or write transmit data buffer 2.
1 R1T1	R1T1 Read receive data buffer 1 or write transmit data buffer 1.
0 R0T0	R0T0 Read receive data buffer 0 or write transmit data buffer 0.

40.4.1.10 LPUART Match Address Register (MATCH)

40.4.1.10.1 Offset

Register	Offset
MATCH	20h

40.4.1.10.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0						MA2									
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						MA1									
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

40.4.1.10.3 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.

40.4.1.11 LPUART Modem IrDA Register (MODIR)

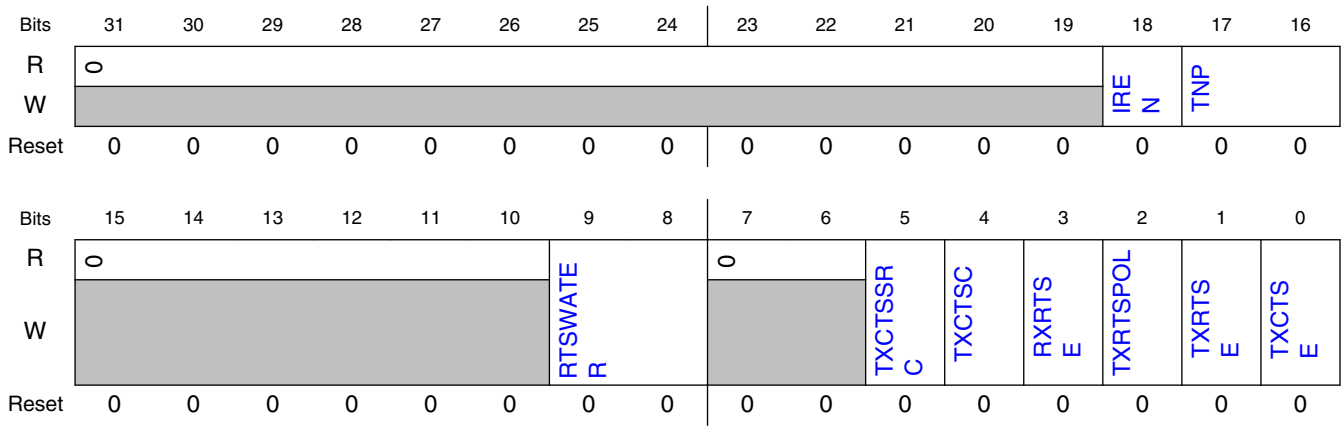
40.4.1.11.1 Offset

Register	Offset
MODIR	24h

40.4.1.11.2 Function

The MODEM register controls options for setting the modem configuration.

40.4.1.11.3 Diagram



40.4.1.11.4 Fields

Field	Function
31-19 —	Reserved
18 IREN	Infrared enable Enables/disables the infrared modulation/demodulation. This bit should only be changed 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 only be changed when the transmitter and receiver are both disabled. 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.

Table continues on the next page...

Field	Function
	This field should only be changed 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 the inverted Receiver 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 only be changed 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 will remain negated in the active low state unless TXRTSE is set. This bit should only be changed 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 only be changed when the transmitter is disabled. 0b - The transmitter has no effect on RTS. 1b - When a character is placed into an empty transmitter data buffer , RTS asserts one bit time before the start bit is transmitted. RTS deasserts one bit time after all characters in the transmitter data buffer 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.

40.4.1.12 LPUART FIFO Register (FIFO)

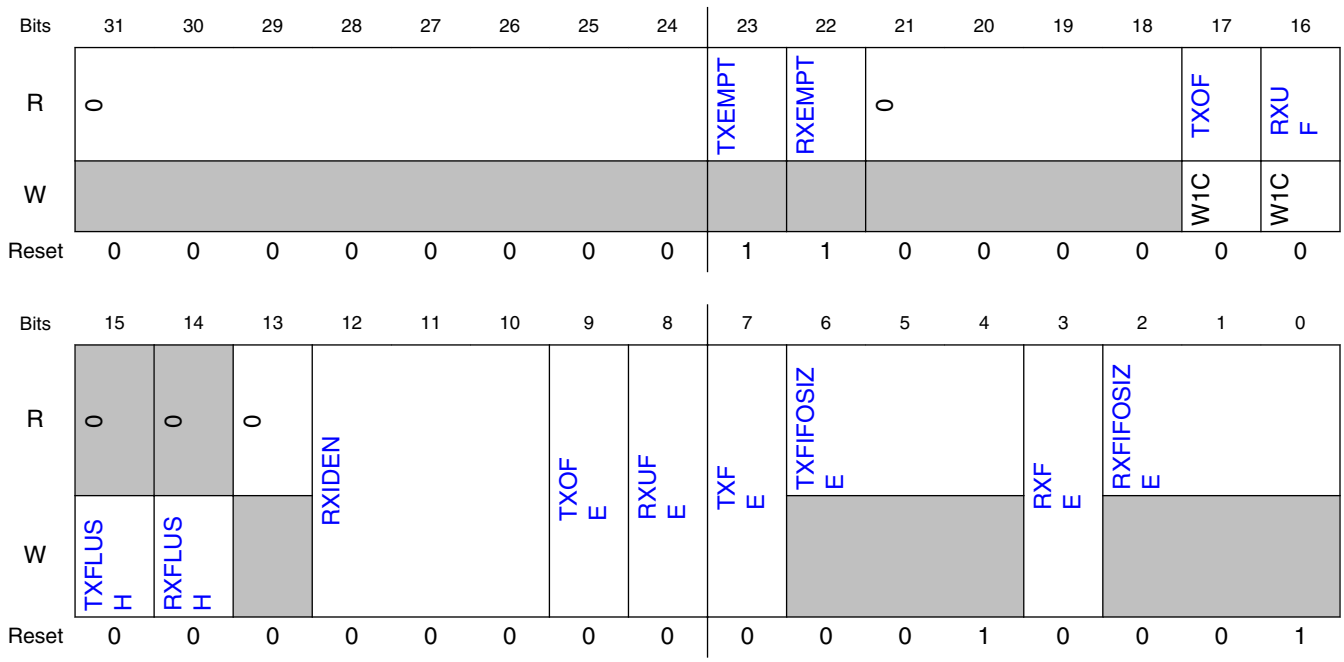
40.4.1.12.1 Offset

Register	Offset
FIFO	28h

40.4.1.12.2 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 data buffer/FIFO is empty.

40.4.1.12.3 Diagram



40.4.1.12.4 Fields

Field	Function
31-24 —	Reserved
23 TXEMPT	Transmit Buffer/FIFO 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.

Table continues on the next page...

Field	Function
	1b - Transmit buffer is empty.
22 RXEMPT	Receive Buffer/FIFO 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 Buffer Overflow Flag Indicates that more data has been written to the transmit buffer than it can hold. This field will assert regardless of the value of TXOFE. However, an interrupt will be issued to the host only if TXOFE is set. This flag is cleared by writing a 1. 0b - No transmit buffer overflow has occurred since the last time the flag was cleared. 1b - At least one transmit buffer overflow has occurred since the last time the flag was cleared.
16 RXUF	Receiver Buffer Underflow Flag Indicates that more data has been read from the receive buffer than was present. This field will assert regardless of the value of RXUFE. However, an interrupt will be issued to the host only if RXUFE is set. This flag is cleared by writing a 1. 0b - No receive buffer underflow has occurred since the last time the flag was cleared. 1b - At least one receive buffer underflow has occurred since the last time the flag was cleared.
15 TXFLUSH	Transmit FIFO/Buffer Flush Writing to this field causes all data that is stored in the transmit FIFO/buffer 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/Buffer is cleared out.
14 RXFLUSH	Receive FIFO/Buffer Flush Writing to this field causes all data that is stored in the receive FIFO/buffer 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	Receive FIFO Underflow Interrupt Enable

Table continues on the next page...

Field	Function
RXUFE	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 is depth 1. 1b - Transmit FIFO is enabled. Buffer is depth 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. 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 is depth 1. 1b - Receive FIFO is enabled. Buffer is depth 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.

40.4.1.13 LPUART Watermark Register (WATER)

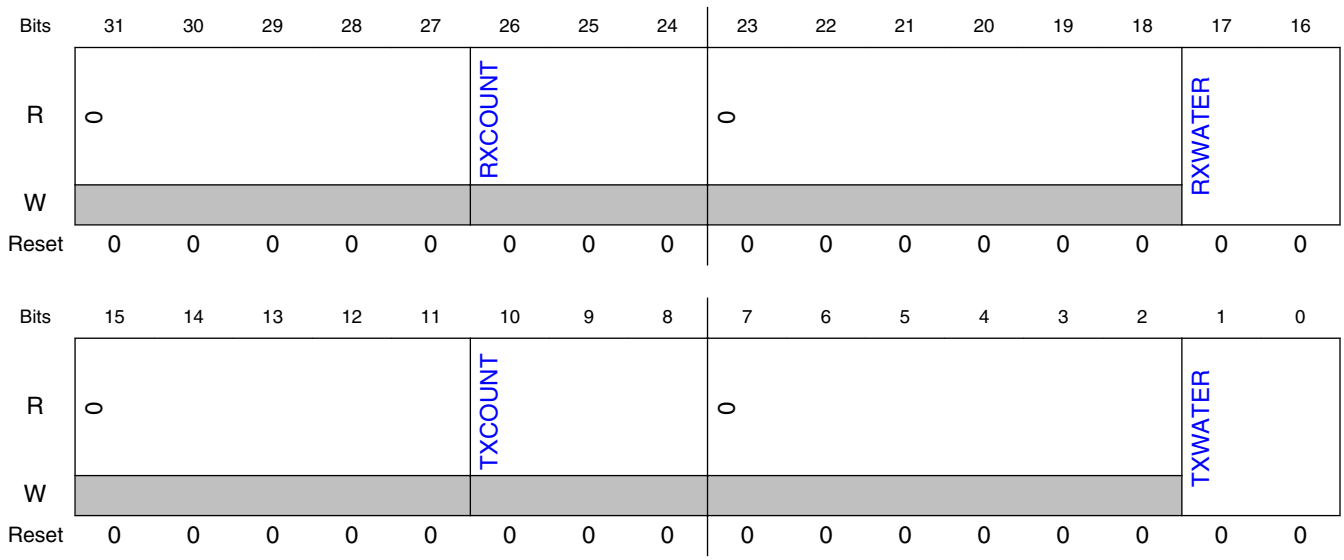
40.4.1.13.1 Offset

Register	Offset
WATER	2Ch

40.4.1.13.2 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.

40.4.1.13.3 Diagram



40.4.1.13.4 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] and must be greater than 0.
15-11 —	Reserved

Table continues on the next page...

Field	Function
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 TXWATER	Transmit Watermark 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 41
Flexible I/O (FlexIO)

41.1 Chip-specific FlexIO 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
Trigger mapping	-	XBAR1 Output Assignments

NOTE

FLEXIO only has 27 pins, in this device.

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.

41.2 Introduction

41.2.1 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

41.2.2 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 and data match 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

41.2.3 Block Diagram

The following diagram gives a high-level overview of the configuration of FlexIO timers and shifters.

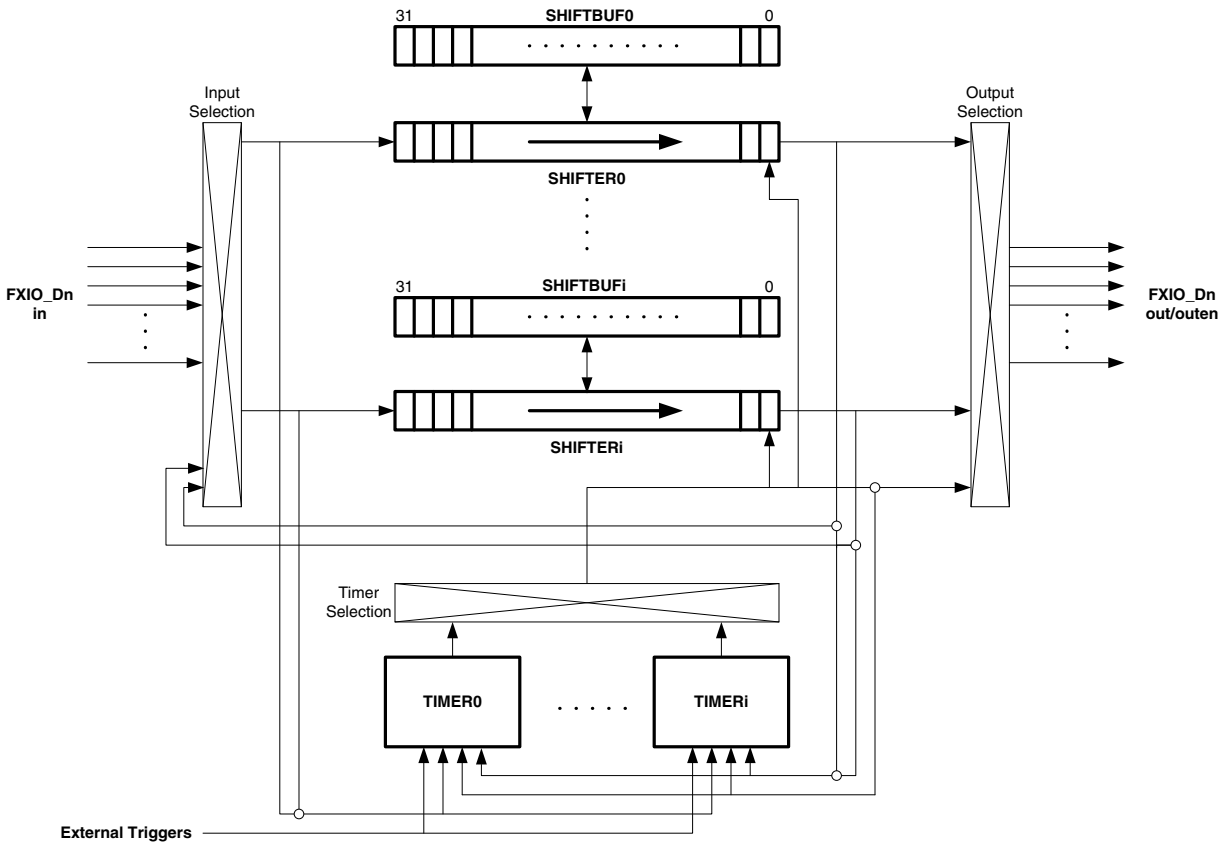


Figure 41-1. FlexIO block diagram

41.2.4 Modes of operation

The FlexIO module supports the chip modes described in the following table.

Table 41-2. Chip modes supported by the FlexIO module

Chip mode	FlexIO Operation
Run	Normal operation
Stop/Wait	Can continue operating provided the Doze Enable bit (CTRL[DOZEN]) is clear and the FlexIO is using an external or internal clock source which remains operating during stop/wait modes.
LLS	The Doze Enable (CTRL[DOZEN]) bit is ignored and the FlexIO will wait for all Timers to complete any pending operation before acknowledging LLS mode entry.
Debug	Can continue operating provided the Debug Enable bit (CTRL[DBGE]) is set.

41.2.5 FlexIO Signal Descriptions

Signal	Description	I/O
FXIO_Dn (n=0...31)	Bidirectional FlexIO Shifter and Timer pin inputs/outputs	I/O

41.3 Memory Map and Registers

41.3.1 FLEXIO register descriptions

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.

41.3.1.1 FLEXIO memory map

FLEXIO base address: 401A_C000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Version ID Register (VERID)	32	RO	0101_0001h
4h	Parameter Register (PARAM)	32	RO	0220_0808h
8h	FlexIO Control Register (CTRL)	32	RW	0000_0000h

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
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
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

41.3.1.2 Version ID Register (VERID)

41.3.1.2.1 Offset

Register	Offset
VERID	0h

41.3.1.2.2 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

41.3.1.2.3 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 Specification Number This read only field returns the feature set number. 0000000000000000b - Standard features implemented. 0000000000000001b - Supports state, logic and parallel modes.

41.3.1.3 Parameter Register (PARAM)

41.3.1.3.1 Offset

Register	Offset
PARAM	4h

41.3.1.3.2 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	TRIGGER								PIN							
W																
Reset	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0

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

41.3.1.3.3 Fields

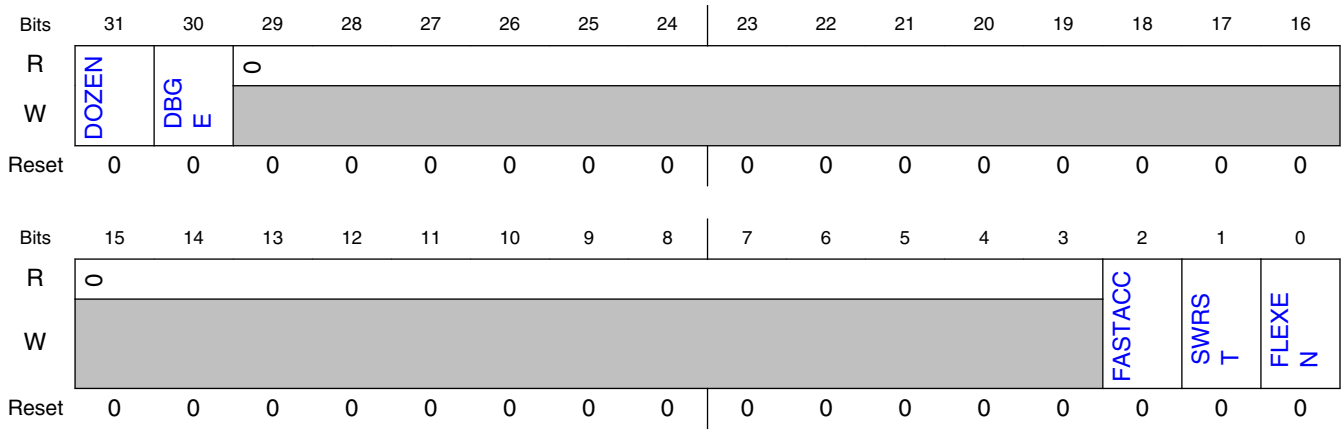
Field	Function
31-24 TRIGGER	Trigger Number Number of external triggers implemented.
23-16 PIN	Pin Number Number of Pins implemented.
15-8 TIMER	Timer Number Number of Timers implemented.
7-0 SHIFTER	Shifter Number Number of Shifters implemented.

41.3.1.4 FlexIO Control Register (CTRL)

41.3.1.4.1 Offset

Register	Offset
CTRL	8h

41.3.1.4.2 Diagram



41.3.1.4.3 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 DBGE	Debug Enable 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 twice 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 register accesses are ignored until this bit is cleared. This register bit will remain 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.

41.3.1.5 Pin State Register (PIN)

41.3.1.5.1 Offset

Register	Offset
PIN	Ch

41.3.1.5.2 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

41.3.1.5.3 Fields

Field	Function
31-0 PDI	Pin Data Input Returns the input data on each of the FlexIO pins.

41.3.1.6 Shifter Status Register (SHIFTSTAT)

41.3.1.6.1 Offset

Register	Offset
SHIFTSTAT	10h

41.3.1.6.2 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

41.3.1.6.3 Fields

Field	Function
31-8 —	Reserved
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 will set 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.

41.3.1.7 Shifter Error Register (SHIFTERR)

41.3.1.7.1 Offset

Register	Offset
SHIFTERR	14h

41.3.1.7.2 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

41.3.1.7.3 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.

41.3.1.8 Timer Status Register (TIMSTAT)

41.3.1.8.1 Offset

Register	Offset
TIMSTAT	18h

41.3.1.8.2 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

41.3.1.8.3 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.

41.3.1.9 Shifter Status Interrupt Enable (SHIFTSIEN)

41.3.1.9.1 Offset

Register	Offset
SHIFTSIEN	20h

41.3.1.9.2 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

41.3.1.9.3 Fields

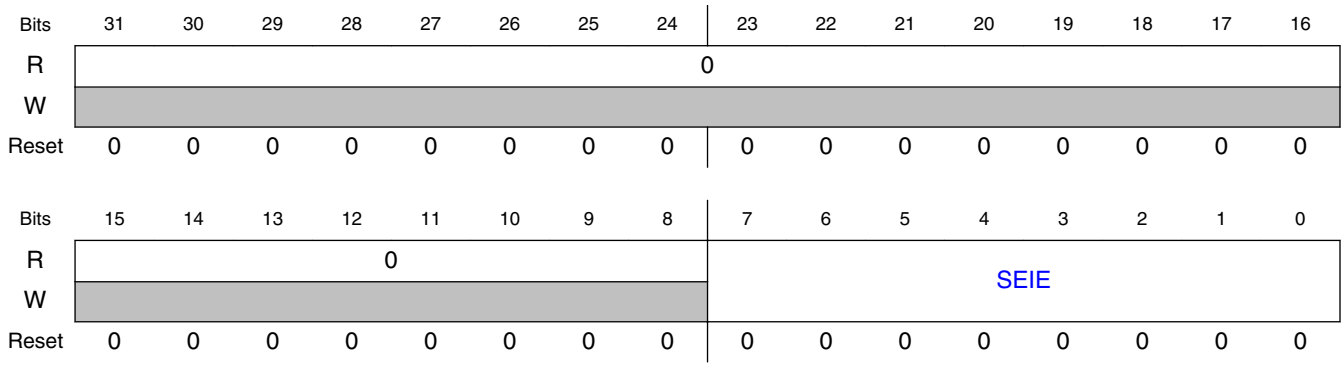
Field	Function
31-8 —	Reserved
7-0 SSIE	Shifter Status Interrupt Enable Enables interrupt generation when corresponding SSF is set. 0b - Shifter Status Flag interrupt disabled. 1b - Shifter Status Flag interrupt enabled.

41.3.1.10 Shifter Error Interrupt Enable (SHIFTEIEN)

41.3.1.10.1 Offset

Register	Offset
SHIFTEIEN	24h

41.3.1.10.2 Diagram



41.3.1.10.3 Fields

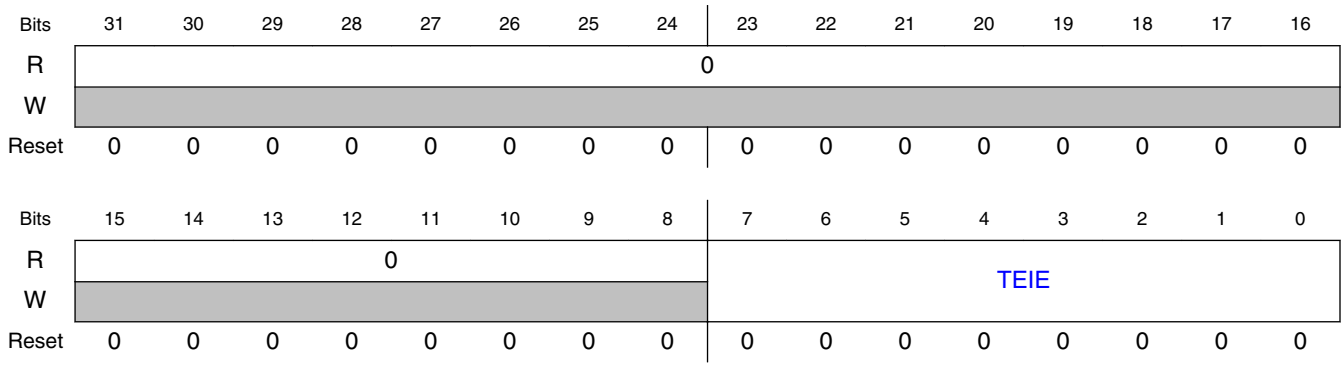
Field	Function
31-8 —	Reserved
7-0 SEIE	Shifter Error Interrupt Enable Enables interrupt generation when corresponding SEF is set. 0b - Shifter Error Flag interrupt disabled. 1b - Shifter Error Flag interrupt enabled.

41.3.1.11 Timer Interrupt Enable Register (TIMIEN)

41.3.1.11.1 Offset

Register	Offset
TIMIEN	28h

41.3.1.11.2 Diagram



41.3.1.11.3 Fields

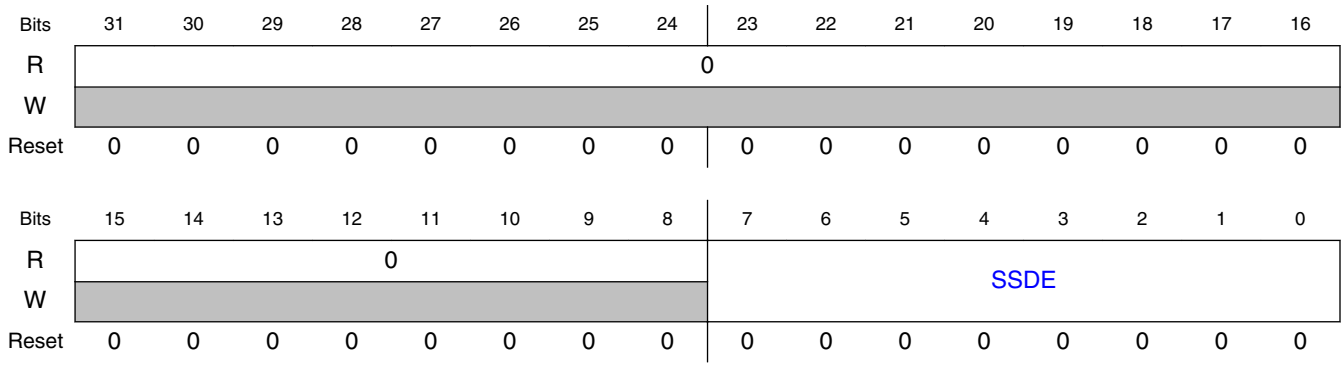
Field	Function
31-8 —	Reserved
7-0 TEIE	Timer Status Interrupt Enable Enables interrupt generation when corresponding TSF is set. 0b - Timer Status Flag interrupt is disabled. 1b - Timer Status Flag interrupt is enabled.

41.3.1.12 Shifter Status DMA Enable (SHIFTSDEN)

41.3.1.12.1 Offset

Register	Offset
SHIFTSDEN	30h

41.3.1.12.2 Diagram



41.3.1.12.3 Fields

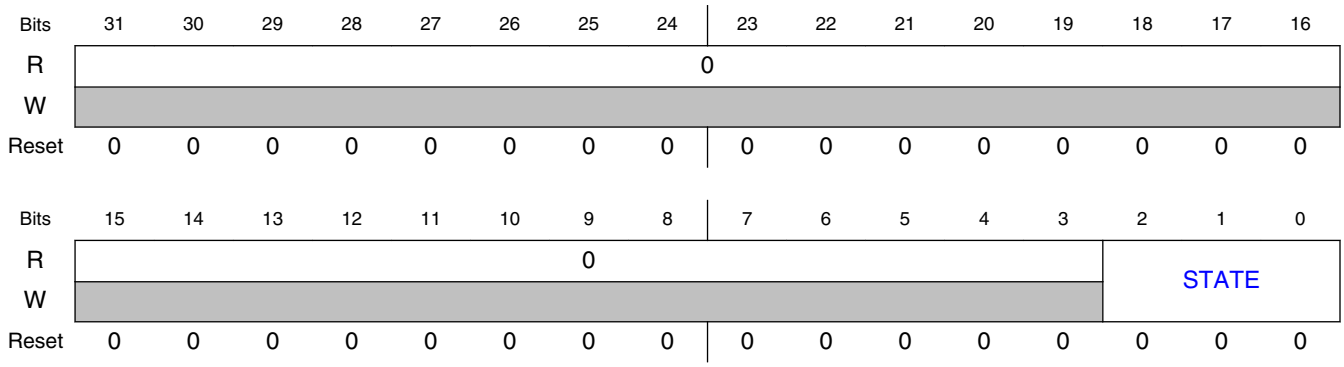
Field	Function
31-8 —	Reserved
7-0 SSDE	Shifter Status DMA Enable Enables DMA request generation when corresponding SSF is set. 0b - Shifter Status Flag DMA request is disabled. 1b - Shifter Status Flag DMA request is enabled.

41.3.1.13 Shifter State Register (SHIFTSTATE)

41.3.1.13.1 Offset

Register	Offset
SHIFTSTATE	40h

41.3.1.13.2 Diagram



41.3.1.13.3 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.

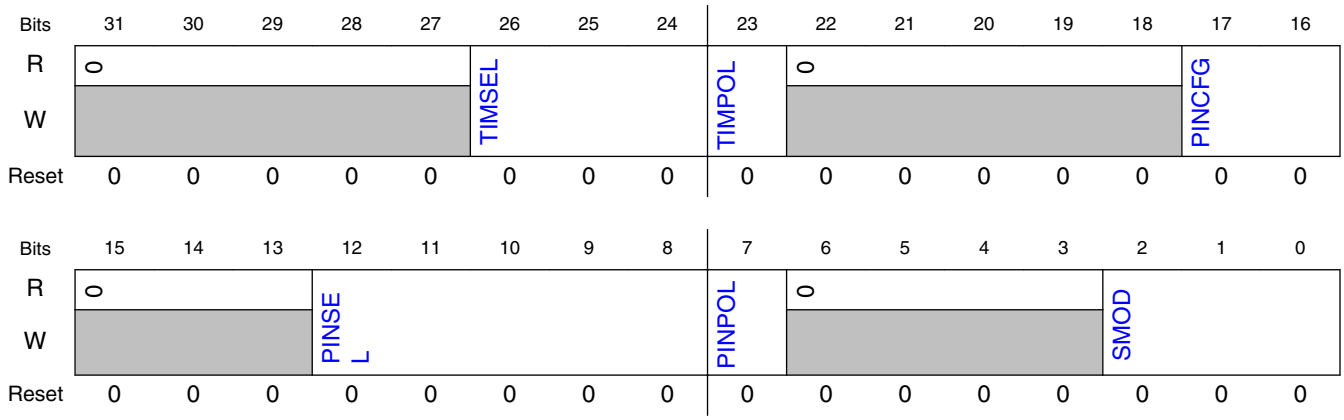
41.3.1.14 Shifter Control N Register (SHIFTCTL0 - SHIFTCTL7)

41.3.1.14.1 Offset

For a = 0 to 7:

Register	Offset
SHIFTCTLa	80h + (a × 4h)

41.3.1.14.2 Diagram



41.3.1.14.3 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 will select 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 will take 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 will select the FXIO_Di pin. For pins configured as an output (PINCFG=11), this field will take effect when the register is written.
7 PINPOL	Shifter Pin Polarity For pins configured as an output (PINCFG=11), this field will take effect when the register is written. 0b - Pin is active high 1b - Pin is active low
6-3 —	Reserved

Table continues on the next page...

Field	Function
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.

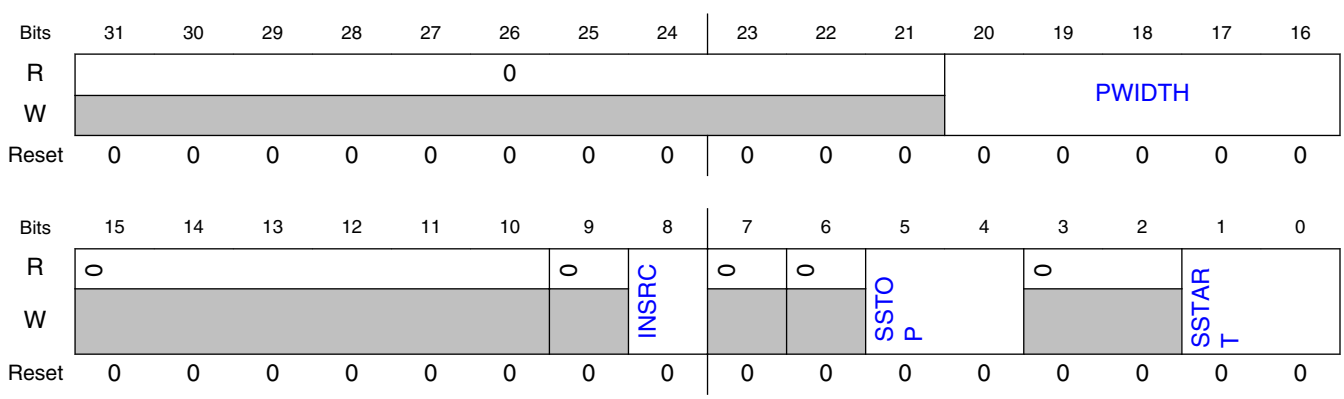
41.3.1.15 Shifter Configuration N Register (SHIFTCFG0 - SHIFTCFG7)

41.3.1.15.1 Offset

For a = 0 to 7:

Register	Offset
SHIFTCFGa	100h + (a × 4h)

41.3.1.15.2 Diagram



41.3.1.15.3 Fields

Field	Function
31-21	Reserved

Table continues on the next page...

Field	Function
—	
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 32-bit shift for PWIDTH=16...31 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.
15-10 —	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

Table continues on the next page...

Field	Function
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

41.3.1.16 Shifter Buffer N Register (SHIFTBUF0 - SHIFTBUF7)

41.3.1.16.1 Offset

For a = 0 to 7:

Register	Offset
SHIFTBUFa	200h + (a × 4h)

41.3.1.16.2 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

41.3.1.16.3 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.

41.3.1.17 Shifter Buffer N Bit Swapped Register (SHIFTBUFBIS0 - SHIFTBUFBIS7)

41.3.1.17.1 Offset

For a = 0 to 7:

Register	Offset
SHIFTBUFBISa	280h + (a × 4h)

41.3.1.17.2 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

41.3.1.17.3 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].

41.3.1.18 Shifter Buffer N Byte Swapped Register (SHIFTBUFBYS0 - SHIFTBUFBYS7)

41.3.1.18.1 Offset

For a = 0 to 7:

Register	Offset
SHIFTBUFBYSa	300h + (a × 4h)

41.3.1.18.2 Diagram

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

41.3.1.18.3 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] }.

41.3.1.19 Shifter Buffer N Bit Byte Swapped Register (SHIFTBUFBBS0 - SHIFTBUFBBS7)

41.3.1.19.1 Offset

For a = 0 to 7:

Register	Offset
SHIFTBUFBBSa	380h + (a × 4h)

41.3.1.19.2 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

41.3.1.19.3 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] }.

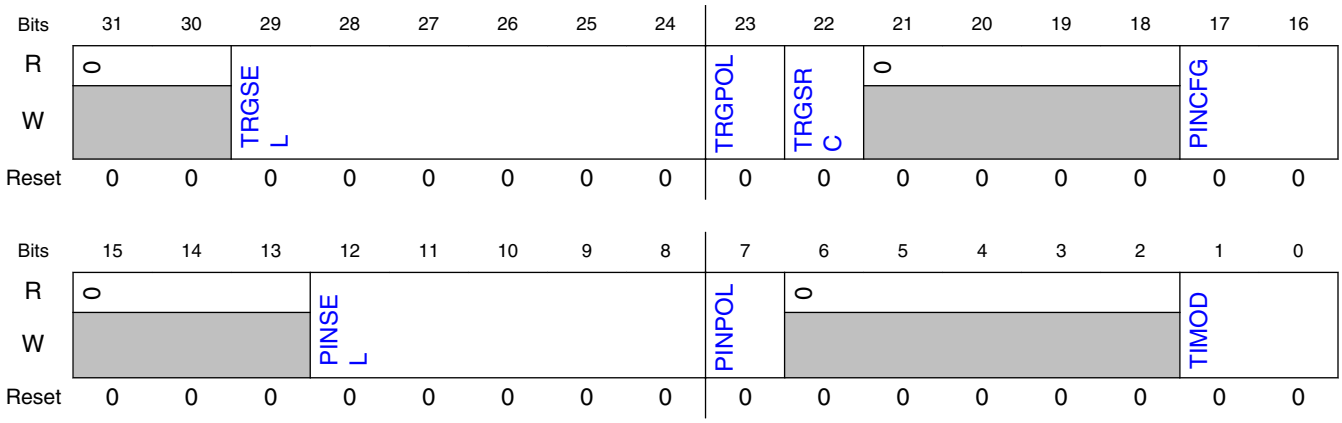
41.3.1.20 Timer Control N Register (TIMCTL0 - TIMCTL7)

41.3.1.20.1 Offset

For a = 0 to 7:

Register	Offset
TIMCTLa	400h + (a × 4h)

41.3.1.20.2 Diagram



41.3.1.20.3 Fields

Field	Function
31-30 —	Reserved
29-24 TRGSEL	Trigger Select The valid values for TRGSEL will depend on the FLEXIO_PARAM register. • When TRGSRC = 1, the valid values for N will depend on PIN, TIMER, SHIFTER fields in the FLEXIO_PARAM register. • When TRGSRC = 0, the valid values for N will depend 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 31. 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 31, Shifter 7 and Timer 7.
23	Trigger Polarity

Table continues on the next page...

Field	Function
TRGPOL	0b - Trigger active high 1b - Trigger active low
22 TRGSRC	Trigger Source 0b - External trigger selected 1b - Internal trigger selected
21-18 —	Reserved
17-16 PINCFG	Timer Pin Configuration Configures the direction of the Timer pin. For pins configured as an output (PINCFG=11), this field will take 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
15-13 —	Reserved
12-8 PINSEL	Timer Pin Select Selects which pin is used by the Timer input or output. PINSEL=i will select the FXIO_Di pin. For pins configured as an output (PINCFG=11), this field will take effect when the register is written.
7 PINPOL	Timer Pin Polarity For pins configured as an output (PINCFG=11), this field will take effect when the register is written. 0b - Pin is active high 1b - Pin is active low
6-2 —	Reserved
1-0 TIMOD	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. 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.

41.3.1.21 Timer Configuration N Register (TIMCFG0 - TIMCFG7)

41.3.1.21.1 Offset

For a = 0 to 7:

Register	Offset
TIMCFGa	480h + (a × 4h)

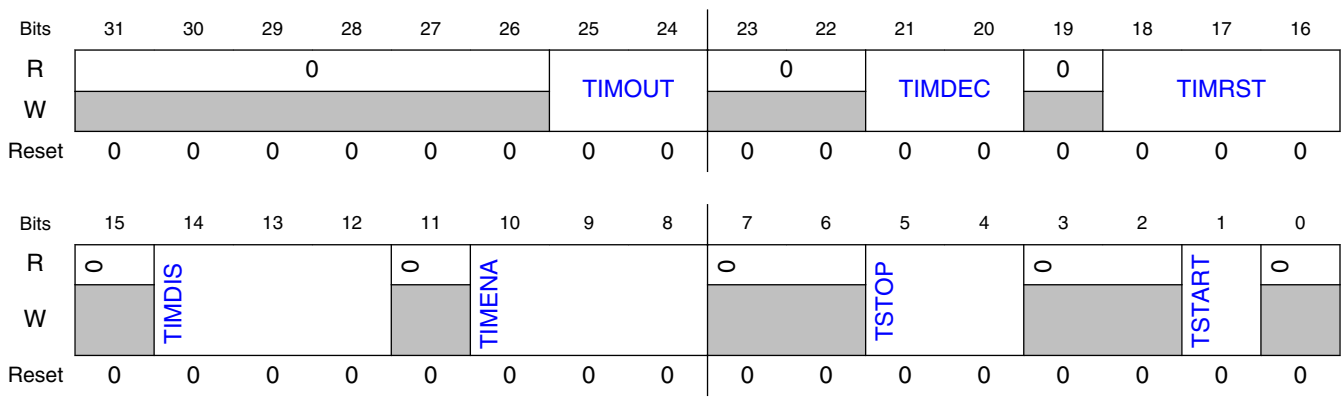
41.3.1.21.2 Function

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).

41.3.1.21.3 Diagram



41.3.1.21.4 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

Table continues on the next page...

Field	Function
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 will only reset the lower 8-bits that configure the baud rate. In all other modes, the timer reset will reset 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
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 will output the contents of the stop bit when the timer is disabled. When

Table continues on the next page...

Field	Function
	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 will output the contents of the start bit when the timer is enabled and the timer counter will reload from the compare register on the first rising edge of the shift clock. 0b - Start bit disabled 1b - Start bit enabled
0 —	Reserved

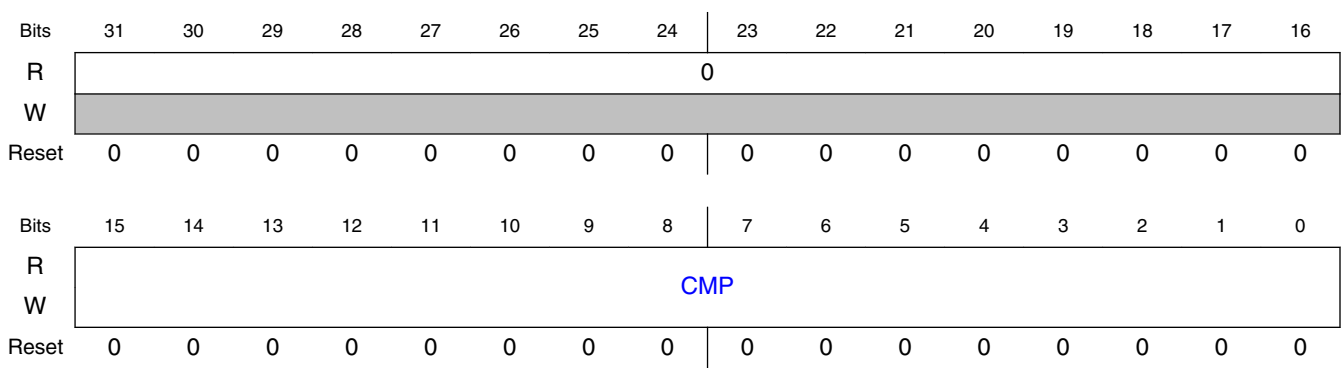
41.3.1.22 Timer Compare N Register (TIMCMP0 - TIMCMP7)

41.3.1.22.1 Offset

For a = 0 to 7:

Register	Offset
TIMCMPa	500h + (a × 4h)

41.3.1.22.2 Diagram



41.3.1.22.3 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.

41.3.1.23 Shifter Buffer N Nibble Byte Swapped Register (SHIFTBUFNBS0 - SHIFTBUFNBS7)

41.3.1.23.1 Offset

For a = 0 to 7:

Register	Offset
SHIFTBUFNBSa	680h + (a × 4h)

41.3.1.23.2 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

41.3.1.23.3 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] }.

41.3.1.24 Shifter Buffer N Half Word Swapped Register (SHIFTBUF HWS0 - SHIFTBUFHWS7)

41.3.1.24.1 Offset

For a = 0 to 7:

Register	Offset
SHIFTBUFHWSa	700h + (a × 4h)

41.3.1.24.2 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

41.3.1.24.3 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] }.

41.3.1.25 Shifter Buffer N Nibble Swapped Register (SHIFTBUFNIS0 - SHIFTBUFNIS7)

41.3.1.25.1 Offset

For a = 0 to 7:

Register	Offset
SHIFTBUFNISa	780h + (a × 4h)

41.3.1.25.2 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

41.3.1.25.3 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] }.

41.4 Functional description

41.4.1 Clocking and Resets

41.4.1.1 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.

41.4.1.2 Bus clock

The bus clock is only used for bus accesses to the control and configuration registers.

41.4.1.3 Chip reset

The logic and registers for the FlexIO are reset to their default state on a chip reset.

41.4.1.4 Software reset

The FlexIO implements a software reset bit in its Control Register. The CTRL[RST] will reset all logic and registers to their default state, except for the CTRL itself.

41.4.2 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.

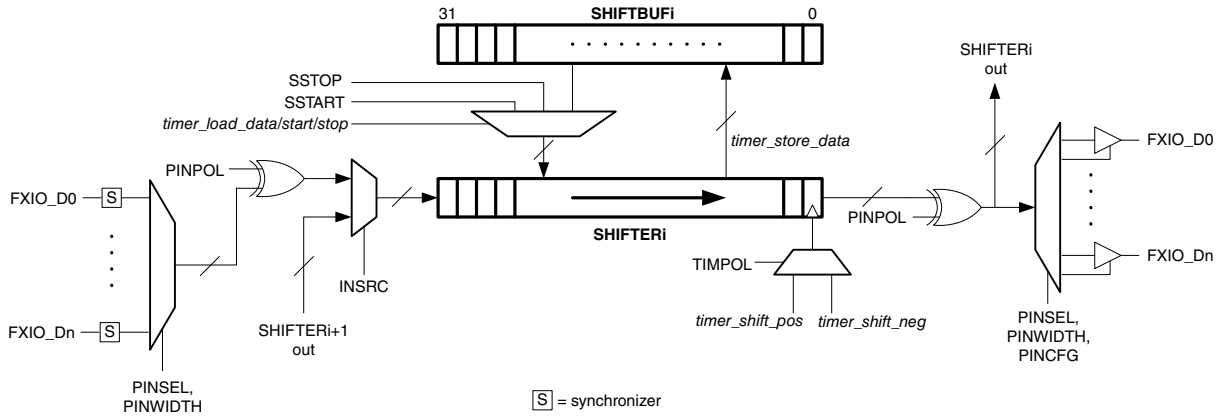


Figure 41-2. Shifter Microarchitecture

41.4.2.1 Transmit Mode

When configured for Transmit mode (SHIFTCTL[SMOD]=Transmit), the shifter will load 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/after SHIFTBUF data by configuring the SHIFTCFG[SSTART], TIMCFG[TSTART] or SHIFTCFG[SSTOP], TIMCFG[TSTOP] registers in the Shifter and Timer. Note that the shifter will immediately load 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 will 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 will clear when new data has been written into the SHIFTBUF register.

The Shifter Error Flag (SHIFTERR[SEF]) and any enabled interrupts will 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.

41.4.2.2 Receive Mode

When configured for Receive mode (SHIFTCTL[SMOD]=Receive), the shifter will shift 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 will set when data has been stored into the SHIFTBUF register from the Shifter. The flag will clear when the data has been read from the SHIFTBUF register.

The Shifter Error Flag (SHIFTERR[SEF]) and any enabled interrupts will set when an attempt to store 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.

41.4.2.3 Match Store Mode

When configured for Match Store mode (SHIFTCTL[SMOD]=Match Store), the shifter will shift 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 will set when a match occurs and matched data has been stored into the SHIFTBUF register from the Shifter. The flag will clear when the matched data has been read from the SHIFTBUF register.

The Shifter Error Flag (SHIFTERR[SEF]) and any enabled interrupts will 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.

41.4.2.4 Match Continuous Mode

When configured for Match Continuous mode (SHIFTCTL[SMOD]=Match Continuous), the shifter will shift 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 will set when a match occurs. The flag will clear automatically as soon as there is no longer a match between Shifter data and SHIFTBUF register. The flag cannot be cleared by reading SHIFTBUF register.

The Shifter Error Flag (SHIFTERR[SEF]) and any enabled interrupts will set when a match occurs. The flag will clear when there is a read from the SHIFTBUF register or it written with logic 1.

41.4.2.5 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 STOP/VLPS 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.

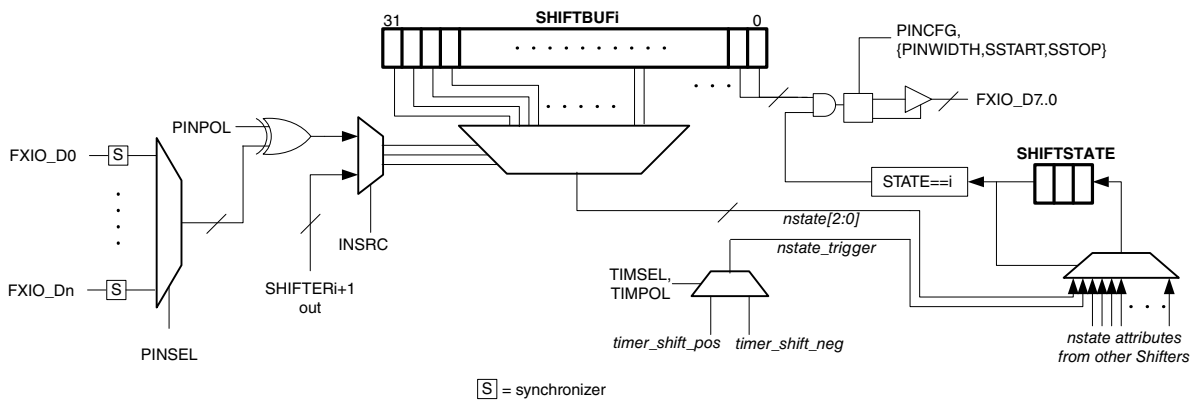


Figure 41-3. State Microarchitecture

When Shifter i has been selected by the current state pointer, output pins FXIO_D[7:0] will be 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 that 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 41-3. Next State computation for SHIFTSTATE[STATE]=i

FXIO_D[PINSEL+2]	FXIO_D[PINSEL+1]	FXIO_D[PINSEL]	Next State Value

Table continues on the next page...

Table 41-3. Next State computation for SHIFTSTATE[STATE]=i (continued)

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 will not be driven and a next state transition is never triggered.

The Shifter Status Flag (SHIFTSTAT[SSF]) and any enabled interrupts or DMA requests will set whenever the Shifter has been selected by the current state pointer. The flag will clear when the current state pointer is updated to a different Shifter.

41.4.2.6 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), the SHIFTBUF 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.

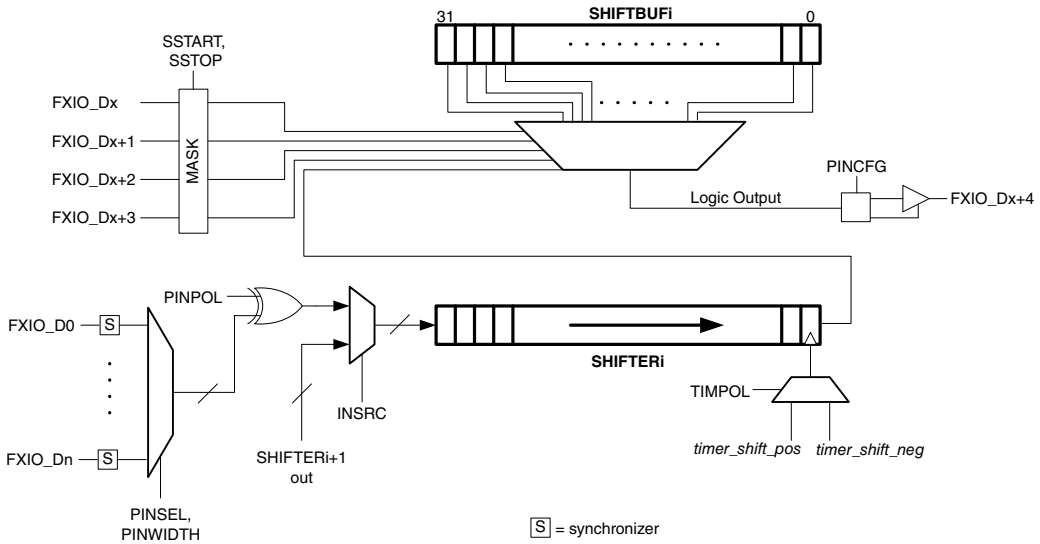


Figure 41-4. Logic Microarchitecture

The look-up table is driven using 4 pin inputs (maskable using SHIFTCFG[SSTOP] and SHFITCFG[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 41-4. 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]} will mask FXIO_D[x+3]...FXIO_D[x] inputs respectively, so that any transitions on these pins will 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] will control 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 will introduce 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 will introduce a 1 Shift clock delay to its input.

The Shifter Status Flag (SHIFTSTAT[SSF]) and any enabled interrupts or DMA requests will 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 will clear 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 will set when the output pin allocated to the logic look-up table has a value of 1. The flag can be cleared by writing it with logic 1.

41.4.3 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).

41.4.3.1 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 will only reset 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.

41.4.3.2 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.

41.4.3.3 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.

41.4.3.4 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 will load 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 will 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 will reload 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 will not load correctly.

41.4.3.5 Timer Decrement and Reset

The Timer will then generate 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 will decrement on both rising and falling edges.

When the Timer is configured to reset as configured in the TIMRST field then the Timer counter will load 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 will cause 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.

41.4.3.6 Timer Disable and Stop Bit

When the is Timer is configured to add a stop bit on each compare, the following additional events will occur.

- Transmit shifters controlled by this timer will output their stop bit value (if configured by SSTOP).
- Receive shifters controlled by this timer will 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 will reload 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 will reload the current value of the Compare Register and start decrementing as configured by TIMDEC.
- Timer output will clear. 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 will output their stop bit value (if configured by SSTOP).
- Receive shifters controlled by this timer will 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 will continue decrementing until the next rising edge of the shift clock is detected, at which point it will finish. 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 will 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.

41.4.4 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.

41.4.4.1 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 will be 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 will be masked with 0 e.g. if PINSEL=7 and PWIDTH=6, then SHIFTER[31:24] will sample {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 will not be driven e.g. if PINSEL=7 and PWIDTH=6, then SHIFTER[5:0] will drive only FXIO_D[12:7] on each Shift clock.

41.4.4.2 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 will make 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.

41.4.5 Interrupts and DMA Requests

The following table illustrates the status flags that can generate the FlexIO interrupt and DMA requests.

Table 41-5. 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

41.4.6 Peripheral Triggers

The connection of the FlexIO peripheral triggers with other peripherals are device specific.

41.4.6.1 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.

41.4.6.2 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.

41.5 Application Information

This section provides examples for a variety of FlexIO module applications.

41.5.1 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 41-6. 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

Table continues on the next page...

Table 41-6. UART Transmit Configuration (continued)

Register	Value	Comments
		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.

41.5.2 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 will cause the error flag to set and the shifter buffer register will return 0x00.

FlexIO does not support automatic verification of parity bits.

Table 41-7. 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.

Table continues on the next page...

Table 41-7. UART Receiver Configuration (continued)

Register	Value	Comments
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 will assert when the start bit is detected and negate when the data is read from the shifter buffer register. No start bit will be detected while the RTS is asserted, the received data is simply ignored.

Table 41-8. 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.
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.

Table continues on the next page...

Table 41-8. UART Receiver with RTS Configuration (continued)

Register	Value	Comments
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.

41.5.3 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.

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 41-9. 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.
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

Table continues on the next page...

Table 41-9. SPI Master (CPHA=0) Configuration (continued)

Register	Value	Comments
		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 41-10. 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. 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.

Table continues on the next page...

Table 41-10. SPI Master (CPHA=1) Configuration (continued)

Register	Value	Comments
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.

41.5.4 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 will be set.

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 41-11. 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.

Table continues on the next page...

Table 41-11. SPI Slave (CPHA=0) Configuration (continued)

Register	Value	Comments
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 41-12. 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.
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.

Table continues on the next page...

Table 41-12. SPI Slave (CPHA=1) Configuration (continued)

Register	Value	Comments
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.

41.5.5 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 will set 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 will load 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 will assert 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.

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 41-13. 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.
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.

41.5.6 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 will set 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.

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 41-14. 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 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_0201	Configure dual 8-bit counter using Pin 2 output (bit clock), with Shifter 0 flag as the inverted trigger. Set PINPOL to invert 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.

Table continues on the next page...

Table 41-14. I2S Master Configuration (continued)

Register	Value	Comments
TIMCTL(n+1)	0x0003_0383	Configure 16-bit counter using inverted Pin 3 output (as 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.

41.5.7 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 will be set.

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 falling edge of bit clock (when frame sync is normally sampled). Timer 0 detects falling edge of bit clock with Timer 2 output asserted and asserts output for length of frame. Timer 1 detects rising edge of bit clock with Timer 0 output asserted and controls shift registers for 32-bit transfers.

Table 41-15. I2S Slave Configuration

Register	Value	Comments
SHIFTCFGn	0x0000_0000	Start and stop bit disabled.

Table continues on the next page...

Table 41-15. I2S Slave Configuration (continued)

Register	Value	Comments
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_0283	Configure 16-bit counter using inverted 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 falling edge (bit clock), initial clock state is logic 1 and decrement on inverted pin input (frame sync).
TIMCTL(n+2)	0x0440_0383	Configure 16-bit counter using inverted Pin 3 input (frame sync), with Pin 2 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.

41.5.8 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 41-16. 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 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

41.5.9 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 41-17. 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.
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 41-18. 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.

Table continues on the next page...

Table 41-18. Motorola 68K/Intel 8080 Read Configuration (continued)

Register	Value	Comments
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 will begin 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

41.5.10 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.

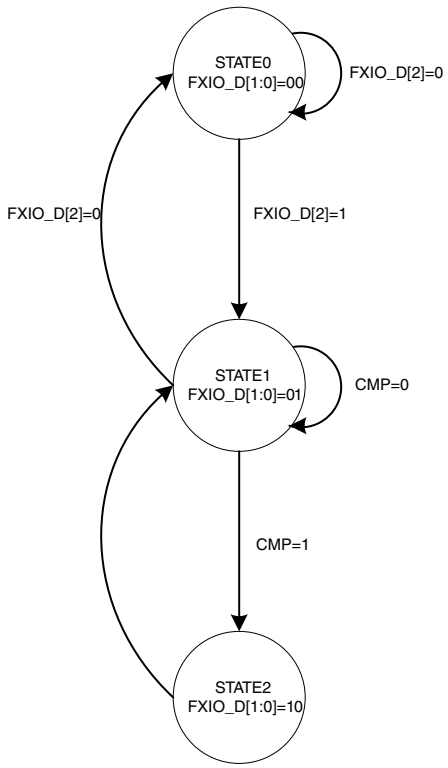


Figure 41-5. State Diagram

Table 41-19. 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.

Table continues on the next page...

Table 41-19. State Machine Configuration (continued)

Register	Value	Comments
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.

Chapter 42
Timers Overview

42.1 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.
- 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 (RTWDOG/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

42.1.1 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 has a 12-bit pre-scaler, which provides a programmable clock frequency derived from multiple clock sources.

42.1.2 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 a slave mode to synchronize count enable for multiple PITs.

42.1.3 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.

42.1.4 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.

42.1.5 Watchdog Timer (RTWDOG /WDOG3)

The WDOG3 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 MCU.

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

42.1.6 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.

The EWM 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

Chapter 43
General Purpose Timer (GPT)

43.1 Chip-specific GPT 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 Overview

This chapter describes the General Purpose Timer (GPT) module interface. It is also a reference for software driver programming. 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 can also generate an event on the output compare pins and an interrupt when the timer reaches a programmed value. The GPT has a 12-bit prescaler, which provides a programmable clock frequency derived from multiple clock sources.

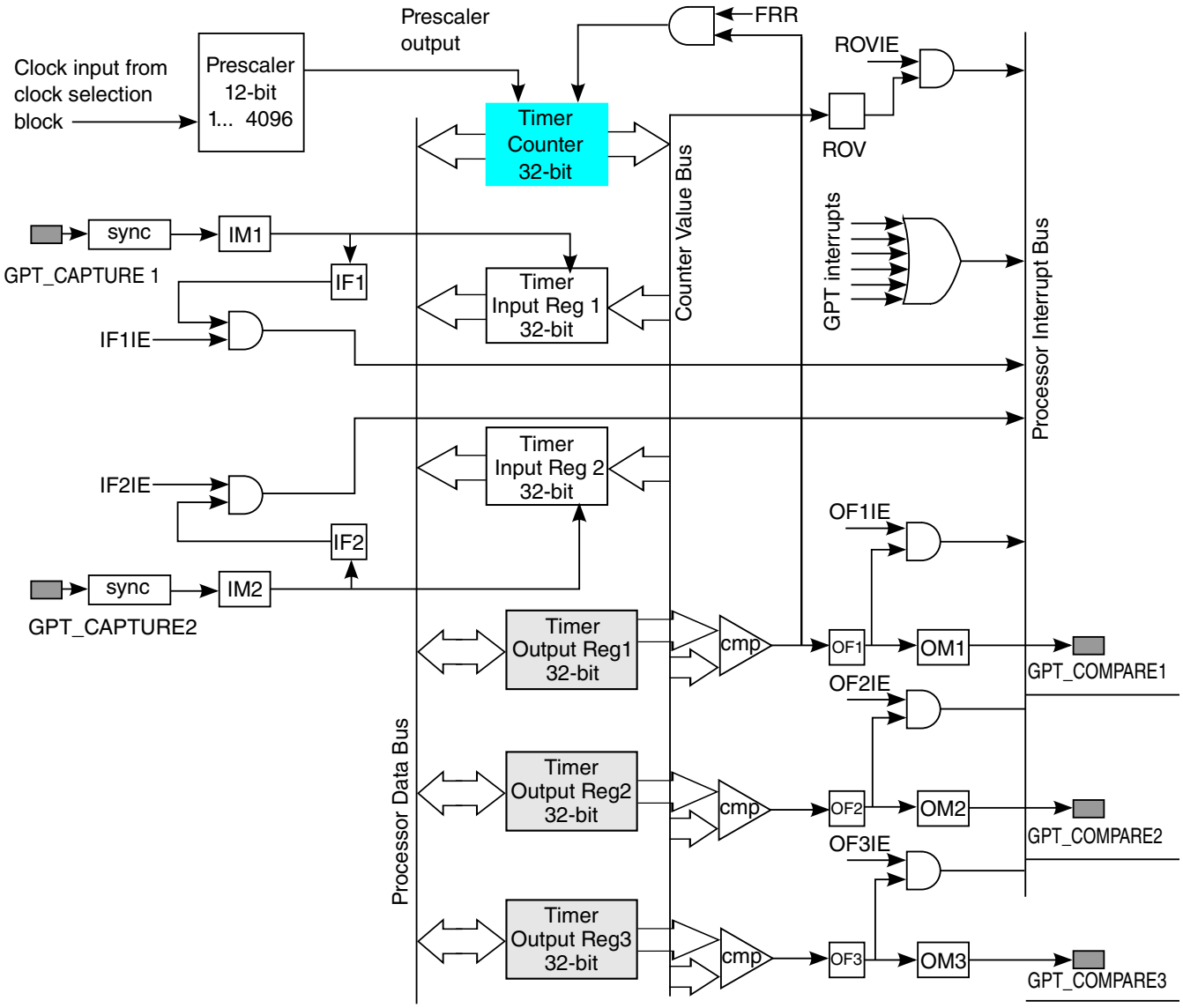


Figure 43-1. GPT Block Diagram

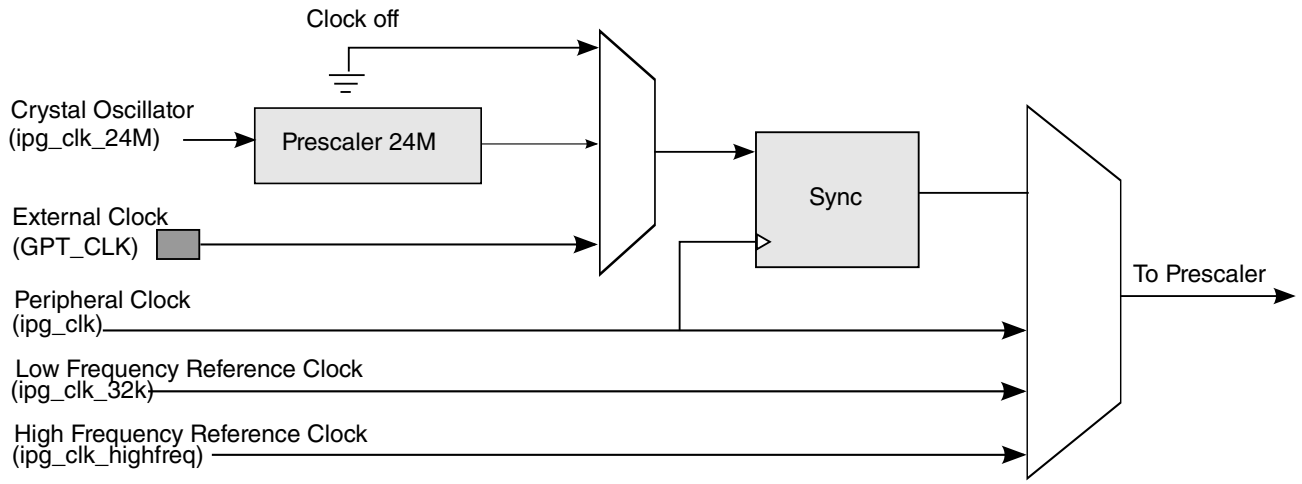


Figure 43-2. GPT Counter Clocks Diagram

43.2.1 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.

43.2.2 Modes and Operation

The GPT supports the modes described in the indicated sections:

- Operating Modes
  - Restart Mode
  - Free-Run Mode

43.3 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.

Table 43-2. GPT External Signals

Name	Description	Pad	Mode	Direction
GPT1_CLK	Input pin for an external clock that the counter can be operated at.	GPIO_08	ALT1	I
GPT1_CAPTURE1	Input pin for a capture event for Input Capture Channel 1	GPIO_06	ALT1	I
GPT1_CAPTURE2	Input pin for a capture event for Input Capture Channel 2	GPIO_04	ALT1	I
GPT1_COMPARE1	Output pin that indicates a "compare event" occurrence in Output Compare Channel 1	GPIO_07	ALT1	O
GPT1_COMPARE2	Output pin that indicates a "compare event" occurrence in Output Compare Channel 2	GPIO_05	ALT1	O
GPT1_COMPARE3	Output pin that indicates a "compare event" occurrence in Output Compare Channel 3	GPIO_03	ALT1	O
GPT2_CAPTURE1	Input pin for a capture event for Input Capture Channel 1	GPIO_AD_05	ALT4	I
GPT2_CAPTURE2	Input pin for a capture event for Input Capture Channel 2	GPIO_AD_07	ALT4	I
GPT2_COMPARE1	Output pin that indicates a "compare event" occurrence in Output Compare Channel 1	GPIO_AD_04	ALT4	O
GPT2_COMPARE2	Output pin that indicates a "compare event" occurrence in	GPIO_AD_06	ALT4	O

Table continues on the next page...

Table 43-2. GPT External Signals (continued)

Name	Description	Pad	Mode	Direction
	Output Compare Channel 2			
GPT2_COMPARE3	Output pin that indicates a "compare event" occurrence in Output Compare Channel 3	GPIO_AD_08	ALT4	O
GPT2_CLK	Input pin for an external clock that the counter can be operated at.	GPIO_AD_05	ALT4	I

There are six signals (three input, three output) in the GPT module that *can be* connected to the chip pads.

43.3.1 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.

43.3.2 Input Capture Trigger Signals

The GPT counter value can be stored in a register, triggered by an event from *outside the device*. A positive or/and negative edge on these signals GPT_CAPTURE1 , GPT_CAPTURE2 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.

43.3.3 Output Compare Signals

The output compare signals: GPT_COMPARE1, GPT_COMPARE2, GPT_COMPARE3, indicate that output compare events have gone through a specified transition.

43.4 Clocks

The clock that is input to the prescaler can be selected from 4 clock sources. The following table describes the clock sources for GPT. Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 43-3. GPT Clocks

Clock name	Clock Root	Description
ipg_clk	ipg_clk_root	Peripheral clock
ipg_clk_32k	ckil_sync_clk_root	Low-frequency reference clock (32 kHz)
ipg_clk_highfreq	perclk_clk_root	High-frequency reference clock
ipg_clk_s	ipg_clk_root	Peripheral access clock

- High-Frequency Clock (ipg_clk_highfreq)

  Provided by the Clock Controller Module (CCM), the High Frequency 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*. The CCM is expected to provide this clock *after* synchronizing it to the System Bus Clock (ahb_clk) in Normal functional mode; the CCM is also expected to switch to the *unsynchronized* version of the High Frequency Clock in a Low Power mode.

- Low-Reference Clock (ipg_clk_32k)

  This 32 kHz Low 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. The CCM is expected to provide the Low Reference Clock *after* synchronizing it to the System Bus Clock (ahb_clk) in Normal functional mode; the CCM is also expected to switch to the *unsynchronized* version of the Low Reference Clock in a Low Power mode.

- Peripheral Clock (ipg_clk)

  If the Peripheral Clock (ipg_clk) or the External Clock (GPT_CLK) is selected (CLKSRC=001 or 011) as Clock Source, 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.

- External Clock (GPT_CLK)

The External Clock comes from *outside the device* and can be selected to run the GPT counter. The External Clock is treated as *asynchronous to the Peripheral Clock,* (ipg_clk) and is synchronized to the Peripheral Clock (ipg_clk), *inside* the module. Therefore, 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.

- Crystal Oscillator Clock (ipg_clk_24M)

  This 24 MHz Crystal Oscillator Clock (provided by the CCM) is intended to be used against frequency change of Peripheral Clock (ipg_clk) changes to provide a more accurate timer clock for operation system. The CCM is expected to provide the 24 MHz Crystal 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 Crystal Oscillator Clock is divided by a 24 MHz clock prescaler, to make sure the clock frequency less than half of System Bus Clock (ahb_clk).

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.

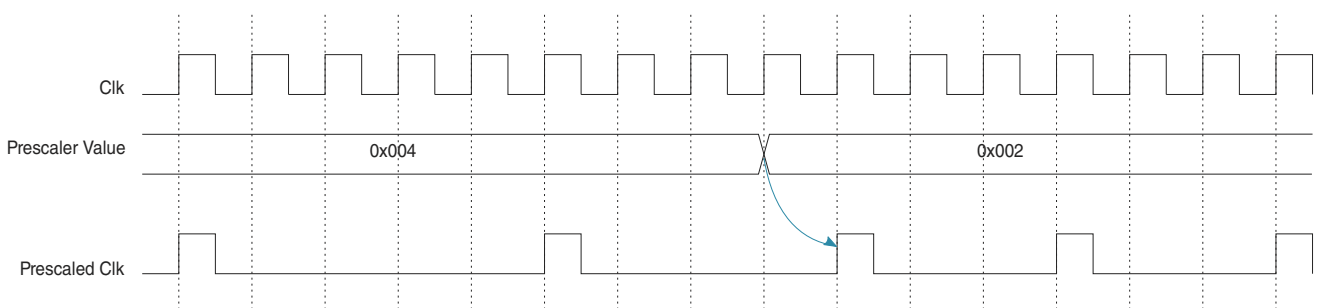


Figure 43-3. Prescaler Value Change Timing Diagram

43.5 Functional Description

This section provides a complete functional description of the GPT.

43.5.1 Operating Modes

The GPT counter can be programmed to work in either of two modes: Restart mode or Free-Run mode.

43.5.1.1 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*.

43.5.1.2 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).

43.5.2 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 43-1.

- 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*.

43.5.2.1 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.

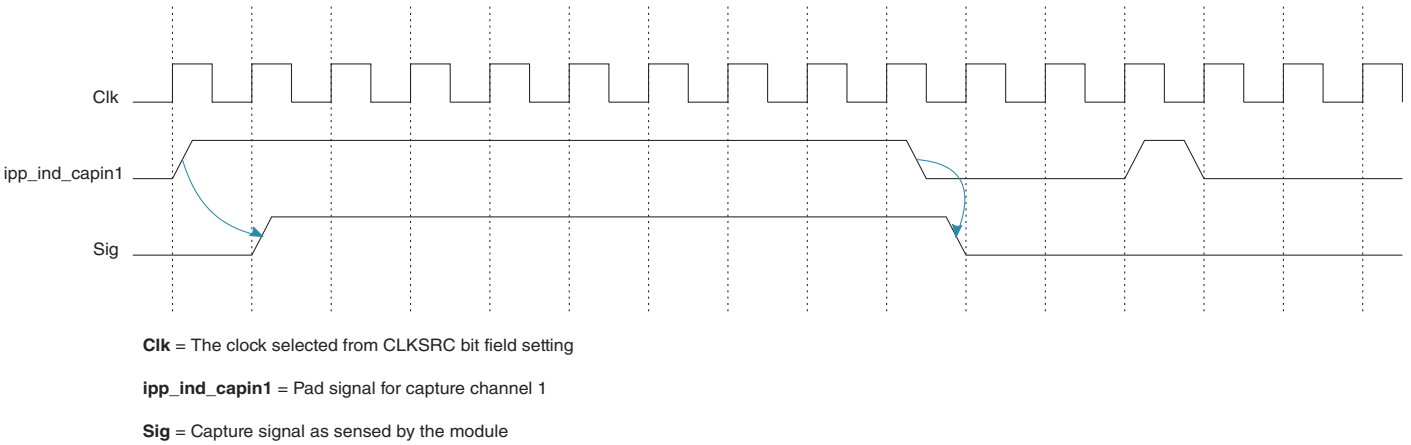


Figure 43-4. Input Capture Event Timing

43.5.2.2 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 will be set, cleared, toggled, not affected at all or provide an active-low pulse for one input clock period (subject to the restriction on the maximum frequency allowed on the pad) according to the mode bits (that were programmed).

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.

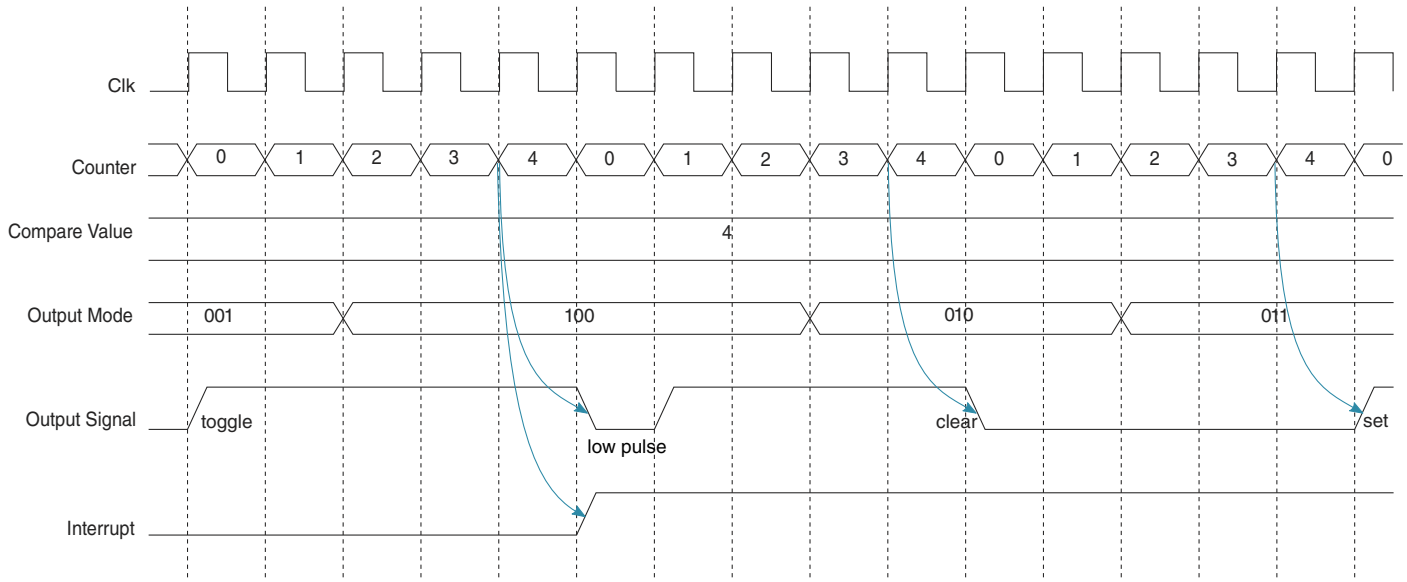


Figure 43-5. Output Compare and Interrupt Timing

43.5.2.3 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.

43.5.2.4 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.

43.5.2.5 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.

43.6 Initialization/ Application Information

43.6.1 Selecting the Clock Source

The CLKSRC field in the GPT_CR register selects the clock source. The CLKSRC field value should be changed only after disabling the GPT (EN=0).

The software sequence to be followed while changing clock source is:

1. Disable GPT by setting EN=0 in GPT_CR register.
2. Disable GPT interrupt register (GPT_IR).

3. Configure Output Mode to unconnected/ disconnected—Write zeros in OM3, OM2, and OM1 in GPT_CR
4. Disable Input Capture Modes—Write zeros in IM1 and IM2 in GPT_CR
5. Change clock source CLKSRC to the desired value in GPT_CR register.
6. Assert the SWR bit in GPT_CR register.
7. Clear GPT status register (GPT_SR) (i.e., w1c).
8. Set ENMOD=1 in GPT_CR register, to bring GPT counter to 0x00000000.
9. Enable GPT (EN=1) in GPT_CR register.
10. Enable GPT interrupt register (GPT_IR).

43.7 GPT Memory Map/Register Definition

The GPT has 10 user-accessible 32-bit registers, which are used to configure, operate, and monitor the state of the GPT.

An IP bus write access to the GPT Control Register (GPT_CR) and the GPT Output Compare Register1 (GPT_OCR1) results in *one cycle of wait state*, while other valid IP bus accesses incur 0 wait states.

Irrespective of the Response Select signal value, a Write access to the GPT Status Registers (Read-only registers GPT_ICR1, GPT_ICR2, GPT_CNT) will generate a bus exception.

- If the Response Select signal is driven Low, then the Read/Write access to the *unimplemented* address space of GPT (*ips_addr* is greater than or equal to $BASE + $028) will generate a bus exception.
- If the Response Select is driven High, then the Read/Write access to the unimplemented address space of GPT will *not* generate any error response (like a bus exception).

GPT memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
401E_C000	GPT Control Register (GPT1_CR)	32	R/W	0000_0000h	43.7.1/1657
401E_C004	GPT Prescaler Register (GPT1_PR)	32	R/W	0000_0000h	43.7.2/1661
401E_C008	GPT Status Register (GPT1_SR)	32	R/W	0000_0000h	43.7.3/1662
401E_C00C	GPT Interrupt Register (GPT1_IR)	32	R/W	0000_0000h	43.7.4/1663
401E_C010	GPT Output Compare Register 1 (GPT1_OCR1)	32	R/W	FFFF_FFFFh	43.7.5/1664
401E_C014	GPT Output Compare Register 2 (GPT1_OCR2)	32	R/W	FFFF_FFFFh	43.7.6/1665

Table continues on the next page...

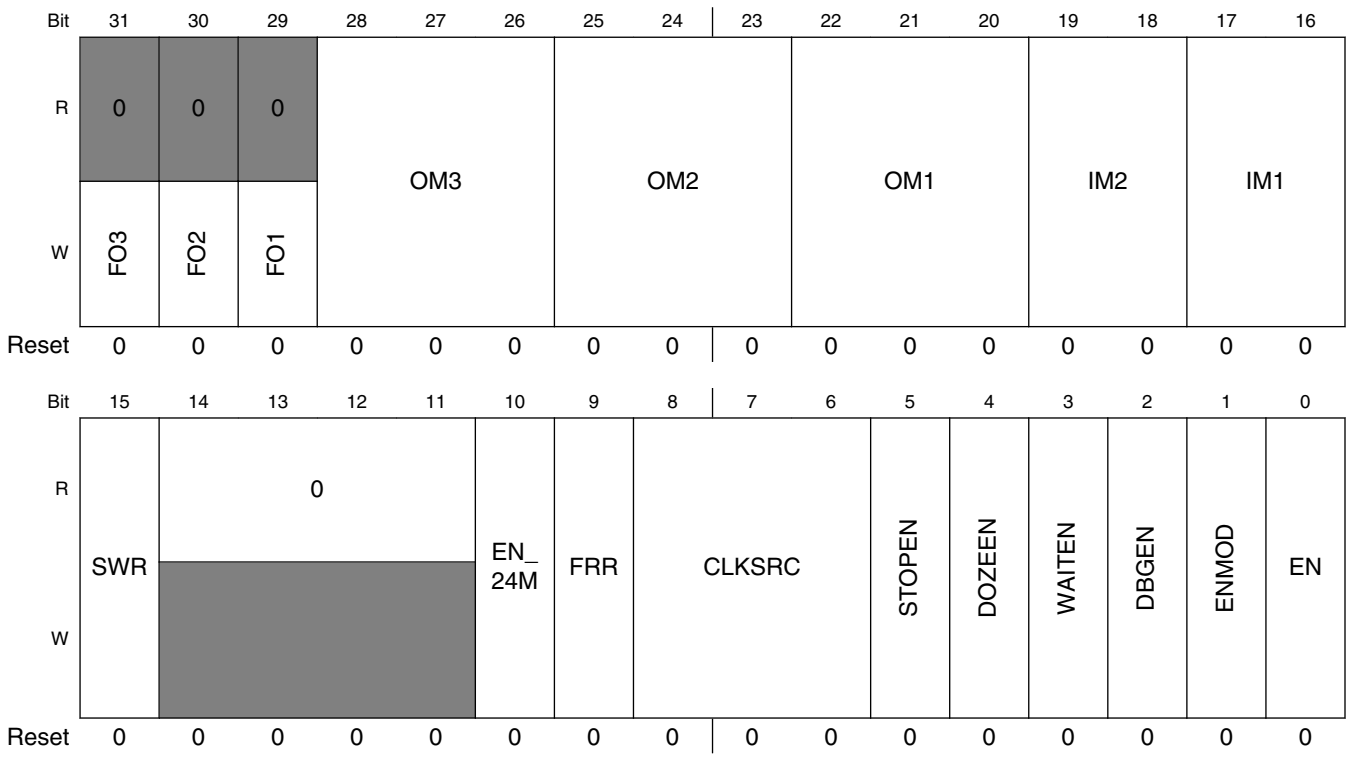
GPT memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
401E_C018	GPT Output Compare Register 3 (GPT1_OCR3)	32	R/W	FFFF_FFFFh	43.7.7/1665
401E_C01C	GPT Input Capture Register 1 (GPT1_ICR1)	32	R	0000_0000h	43.7.8/1666
401E_C020	GPT Input Capture Register 2 (GPT1_ICR2)	32	R	0000_0000h	43.7.9/1666
401E_C024	GPT Counter Register (GPT1_CNT)	32	R	0000_0000h	43.7.10/ 1667
401F_0000	GPT Control Register (GPT2_CR)	32	R/W	0000_0000h	43.7.1/1657
401F_0004	GPT Prescaler Register (GPT2_PR)	32	R/W	0000_0000h	43.7.2/1661
401F_0008	GPT Status Register (GPT2_SR)	32	R/W	0000_0000h	43.7.3/1662
401F_000C	GPT Interrupt Register (GPT2_IR)	32	R/W	0000_0000h	43.7.4/1663
401F_0010	GPT Output Compare Register 1 (GPT2_OCR1)	32	R/W	FFFF_FFFFh	43.7.5/1664
401F_0014	GPT Output Compare Register 2 (GPT2_OCR2)	32	R/W	FFFF_FFFFh	43.7.6/1665
401F_0018	GPT Output Compare Register 3 (GPT2_OCR3)	32	R/W	FFFF_FFFFh	43.7.7/1665
401F_001C	GPT Input Capture Register 1 (GPT2_ICR1)	32	R	0000_0000h	43.7.8/1666
401F_0020	GPT Input Capture Register 2 (GPT2_ICR2)	32	R	0000_0000h	43.7.9/1666
401F_0024	GPT Counter Register (GPT2_CNT)	32	R	0000_0000h	43.7.10/ 1667

43.7.1 GPT Control Register (GPTx_CR)

The GPT Control Register (GPT_CR) is used to program and configure GPT operations. An IP Bus Write to the GPT Control Register occurs after one cycle of wait state, while an IP Bus Read occurs after 0 wait states.

Address: Base address + 0h offset



GPTx_CR field descriptions

Field	Description
31 FO3	FO3 Force Output Compare Channel 3 FO2 Force Output Compare Channel 2 FO1 Force Output Compare Channel 1 The FO*n* bit causes the pin action *programmed* for the timer Output Compare *n* pin (according to the OM*n* bits in this register). • The OF*n* flag (OF3, OF2, OF1) in the status register is **not affected**. • This bit is self-negating and always read as zero. 0 Writing a 0 has no effect. 1 Causes the programmed pin action on the timer Output Compare *n* pin; the OF*n* flag is not set.
30 FO2	See F03

Table continues on the next page...

GPTx_CR field descriptions (continued)

Field	Description
29 FO1	See F03
28–26 OM3	OM3 (bits 28-26) controls the Output Compare Channel 3 operating mode. OM2 (bits 25-23) controls the Output Compare Channel 2 operating mode. OM1 (bits 22-20) controls the Output Compare Channel 1 operating mode. The OM*n* bits specify the response that a compare event will generate on the output pin of Output Compare Channel *n*. • The toggle, clear, and set options cause a change on the output pin *only* if a compare event occurs. • When OM*n* is programmed as 1xx (active low pulse), the output pin is set to one immediately on the next input clock; a low pulse (that is an input clock in width) occurs when there is a compare event. Note that here, "input clock" refers to the clock selected by the CLKSRC bits of the GPT Control Register. 000 Output disconnected. No response on pin. 001 Toggle output pin 010 Clear output pin 011 Set output pin 1xx Generate an active low pulse (that is one input clock wide) on the output pin.
25–23 OM2	See OM3
22–20 OM1	See OM3
19–18 IM2	IM2 (bits 19-18, Input Capture Channel 2 operating mode) IM1 (bits 17-16, Input Capture Channel 1 operating mode) The IM*n* bit field determines the transition on the input pin (for Input capture channel *n*), which will trigger a capture event. 00 capture disabled 01 capture on rising edge only 10 capture on falling edge only 11 capture on both edges
17–16 IM1	See IM2
15 SWR	Software reset. This is the software reset of the GPT module. It is a self-clearing bit. • The SWR bit is set when the module is in reset state. • The SWR bit is cleared when the reset procedure finishes. • Setting the SWR bit resets **all of the registers** to their default reset values, except for the EN, ENMOD, STOPEN, DOZEEN, WAITEN, and DBGEN bits in the GPT Control Register (this control register). 0 GPT is not in reset state 1 GPT is in reset state
14–11 Reserved	This read-only field is reserved and always has the value 0.
10 EN_24M	Enable 24 MHz clock input from crystal.

Table continues on the next page...

GPTx_CR field descriptions (continued)

Field	Description
	• A hardware reset resets the EN_24M bit. • A software reset *does not affect* the EN_24M bit. 0 24M clock disabled 1 24M clock enabled
9 FRR	Free-Run or Restart mode. The FFR bit determines the behavior of the GPT when a compare event in channel 1 occurs. • In Restart mode, after a compare event, the counter resets to 0x00000000 and resumes counting (after the occurrence of a compare event). • In Free-Run mode, after a compare event, the counter continues counting until 0xFFFFFFFF and then rolls over to 0. 0 Restart mode 1 Free-Run mode
8–6 CLKSRC	Clock Source select. The CLKSRC bits select which clock will go to the prescaler (and subsequently be used to run the GPT counter). • The CLKSRC bit field value should only be changed after disabling the GPT by clearing the EN bit in this register (GPT_CR). • A software reset does not affect the CLKSRC bit. 000 No clock 001 Peripheral Clock (ipg_clk) 010 High Frequency Reference Clock (ipg_clk_highfreq) 011 External Clock 100 Low Frequency Reference Clock (ipg_clk_32k) 101 Crystal oscillator as Reference Clock (ipg_clk_24M) others Reserved
5 STOPEN	GPT Stop Mode enable. The STOPEN read/write control bit enables GPT operation *during Stop mode*. • A hardware reset resets the STOPEN bit. • A software reset *does not affect* the STOPEN bit. 0 GPT is disabled in Stop mode. 1 GPT is enabled in Stop mode.
4 DOZEEN	GPT Doze Mode Enable. • A hardware reset resets the DOZEEN bit. • A software reset *does not affect* the DOZEEN bit. 0 GPT is disabled in doze mode. 1 GPT is enabled in doze mode.
3 WAITEN	GPT Wait Mode enable. The WAITEN read/write control bit enables GPT operation *during Wait mode*. • A hardware reset resets the WAITEN bit. • A software reset *does not affect* the WAITEN bit. 0 GPT is disabled in wait mode. 1 GPT is enabled in wait mode.

Table continues on the next page...

GPTx_CR field descriptions (continued)

Field	Description
2 DBGEN	GPT debug mode enable. The DBGEN read/write control bit enables GPT operation *during Debug mode*. • A hardware reset resets the DBGEN bit. • A software reset *does not affect* the DBGEN bit. 0 GPT is disabled in debug mode. 1 GPT is enabled in debug mode.
1 ENMOD	GPT Enable mode. When the GPT is disabled (EN=0), then both the Main Counter and Prescaler Counter *freeze their current count values*. The ENMOD bit determines the value of the GPT counter when Counter is enabled again (if the EN bit is set). • If the ENMOD bit is 1, then the Main Counter and Prescaler Counter values are reset to 0 after GPT is enabled (EN=1). • If the ENMOD bit is 0, then the Main Counter and Prescaler Counter restart counting *from their frozen values* after GPT is enabled (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 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. • Setting the SWR bit will clear the Main Counter and Prescaler Counter values, regardless of the value of EN or ENMOD bits. • A hardware reset resets the ENMOD bit. • A software reset *does not affect* the ENMOD bit. 0 GPT counter will retain its value when it is disabled. 1 GPT counter value is reset to 0 when it is disabled.
0 EN	GPT Enable. The EN bit is the GPT module enable bit. **Before setting the EN bit**, we recommend that *all registers be properly programmed*. • A hardware reset resets the EN bit. • A software reset *does not affect* the EN bit. 0 GPT is disabled. 1 GPT is enabled.

43.7.2 GPT Prescaler Register (GPTx_PR)

The GPT Prescaler Register (GPT_PR) contains bits that determine the *divide value* of the clock that runs the counter.

Address: Base address + 4h 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 W	PRESCALER24M				PRESCALER											
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

GPTx_PR field descriptions

Field	Description
31–16 Reserved	This read-only field is reserved and always has the value 0.
15–12 PRESCALER24M	Prescaler bits. 24M crystal clock is divided by [PRESCALER24M + 1] before selected by the CLKSRC field. If 24M crystal clock is not selected, this feild takes no effect. 0x0 Divide by 1 0x1 Divide by 2 ... ... 0xF Divide by 16
PRESCALER	Prescaler bits. The clock selected by the CLKSRC field is divided by [PRESCALER + 1], and then used to run the counter. • A change in the value of the PRESCALER bits cause the Prescaler counter to reset and a new count period to start immediately. • See Figure 43-3 for the timing diagram. 0x000 Divide by 1 0x001 Divide by 2 ... ... 0xFFF Divide by 4096

43.7.3 GPT Status Register (GPTx_SR)

The GPT Status Register (GPT_SR) contains bits that indicate that a counter has rolled over, and if any event has occurred on the Input Capture and Output Compare channels. The bits are cleared by writing a 1 to them.

Address: Base address + 8h 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										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

GPTx_SR field descriptions

Field	Description
31–6 Reserved	This read-only field is reserved and always has the value 0.
5 ROV	Rollover Flag. The ROV bit indicates that the counter has reached its *maximum possible value* and *rolled over* to 0 (from which the counter continues counting). The ROV bit is only set if the counter has reached 0xFFFFFFFF in both Restart and Free-Run modes. 0 Rollover has not occurred. 1 Rollover has occurred.
4 IF2	IF2 Input capture 2 Flag IF1 Input capture 1 Flag The IF*n* bit indicates that a capture event has occurred on Input Capture channel *n*. 0 Capture event has not occurred. 1 Capture event has occurred.
3 IF1	See IF2
2 OF3	OF3 Output Compare 3 Flag OF2 Output Compare 2 Flag OF1 Output Compare 1 Flag The OF*n* bit indicates that a compare event has occurred on Output Compare channel *n*. 0 Compare event has not occurred. 1 Compare event has occurred.

Table continues on the next page...

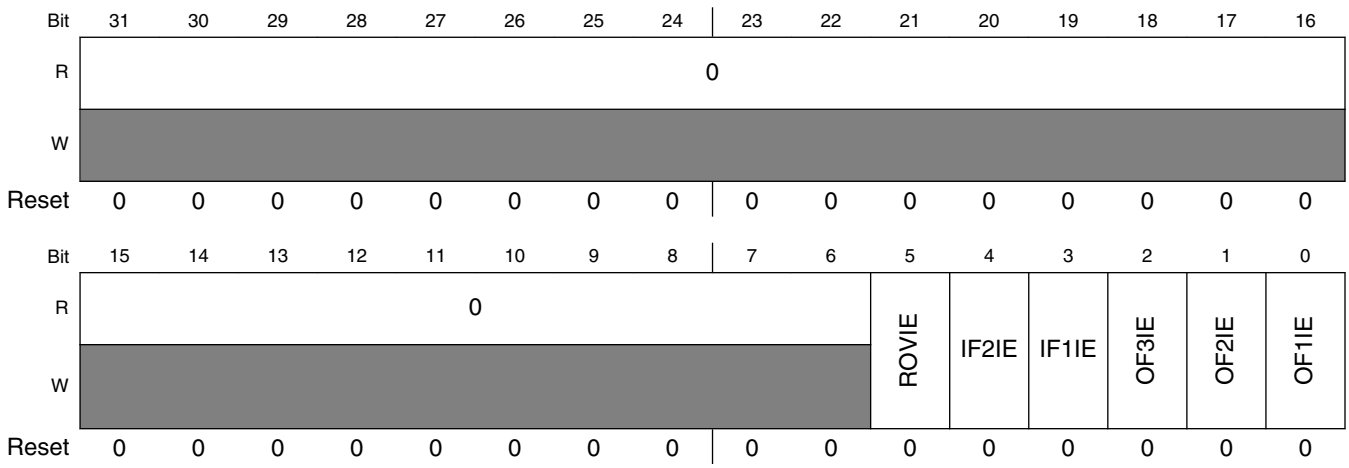
GPTx_SR field descriptions (continued)

Field	Description
1 OF2	See OF3
0 OF1	See OF3

43.7.4 GPT Interrupt Register (GPTx_IR)

The GPT Interrupt Register (GPT_IR) contains bits that control whether interrupts are generated after rollover, input capture and output compare events.

Address: Base address + Ch offset



GPTx_IR field descriptions

Field	Description
31–6 Reserved	This read-only field is reserved and always has the value 0.
5 ROVIE	Rollover Interrupt Enable. The ROVIE bit controls the Rollover interrupt. 0 Rollover interrupt is disabled. 1 Rollover interrupt enabled.
4 IF2IE	IF2IE Input capture 2 Interrupt Enable IF1IE Input capture 1 Interrupt Enable The IF*n*IE bit controls the IF*n*IE Input Capture *n* Interrupt Enable. 0 IF2IE Input Capture *n* Interrupt Enable is disabled. 1 IF2IE Input Capture *n* Interrupt Enable is enabled.

Table continues on the next page...

GPTx_IR field descriptions (continued)

Field	Description
3 IF1IE	See IF2IE
2 OF3IE	OF3IE Output Compare 3 Interrupt Enable OF2IE Output Compare 2 Interrupt Enable OF1IE Output Compare 1 Interrupt Enable The OF*n*IE bit controls the Output Compare Channel *n* interrupt. 0 Output Compare Channel *n* interrupt is disabled. 1 Output Compare Channel *n* interrupt is enabled.
1 OF2IE	See OF3IE
0 OF1IE	See OF3IE

43.7.5 GPT Output Compare Register 1 (GPTx_OCR1)

The GPT Compare Register 1 (GPT_OCR1) holds the value that determines when a compare event will be generated on Output Compare Channel 1. Any write access to the Compare register of Channel 1 while in Restart mode (FRR=0) will reset the GPT counter.

An IP Bus Write access to the GPT Output Compare Register1 (GPT_OCR1) occurs *after* one cycle of wait state; an IP Bus Read access occurs *immediately* (0 wait states).

Address: Base address + 10h 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	COMP																															
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

GPTx_OCR1 field descriptions

Field	Description
COMP	Compare Value. When the counter value equals the COMP bit field value, a compare event is generated on Output Compare Channel 1.

43.7.6 GPT Output Compare Register 2 (GPTx_OCR2)

The GPT Compare Register 2 (GPT_OCR2) holds the value that determines when a compare event will be generated on Output Compare Channel 2.

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	COMP																															
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

GPTx_OCR2 field descriptions

Field	Description
COMP	Compare Value. When the counter value equals the COMP bit field value, a compare event is generated on Output Compare Channel 2.

43.7.7 GPT Output Compare Register 3 (GPTx_OCR3)

The GPT Compare Register 3 (GPT_OCR3) holds the value that determines when a compare event will be generated on Output Compare Channel 3.

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	COMP																															
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

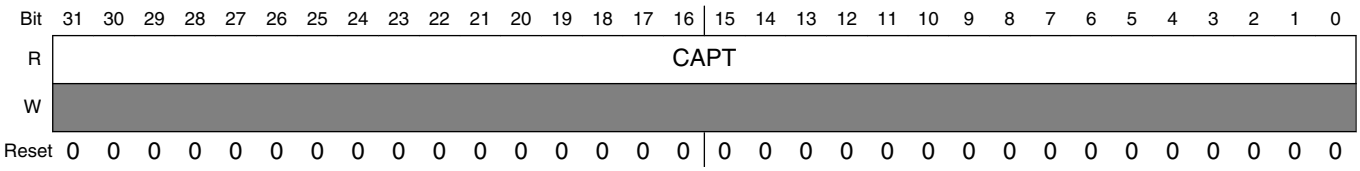
GPTx_OCR3 field descriptions

Field	Description
COMP	Compare Value. When the counter value equals the COMP bit field value, a compare event is generated on Output Compare Channel 3.

43.7.8 GPT Input Capture Register 1 (GPTx_ICR1)

The GPT Input Capture Register 1 (GPT_ICR1) is a read-only register that holds the value *that was in the counter during the last capture event* on Input Capture Channel 1.

Address: Base address + 1Ch offset



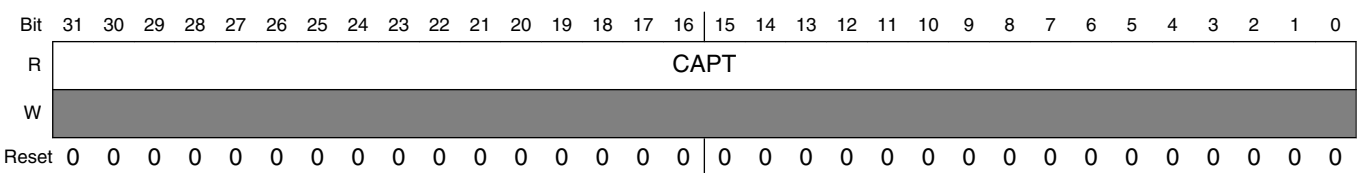
GPTx_ICR1 field descriptions

Field	Description
CAPT	Capture Value. After a capture event on Input Capture Channel 1 occurs, the current value of the counter is loaded into GPT Input Capture Register 1.

43.7.9 GPT Input Capture Register 2 (GPTx_ICR2)

The GPT Input capture Register 2 (GPT_ICR2) is a read-only register which holds the value that was in the counter during the last capture event on input capture channel 2.

Address: Base address + 20h offset



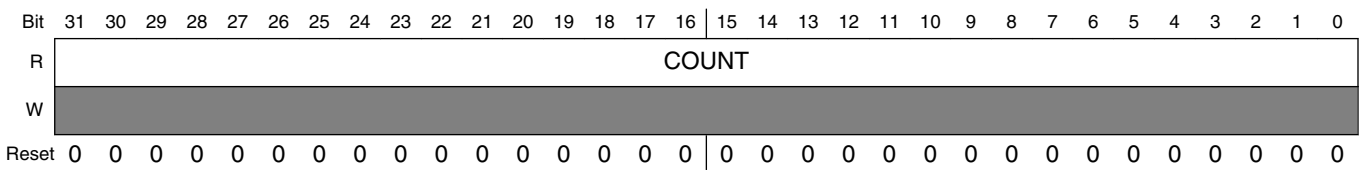
GPTx_ICR2 field descriptions

Field	Description
CAPT	Capture Value. After a capture event on Input Capture Channel 2 occurs, the current value of the counter is loaded into GPT Input Capture Register 2.

43.7.10 GPT Counter Register (GPTx_CNT)

The GPT Counter Register (GPT_CNT) is the main counter's register. GPT_CNT is a read-only register and can be read *without affecting the counting process* of the GPT.

Address: Base address + 24h offset



GPTx_CNT field descriptions

Field	Description
COUNT	Counter Value. The COUNT bits show the current count value of the GPT counter.

Chapter 44
Periodic Interrupt Timer (PIT)

44.1 Chip-specific PIT 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

The number of PIT channels is 4, in this device.

44.2 Introduction

The PIT module is an array of timers that can be used to raise interrupts and trigger DMA channels.

44.2.1 Block diagram

The following figure shows the block diagram of PIT module.

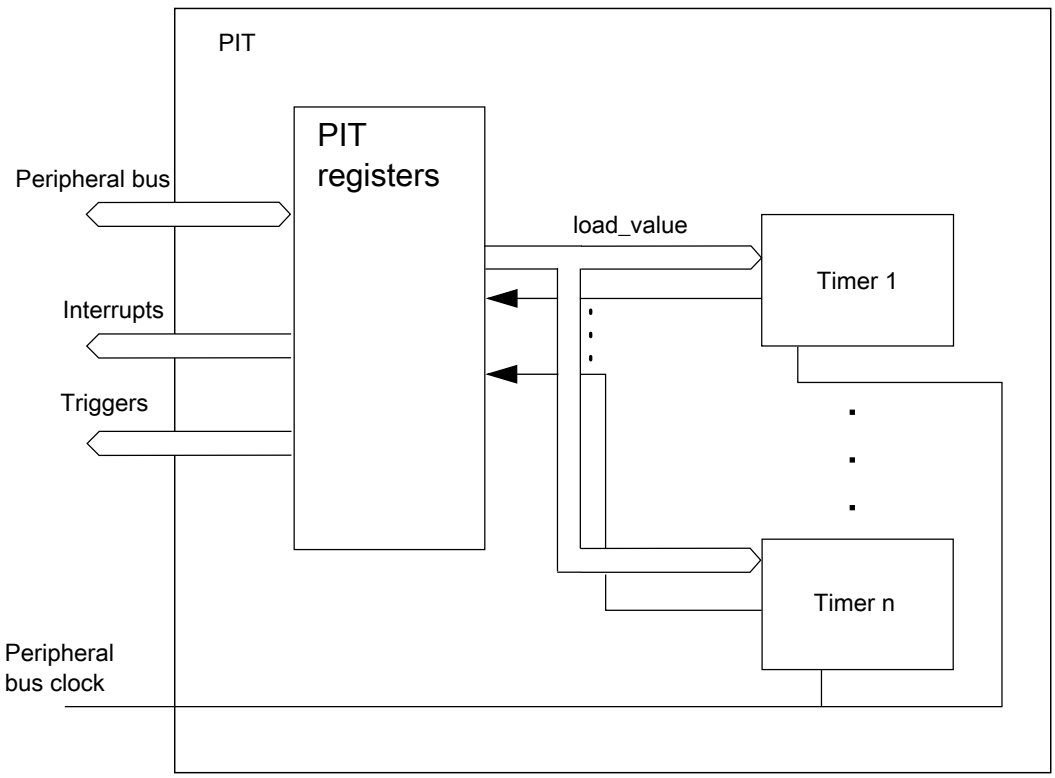


Figure 44-1. Block diagram of PIT

44.2.2 Features

The key features of the module are these:

- Ability of timers to generate DMA trigger pulses
- Ability of timers to generate interrupts
- Maskable interrupts
- Independent timeout periods for each timer

44.3 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.

- Stop mode

44.4 PIT External Signals

Signal	Description	I/O
TRIGGERn	Trigger signal	O

44.5 Functional description

This section provides the functional description of the module.

44.5.1 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.

44.5.1.1 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.

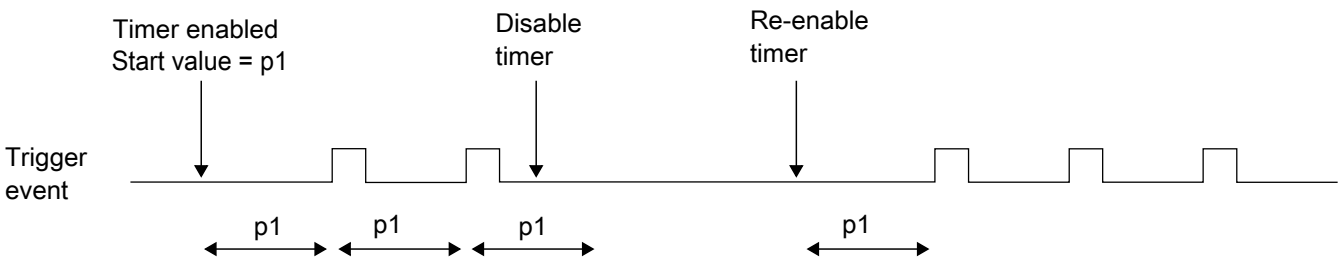


Figure 44-2. Stopping and starting a timer

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.

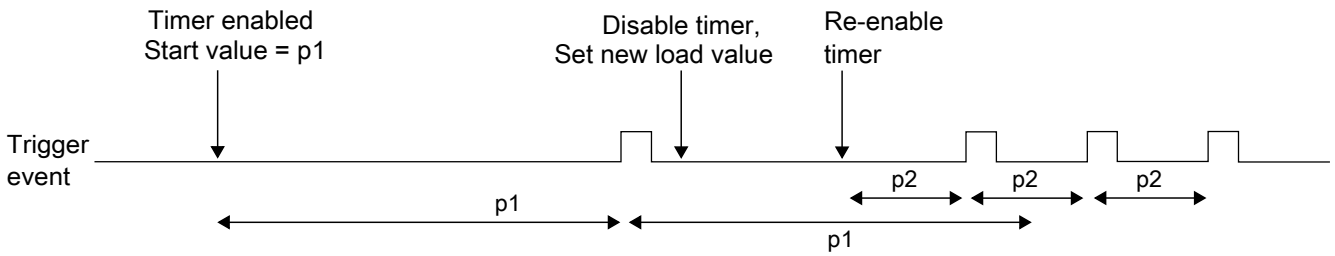


Figure 44-3. Modifying running timer period

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.

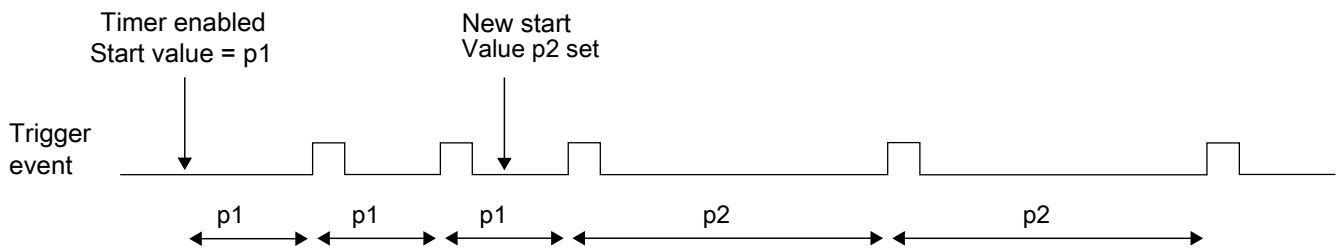


Figure 44-4. Dynamically setting a new load value

NOTE

- If the pause is initiated, when the timer is nearing 0 (CVALn =0x0), the pause command may not make it to the IP before the timer expires and generates a trigger.

- Pause will be ignored if (CVALn =0x0), but the trigger will remain asserted until the pause is removed. The user is recommended to remove Pause and then clear the interrupt TFLGn[TIF].
- If the timers are to be paused, sufficient time must be ensured for the IP to react.

44.5.1.2 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.

44.5.2 Interrupts

All the timers support interrupt generation. See the MCU specification for related vector addresses and priorities.

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].

44.5.3 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.

44.6 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
```

44.7 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
```

44.8 Example configuration for the lifetime timer

To configure the lifetimer 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 timer is a downcounter.

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, read first LTMR64H and then LTMR64L.

```
current_uptime = PIT_LTMR64H<<32;
current_uptime = current_uptime + PIT_LTMR64L;
```

44.9 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.

44.9.1 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

44.9.2 PIT Module Control Register (MCR)

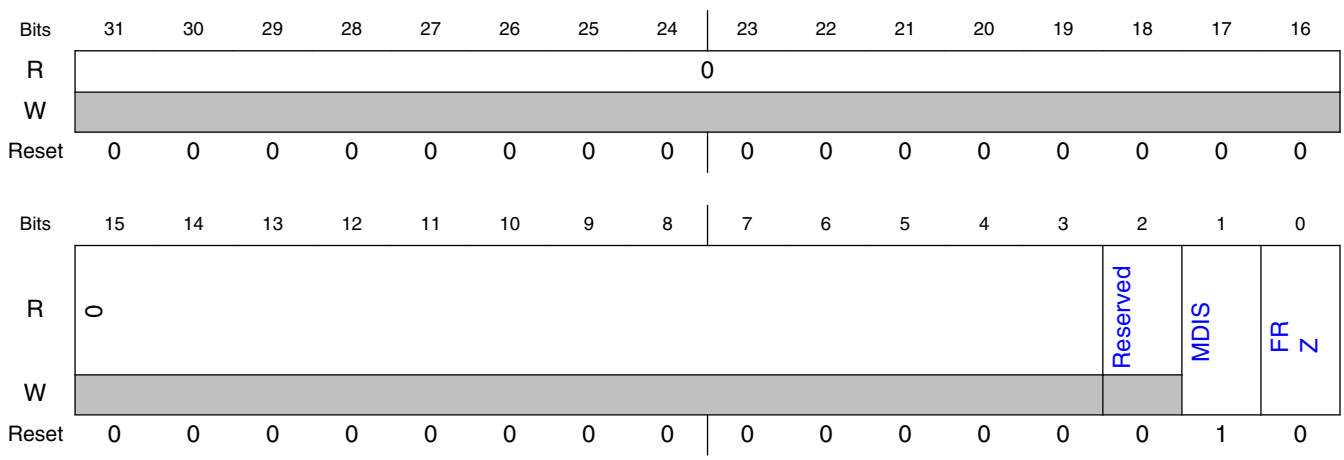
44.9.2.1 Offset

Register	Offset
MCR	0h

44.9.2.2 Function

This register enables or disables the PIT timer clocks and controls the timers when the PIT enters the Debug mode.

44.9.2.3 Diagram



44.9.2.4 Fields

Field	Function
31-3 —	Reserved
2 —	Reserved

Table continues on the next page...

Field	Function
1 MDIS	Module Disable for PIT Disables the standard timers. The field must be enabled before any other setup is done. 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.

44.9.3 PIT Upper Lifetime Timer Register (LTMR64H)

44.9.3.1 Offset

Register	Offset
LTMR64H	E0h

44.9.3.2 Function

This register is intended for applications that chain timer 0 and timer 1 to build a 64-bit lifetimer.

44.9.3.3 Diagram

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

44.9.3.4 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.

44.9.4 PIT Lower Lifetime Timer Register (LTMR64L)

44.9.4.1 Offset

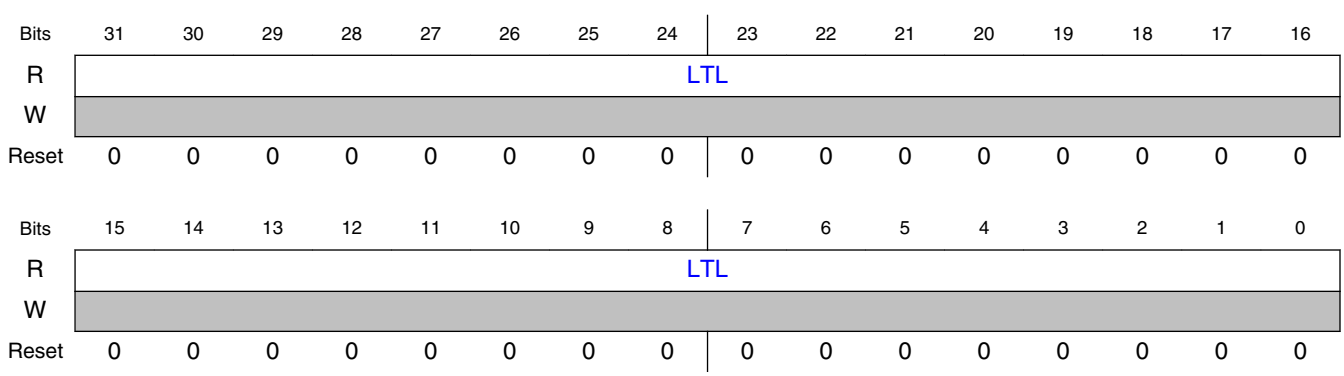
Register	Offset
LTMR64L	E4h

44.9.4.2 Function

This register is intended for applications that chain timer 0 and timer 1 to build a 64-bit lifetimer.

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.

44.9.4.3 Diagram



44.9.4.4 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.

44.9.5 Timer Load Value Register (LDVAL0 - LDVAL3)

44.9.5.1 Offset

Register	Offset
LDVAL0	100h
LDVAL1	110h
LDVAL2	120h
LDVAL3	130h

44.9.5.2 Function

These registers select the timeout period for the timer interrupts.

44.9.5.3 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

44.9.5.4 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.

44.9.6 Current Timer Value Register (CVAL0 - CVAL3)

44.9.6.1 Offset

Register	Offset
CVAL0	104h
CVAL1	114h
CVAL2	124h
CVAL3	134h

44.9.6.2 Function

These registers indicate the current timer position.

44.9.6.3 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

44.9.6.4 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.

44.9.7 Timer Control Register (TCTRL0 - TCTRL3)

44.9.7.1 Offset

Register	Offset
TCTRL0	108h
TCTRL1	118h
TCTRL2	128h
TCTRL3	138h

44.9.7.2 Function

These registers contain the control bits for each timer.

44.9.7.3 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

44.9.7.4 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.

44.9.8 Timer Flag Register (TFLG0 - TFLG3)

44.9.8.1 Offset

Register	Offset
TFLG0	10Ch
TFLG1	11Ch
TFLG2	12Ch
TFLG3	13Ch

44.9.8.2 Function

These registers hold the PIT interrupt flags.

44.9.8.3 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

44.9.8.4 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 45
Enhanced Flex Pulse Width Modulator (eFlexPWM)

45.1 Chip-specific FlexPWM 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

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.

45.2 Introduction

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.

This PWM module can generate various switching patterns, including highly sophisticated waveforms. It can be used to control all known motor types and is ideal for controlling different Switched Mode Power Supplies (SMPS) topologies as well.

45.2.1 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
- Dithering to simulate enhanced resolution when fine edge placement is not available
- 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
- The option to supply the source for each complementary PWM signal pair from any of the following:
  - Crossbar module outputs
  - External ADC input, taking into account values set in ADC high and low limit registers

45.2.2 Modes of Operation

Be careful when using this module in stop, wait and debug operating modes.

CAUTION

Some applications (such as three-phase AC motors) 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 (EOnCE) modes. PWM outputs are reactivated (assuming they were active beforehand) when these modes are exited.

Table 45-2. 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].

45.2.3 Block Diagram

The following figure is a block diagram of the PWM.

45.2.3.1 PWM Submodule

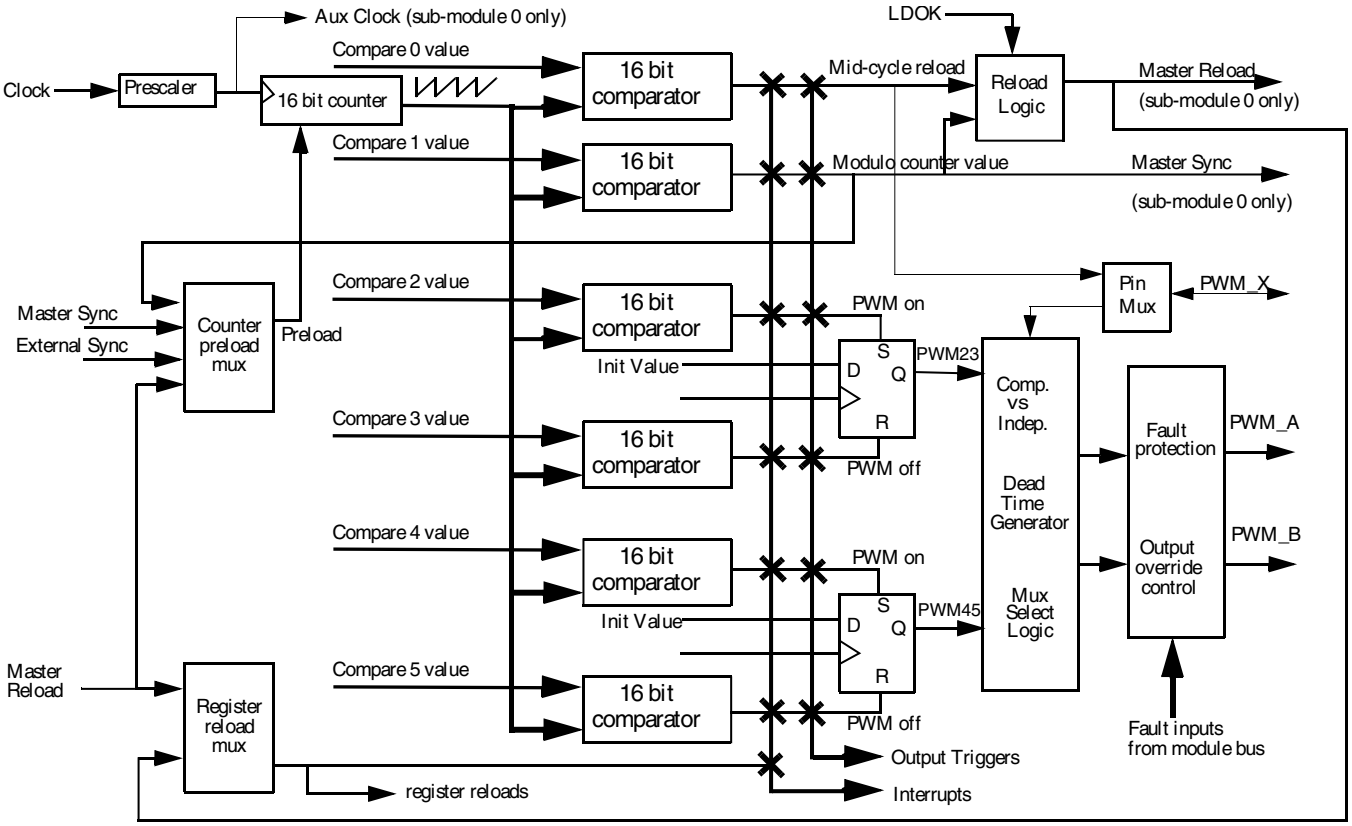


Figure 45-1. PWM Submodule Block Diagram

45.3 Signal Descriptions

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 output signals called PWM[n]_OUT_TRIGx.

45.3.1 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.

45.3.2 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.

45.3.3 FAULT[n] - Fault Inputs

These are input pins for disabling selected PWM outputs.

45.3.4 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.

45.3.5 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.

45.3.6 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.

45.3.7 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 Output Trigger Control Register for information about how to enable these outputs and how the compare registers match up to the output triggers.

45.3.8 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.

45.4 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. While the registers are 16-bit wide, they can be accessed in pairs as 32-bit registers.

Submodule registers are repeated for each PWM submodule. To designate which submodule they are in, register names are prefixed with SM0, SM1, SM2, and SM3. 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 $60 from the base address for the PWM module as a whole. This $60 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 $60 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 $180.

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 $18C. The base address of fault channel 1 is the base address of fault channel 0 + $4. This $4 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.

45.4.1 PWM memory map

PWM base address: 401C_C000h

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
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
2Eh	Fault Disable Mapping Register 1 (SM0DISMAP1)	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

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
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
58h	Phase Delay Register (SM0PHASEDLY)	16	RW	0000h
60h	Counter Register (SM1CNT)	16	RO	0000h
62h	Initial Count Register (SM1INIT)	16	RW	0000h
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
8Eh	Fault Disable Mapping Register 1 (SM1DISMAP1)	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

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
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
B8h	Phase Delay Register (SM1PHASEDLY)	16	RW	0000h
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
EEh	Fault Disable Mapping Register 1 (SM2DISMAP1)	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

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
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
114h	Capture Value 5 Register (SM2CVAL5)	16	RO	0000h
116h	Capture Value 5 Cycle Register (SM2CVAL5CYC)	16	RO	0000h
118h	Phase Delay Register (SM2PHASEDLY)	16	RW	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
14Eh	Fault Disable Mapping Register 1 (SM3DISMAP1)	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

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
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
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
178h	Phase Delay Register (SM3PHASEDLY)	16	RW	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

45.4.2 Counter Register (SM0CNT - SM3CNT)

45.4.2.1 Offset

Register	Offset
SM0CNT	0h
SM1CNT	60h
SM2CNT	C0h

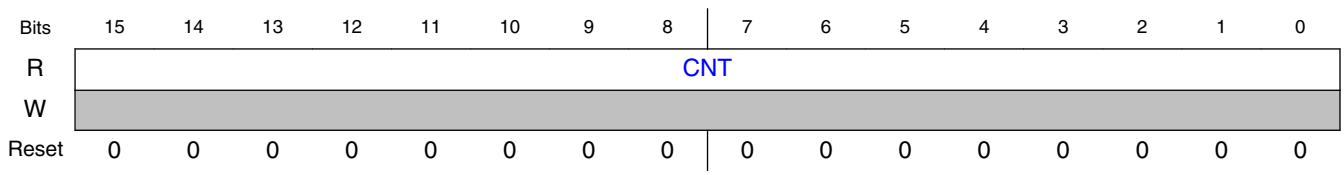
Table continues on the next page...

Register	Offset
SM3CNT	120h

45.4.2.2 Function

This read-only register displays the state of the signed 16-bit submodule counter. This register is not byte accessible.

45.4.2.3 Diagram



45.4.2.4 Fields

Field	Function
15-0 CNT	Counter Register Bits

45.4.3 Initial Count Register (SM0INIT - SM3INIT)

45.4.3.1 Offset

Register	Offset
SM0INIT	2h
SM1INIT	62h
SM2INIT	C2h
SM3INIT	122h

45.4.3.2 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.

45.4.3.3 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

45.4.3.4 Fields

Field	Function
15-0 INIT	Initial Count Register Bits

45.4.4 Control 2 Register (SM0CTRL2 - SM3CTRL2)

45.4.4.1 Offset

Register	Offset
SM0CTRL2	4h
SM1CTRL2	64h
SM2CTRL2	C4h
SM3CTRL2	124h

45.4.4.2 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

45.4.4.3 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. For certain types of motors (such as 3-phase AC), it is imperative that this bit be left in its default state (in which the PWM is disabled in debug mode). Failure to do so could result in damage to the motor or inverter. For other types of motors (example: DC motors), this bit might safely be set to one, enabling the PWM in debug mode. The key point is PWM parameter updates will not occur in debug mode. Any motors requiring such updates should be disabled during debug mode. If in doubt, leave this bit set to zero.
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. For certain types of motors (such as 3-phase AC), it is imperative that this bit be left in its default state (in which the PWM is disabled in WAIT mode). Failure to do so could result in damage to the motor or inverter. For other types of motors (example: DC motors), this bit might safely be set to one, enabling the PWM in WAIT mode. The key point is PWM parameter updates will not occur in this mode. Any motors requiring such updates should be disabled during WAIT mode. If in doubt, leave this bit set to zero.

Table continues on the next page...

Field	Function
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. 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 local 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 Source Select

Table continues on the next page...

Field	Function
RELOAD_SEL	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

45.4.5 Control Register (SM0CTRL - SM3CTRL)

45.4.5.1 Offset

Register	Offset
SM0CTRL	6h
SM1CTRL	66h
SM2CTRL	C6h
SM3CTRL	126h

45.4.5.2 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

45.4.5.3 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
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.

Table continues on the next page...

Field	Function
	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 - PWM clock frequency = f_{clk} 001b - PWM clock frequency = $f_{clk}/2$ 010b - PWM clock frequency = $f_{clk}/4$ 011b - PWM clock frequency = $f_{clk}/8$ 100b - PWM clock frequency = $f_{clk}/16$ 101b - PWM clock frequency = $f_{clk}/32$ 110b - PWM clock frequency = $f_{clk}/64$ 111b - PWM clock frequency = $f_{clk}/128$
3 SPLIT	Split the DBLPWM signal to PWMA and PWMB This read/write bit is only used 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 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.

45.4.6 Value Register 0 (SM0VAL0 - SM3VAL0)

45.4.6.1 Offset

Register	Offset
SM0VAL0	Ah
SM1VAL0	6Ah
SM2VAL0	CAh
SM3VAL0	12Ah

45.4.6.2 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

45.4.6.3 Fields

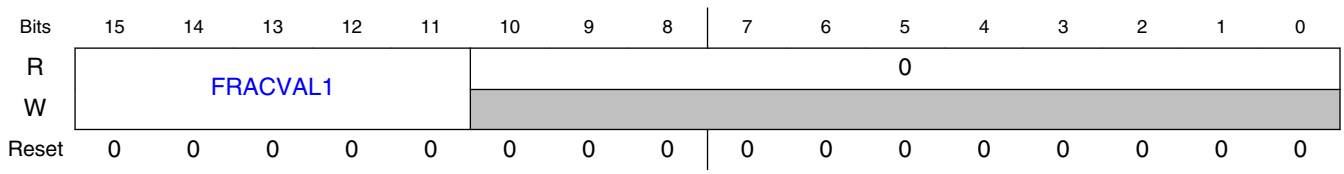
Field	Function
15-0 VAL0	Value Register 0 The 16-bit signed value in this buffered, read/write register defines the mid-cycle reload point for the PWM in PWM clock periods. This value also defines when the PWM_X signal is set and the local sync signal is reset. This register is not byte accessible. **NOTE:** 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.

45.4.7 Fractional Value Register 1 (SM0FRACVAL1 - SM3FRACVAL1)

45.4.7.1 Offset

Register	Offset
SM0FRACVAL1	Ch
SM1FRACVAL1	6Ch
SM2FRACVAL1	CCh
SM3FRACVAL1	12Ch

45.4.7.2 Diagram



45.4.7.3 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

45.4.8 Value Register 1 (SM0VAL1 - SM3VAL1)

45.4.8.1 Offset

Register	Offset
SM0VAL1	Eh
SM1VAL1	6Eh
SM2VAL1	CEh
SM3VAL1	12Eh

45.4.8.2 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

45.4.8.3 Fields

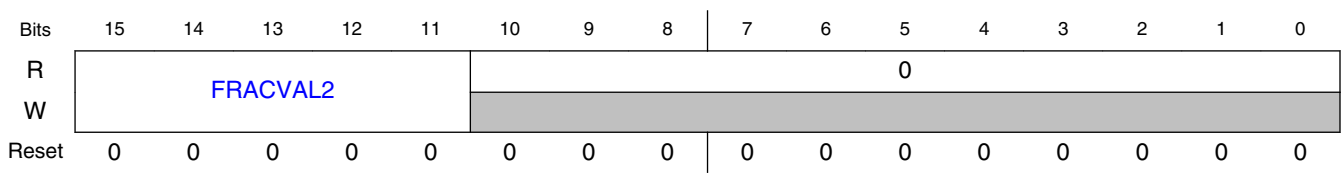
Field	Function
15-0 VAL1	Value Register 1 The 16-bit signed value written to this buffered, read/write register defines the modulo count value (maximum count) for the submodule counter. Upon reaching this count value, the counter reloads itself with the contents of the INIT register and asserts the local 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.

45.4.9 Fractional Value Register 2 (SM0FRACVAL2 - SM3FRACVAL2)

45.4.9.1 Offset

Register	Offset
SM0FRACVAL2	10h
SM1FRACVAL2	70h
SM2FRACVAL2	D0h
SM3FRACVAL2	130h

45.4.9.2 Diagram



45.4.9.3 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

45.4.10 Value Register 2 (SM0VAL2 - SM3VAL2)

45.4.10.1 Offset

Register	Offset
SM0VAL2	12h
SM1VAL2	72h
SM2VAL2	D2h
SM3VAL2	132h

45.4.10.2 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

45.4.10.3 Fields

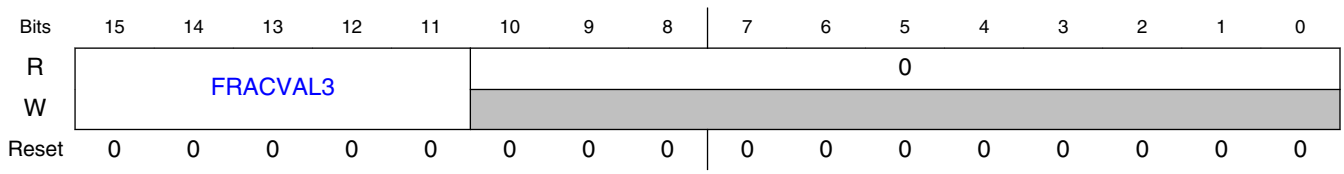
Field	Function
15-0 VAL2	Value Register 2 The 16-bit signed value in this buffered, read/write register defines the count value to set PWM23 high. This register is not byte accessible. **NOTE:** The VAL2 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next 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.

45.4.11 Fractional Value Register 3 (SM0FRACVAL3 - SM3FRACVAL3)

45.4.11.1 Offset

Register	Offset
SM0FRACVAL3	14h
SM1FRACVAL3	74h
SM2FRACVAL3	D4h
SM3FRACVAL3	134h

45.4.11.2 Diagram



45.4.11.3 Fields

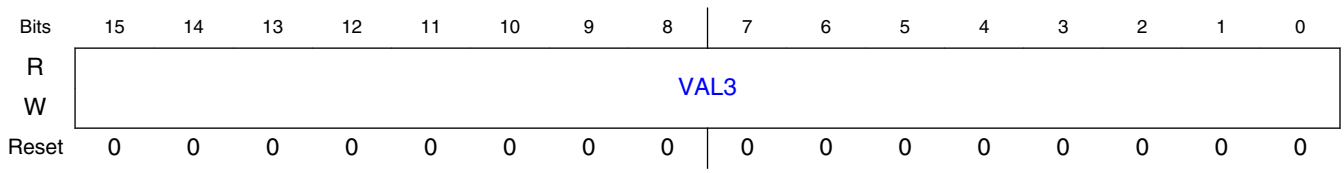
Field	Function
15-11 FRACVAL3	Fractional Value 3 These bits act as a fractional addition to the value in the VAL3 register which controls the PWM_A turn off timing. It is also used to control the fractional addition to the turn on delay of PWM_B when MCTRL[IPOLx]=0 in complementary mode, CTRL2[INDEP]=0. **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

45.4.12 Value Register 3 (SM0VAL3 - SM3VAL3)

45.4.12.1 Offset

Register	Offset
SM0VAL3	16h
SM1VAL3	76h
SM2VAL3	D6h
SM3VAL3	136h

45.4.12.2 Diagram



45.4.12.3 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.

45.4.13 Fractional Value Register 4 (SM0FRACVAL4 - SM3FRACVAL4)

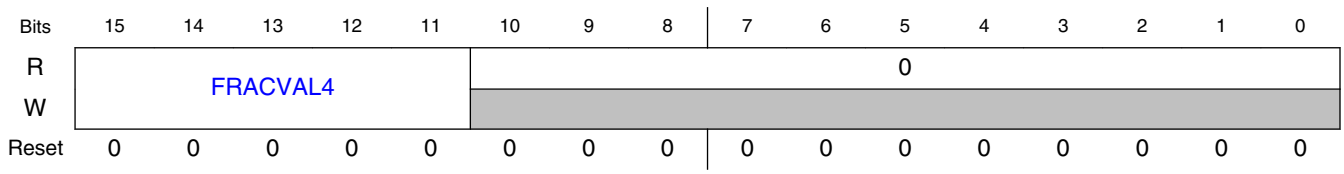
45.4.13.1 Offset

Register	Offset
SM0FRACVAL4	18h
SM1FRACVAL4	78h

Table continues on the next page...

Register	Offset
SM2FRACVAL4	D8h
SM3FRACVAL4	138h

45.4.13.2 Diagram



45.4.13.3 Fields

Field	Function
15-11 FRACVAL4	Fractional Value 4 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

45.4.14 Value Register 4 (SM0VAL4 - SM3VAL4)

45.4.14.1 Offset

Register	Offset
SM0VAL4	1Ah
SM1VAL4	7Ah
SM2VAL4	DAh
SM3VAL4	13Ah

45.4.14.2 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

45.4.14.3 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. **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.

45.4.15 Fractional Value Register 5 (SM0FRACVAL5 - SM3FRACVAL5)

45.4.15.1 Offset

Register	Offset
SM0FRACVAL5	1Ch
SM1FRACVAL5	7Ch
SM2FRACVAL5	DCh
SM3FRACVAL5	13Ch

45.4.15.2 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

45.4.15.3 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. **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

45.4.16 Value Register 5 (SM0VAL5 - SM3VAL5)

45.4.16.1 Offset

Register	Offset
SM0VAL5	1Eh
SM1VAL5	7Eh
SM2VAL5	DEh
SM3VAL5	13Eh

45.4.16.2 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

45.4.16.3 Fields

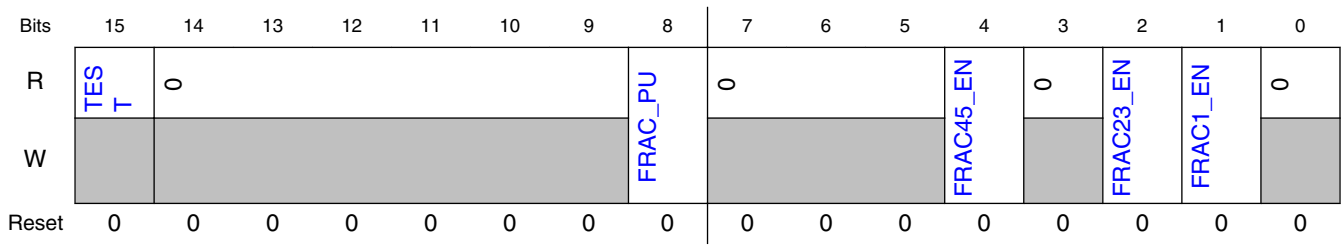
Field	Function
15-0 VAL5	Value Register 5 The 16-bit signed value in this buffered, read/write register defines the count value to set PWM45 low. This register is not byte accessible. **NOTE:** The VAL5 register is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next 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.

45.4.17 Fractional Control Register (SM0FRCTRL - SM3FRCTRL)

45.4.17.1 Offset

Register	Offset
SM0FRCTRL	20h
SM1FRCTRL	80h
SM2FRCTRL	E0h
SM3FRCTRL	140h

45.4.17.2 Diagram



45.4.17.3 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. 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.

Table continues on the next page...

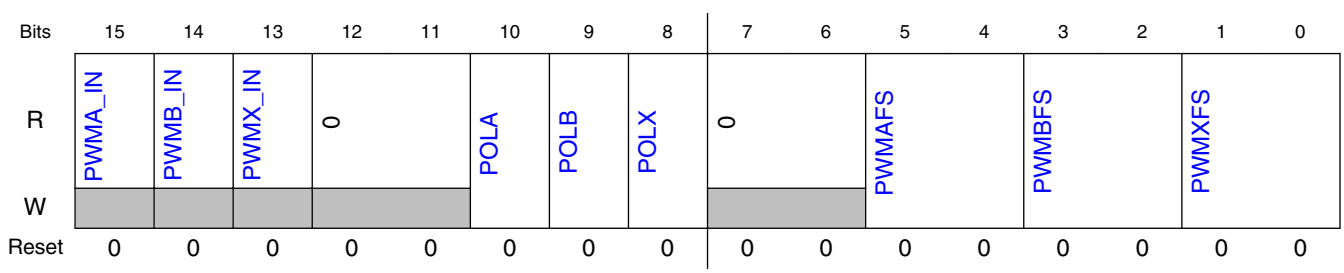
Field	Function
	NOTE: The FRAC23_EN bit is buffered. The value written does not take effect until MCTRL[LDOK] is set and the next PWM load cycle begins or CTRL[LDMOD] is set. FRAC23_EN cannot be written when MCTRL[LDOK] is set. Reading FRAC23_EN reads the value in a buffer and not necessarily the value 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

45.4.18 Output Control Register (SM0OCTRL - SM3OCTRL)

45.4.18.1 Offset

Register	Offset
SM0OCTRL	22h
SM1OCTRL	82h
SM2OCTRL	E2h
SM3OCTRL	142h

45.4.18.2 Diagram



45.4.18.3 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 POLX	PWM_X Output Polarity 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. 10b - Output is tristated. 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.

Table continues on the next page...

Field	Function
	10b - Output is tristated. 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. 10b - Output is tristated. 11b - Output is tristated.

45.4.19 Status Register (SM0STS - SM3STS)

45.4.19.1 Offset

Register	Offset
SM0STS	24h
SM1STS	84h
SM2STS	E4h
SM3STS	144h

45.4.19.2 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

45.4.19.3 Fields

Field	Function
15	RESERVED

Table continues on the next page...

Field	Function
—	
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 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.

Field	Function
	000000b - No compare event has occurred for a particular VALx value. 000001b - A compare event has occurred for a particular VALx value.

45.4.20 Interrupt Enable Register (SM0INTEN - SM3INTEN)

45.4.20.1 Offset

Register	Offset
SM0INTEN	26h
SM1INTEN	86h
SM2INTEN	E6h
SM3INTEN	146h

45.4.20.2 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

45.4.20.3 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.

Table continues on the next page...

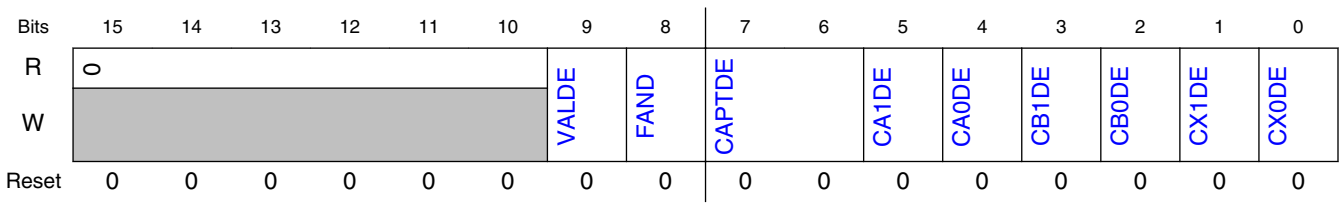
Field	Function
	0b - STS[RF] CPU interrupt requests disabled 1b - STS[RF] CPU interrupt requests enabled
11 CA1IE	Capture A 1 Interrupt Enable This bit allows the STS[CFA1] flag to create an interrupt request to the CPU. Do not set both this bit and DMAEN[CA1DE]. 0b - Interrupt request disabled for STS[CFA1]. 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]. 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.

45.4.21 DMA Enable Register (SM0DMAEN - SM3DMAEN)

45.4.21.1 Offset

Register	Offset
SM0DMAEN	28h
SM1DMAEN	88h
SM2DMAEN	E8h
SM3DMAEN	148h

45.4.21.2 Diagram



45.4.21.3 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 1b - 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.

Table continues on the next page...

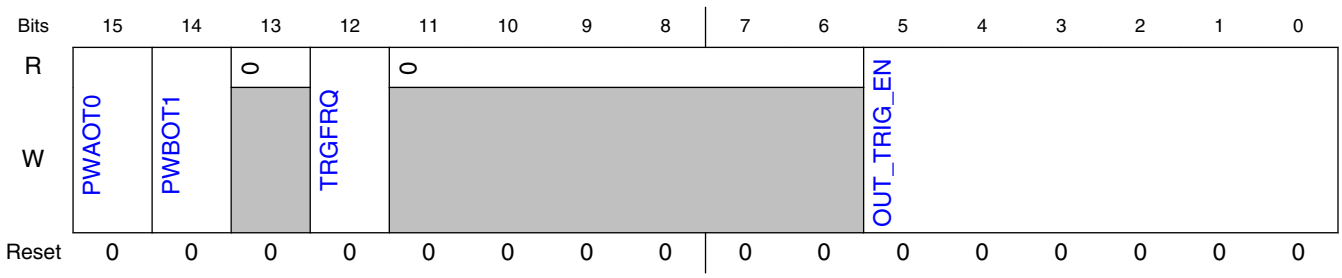
Field	Function
	10b - A local sync (VAL1 matches counter) sets the read DMA request. 11b - A local reload (STS[RF] being set) sets the read DMA request.
5 CA1DE	Capture A1 FIFO DMA Enable This read/write bit enables DMA read requests for the Capture A1 FIFO data when STS[CFA1] is set. Reset clears this 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].

45.4.22 Output Trigger Control Register (SM0TCTRL - SM3T CTRL)

45.4.22.1 Offset

Register	Offset
SM0TCTRL	2Ah
SM1TCTRL	8Ah
SM2TCTRL	EAh
SM3TCTRL	14Ah

45.4.22.2 Diagram



45.4.22.3 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]. 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. VAL0, VAL2, and VAL4 are used to generate PWM_OUT_TRIG0, and VAL1, VAL3, and VAL5 are used to generate PWM_OUT_TRIG1. The PWM_OUT_TRIGx signals are only asserted as long as the counter value matches the VALx value; therefore, up to six triggers can be generated (three each on PWM_OUT_TRIG0 and PWM_OUT_TRIG1) per PWM cycle per submodule.

Field	Function
	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. 000000b - PWM_OUT_TRIGx will not set when the counter value matches the VALx value. 000001b - PWM_OUT_TRIGx will set when the counter value matches the VALx value.

45.4.23 Fault Disable Mapping Register 0 (SM0DISMAP0 - SM3DISMAP0)

45.4.23.1 Offset

Register	Offset
SM0DISMAP0	2Ch
SM1DISMAP0	8Ch
SM2DISMAP0	ECh
SM3DISMAP0	14Ch

45.4.23.2 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.

45.4.23.3 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

45.4.23.4 Fields

Field	Function
15-12	RESERVED

Table continues on the next page...

Field	Function
—	
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.

45.4.24 Fault Disable Mapping Register 1 (SM0DISMAP1 - SM3DISMAP1)

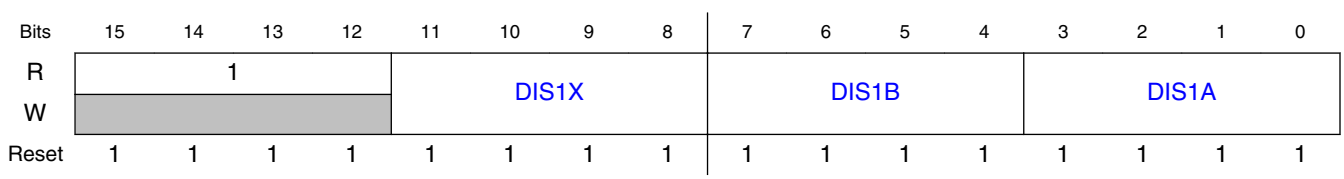
45.4.24.1 Offset

Register	Offset
SM0DISMAP1	2Eh
SM1DISMAP1	8Eh
SM2DISMAP1	EEh
SM3DISMAP1	14Eh

45.4.24.2 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.

45.4.24.3 Diagram



45.4.24.4 Fields

Field	Function
15-12 —	RESERVED
11-8 DIS1X	PWM_X Fault Disable Mask 1 Each of the four bits of this read/write field is one-to-one associated with the four FAULTx inputs of fault channel 1. 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 DIS1B	PWM_B Fault Disable Mask 1 Each of the four bits of this read/write field is one-to-one associated with the four FAULTx inputs of fault channel 1. 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 DIS1A	PWM_A Fault Disable Mask 1 Each of the four bits of this read/write field is one-to-one associated with the four FAULTx inputs of fault channel 1. 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.

45.4.25 Deadtime Count Register 0 (SM0DTCNT0 - SM3DTCNT0)

45.4.25.1 Offset

Register	Offset
SM0DTCNT0	30h
SM1DTCNT0	90h
SM2DTCNT0	F0h
SM3DTCNT0	150h

45.4.25.2 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.

45.4.25.3 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

45.4.25.4 Fields

Field	Function
15-0 DTCNT0	DTCNT0 The DTCNT0 field is interpreted differently depending on whether or not the fractional delays are being used (FRCNTRL[FRAC23_EN] is set). 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).

45.4.26 Deadtime Count Register 1 (SM0DTCNT1 - SM3DTCNT1)

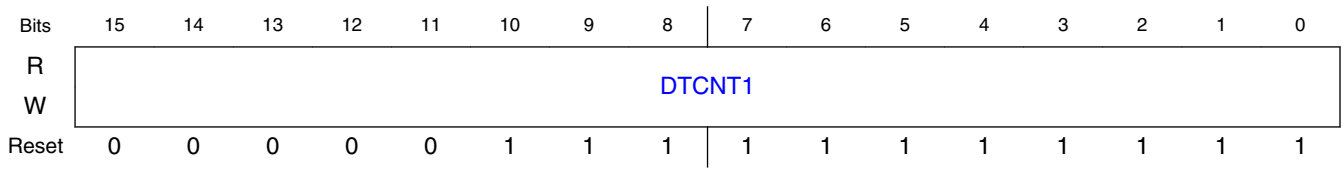
45.4.26.1 Offset

Register	Offset
SM0DTCNT1	32h
SM1DTCNT1	92h
SM2DTCNT1	F2h
SM3DTCNT1	152h

45.4.26.2 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.

45.4.26.3 Diagram



45.4.26.4 Fields

Field	Function
15-0 DTCNT1	DTCNT1 The DTCNT1 field is interpreted differently depending on whether or not the fractional delays are being used (FRCNTRL[FRAC45_EN] is set). 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).

45.4.27 Capture Control A Register (SM0CAPTCTRLA - SM3CAPTCTRLA)

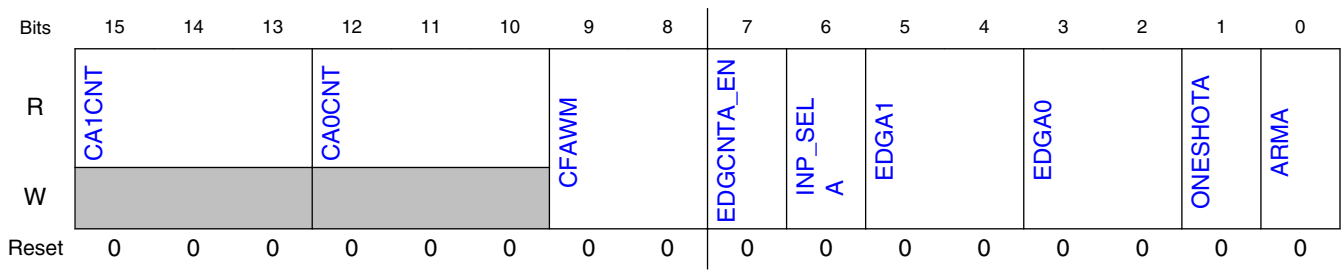
45.4.27.1 Offset

Register	Offset
SM0CAPTCTRLA	34h

Table continues on the next page...

Register	Offset
SM1CAPTCTRLA	94h
SM2CAPTCTRLA	F4h
SM3CAPTCTRLA	154h

45.4.27.2 Diagram



45.4.27.3 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 INP_SELA	Input Select A 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 - 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	Edge A 1

Table continues on the next page...

Field	Function
EDGA1	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 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 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.

45.4.28 Capture Compare A Register (SM0CAPTCOMPA - SM3CAPTCOMPA)

45.4.28.1 Offset

Register	Offset
SM0CAPTCOMPA	36h
SM1CAPTCOMPA	96h
SM2CAPTCOMPA	F6h
SM3CAPTCOMPA	156h

45.4.28.2 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

45.4.28.3 Fields

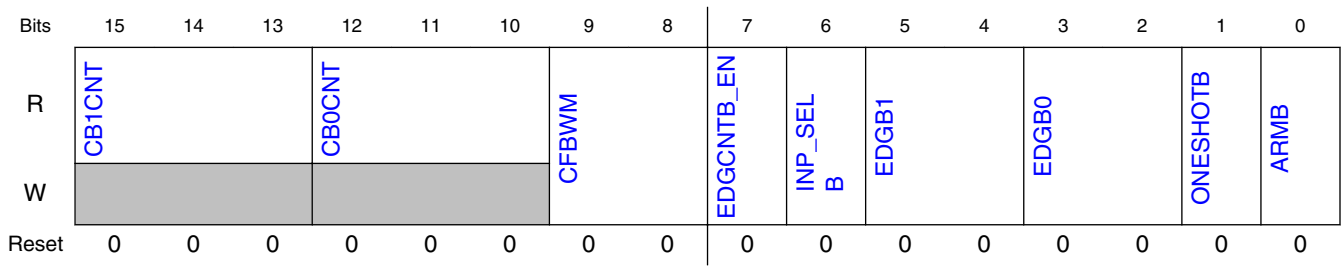
Field	Function
15-8 EDGCNTA	Edge Counter A This read-only field contains the edge counter value for the PWM_A input capture circuitry.
7-0 EDGCMPA	Edge Compare A This read/write field is the compare value associated with the edge counter for the PWM_A input capture circuitry.

45.4.29 Capture Control B Register (SM0CAPTCTRLB - SM3CAPTCTRLB)

45.4.29.1 Offset

Register	Offset
SM0CAPTCTRLB	38h
SM1CAPTCTRLB	98h
SM2CAPTCTRLB	F8h
SM3CAPTCTRLB	158h

45.4.29.2 Diagram



45.4.29.3 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 - 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 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

Table continues on the next page...

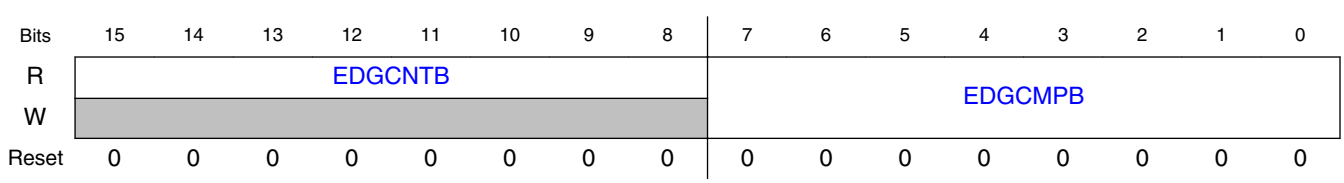
Field	Function
	01b - Capture falling edges 10b - Capture rising edges 11b - Capture any edge
1 ONESHOTB	One Shot Mode B This bit selects between free running and one shot mode for the input capture circuitry. 0b - Free running mode is selected. If both capture circuits are enabled, then capture circuit 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 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.

45.4.30 Capture Compare B Register (SM0CAPTCOMPB - SM3CAPTCOMPB)

45.4.30.1 Offset

Register	Offset
SM0CAPTCOMPB	3Ah
SM1CAPTCOMPB	9Ah
SM2CAPTCOMPB	FAh
SM3CAPTCOMPB	15Ah

45.4.30.2 Diagram



45.4.30.3 Fields

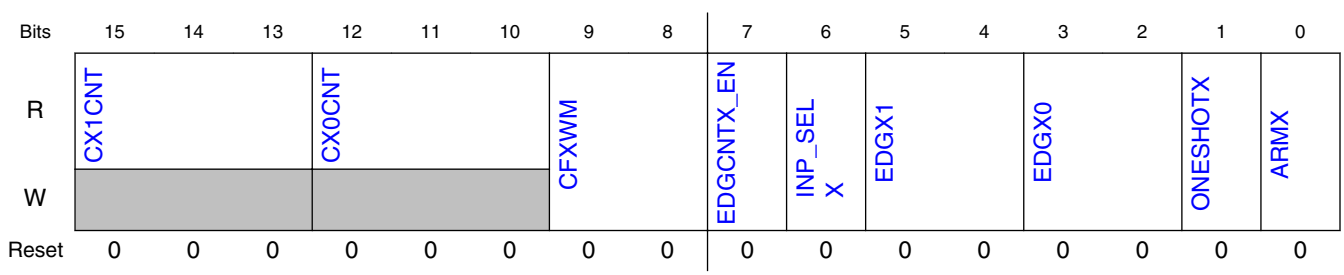
Field	Function
15-8 EDGCNTB	Edge Counter B This read-only field contains the edge counter value for the PWM_B input capture circuitry.
7-0 EDGCMPB	Edge Compare B This read/write field is the compare value associated with the edge counter for the PWM_B input capture circuitry.

45.4.31 Capture Control X Register (SM0CAPTCTRLX - SM3CAPTCTRLX)

45.4.31.1 Offset

Register	Offset
SM0CAPTCTRLX	3Ch
SM1CAPTCTRLX	9Ch
SM2CAPTCTRLX	FCh
SM3CAPTCTRLX	15Ch

45.4.31.2 Diagram



45.4.31.3 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 - 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 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 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 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

Table continues on the next page...

Field	Function
	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.

45.4.32 Capture Compare X Register (SM0CAPTCOMPX - SM3CAPTCOMPX)

45.4.32.1 Offset

Register	Offset
SM0CAPTCOMPX	3Eh
SM1CAPTCOMPX	9Eh
SM2CAPTCOMPX	FEh
SM3CAPTCOMPX	15Eh

45.4.32.2 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

45.4.32.3 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	Edge Compare X

Field	Function
EDGCMPX	This read/write field is the compare value associated with the edge counter for the PWM_X input capture circuitry.

45.4.33 Capture Value 0 Register (SM0CVAL0 - SM3CVAL0)

45.4.33.1 Offset

Register	Offset
SM0CVAL0	40h
SM1CVAL0	A0h
SM2CVAL0	100h
SM3CVAL0	160h

45.4.33.2 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

45.4.33.3 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.

45.4.34 Capture Value 0 Cycle Register (SM0CVAL0CYC - SM3CVAL0CYC)

45.4.34.1 Offset

Register	Offset
SM0CVAL0CYC	42h
SM1CVAL0CYC	A2h
SM2CVAL0CYC	102h
SM3CVAL0CYC	162h

45.4.34.2 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

45.4.34.3 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.

45.4.35 Capture Value 1 Register (SM0CVAL1 - SM3CVAL1)

45.4.35.1 Offset

Register	Offset
SM0CVAL1	44h
SM1CVAL1	A4h
SM2CVAL1	104h
SM3CVAL1	164h

45.4.35.2 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

45.4.35.3 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.

45.4.36 Capture Value 1 Cycle Register (SM0CVAL1CYC - SM3CVAL1CYC)

45.4.36.1 Offset

Register	Offset
SM0CVAL1CYC	46h
SM1CVAL1CYC	A6h
SM2CVAL1CYC	106h
SM3CVAL1CYC	166h

45.4.36.2 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

45.4.36.3 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.

45.4.37 Capture Value 2 Register (SM0CVAL2 - SM3CVAL2)

45.4.37.1 Offset

Register	Offset
SM0CVAL2	48h
SM1CVAL2	A8h
SM2CVAL2	108h
SM3CVAL2	168h

45.4.37.2 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

45.4.37.3 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.

45.4.38 Capture Value 2 Cycle Register (SM0CVAL2CYC - SM3CVAL2CYC)

45.4.38.1 Offset

Register	Offset
SM0CVAL2CYC	4Ah
SM1CVAL2CYC	AAh
SM2CVAL2CYC	10Ah
SM3CVAL2CYC	16Ah

45.4.38.2 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

45.4.38.3 Fields

Field	Function
15-4 —	RESERVED

Table continues on the next page...

Field	Function
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.

45.4.39 Capture Value 3 Register (SM0CVAL3 - SM3CVAL3)

45.4.39.1 Offset

Register	Offset
SM0CVAL3	4Ch
SM1CVAL3	ACh
SM2CVAL3	10Ch
SM3CVAL3	16Ch

45.4.39.2 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

45.4.39.3 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.

45.4.40 Capture Value 3 Cycle Register (SM0CVAL3CYC - SM3CVAL3CYC)

45.4.40.1 Offset

Register	Offset
SM0CVAL3CYC	4Eh
SM1CVAL3CYC	AEh
SM2CVAL3CYC	10Eh
SM3CVAL3CYC	16Eh

45.4.40.2 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

45.4.40.3 Fields

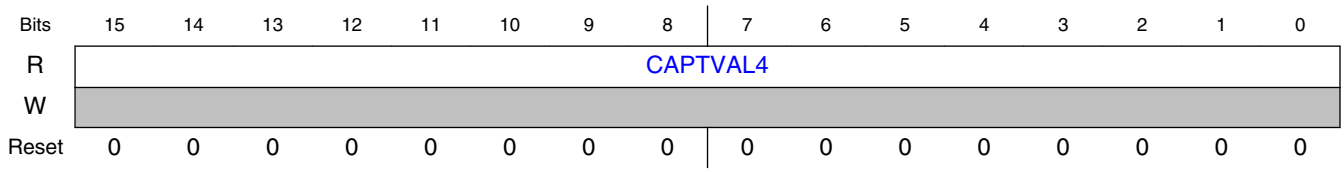
Field	Function
15-4 —	RESERVED
3-0 CVAL3CYC	CVAL3CYC This read-only register stores the cycle number corresponding to the value captured in CVAL3. This register is incremented each time the counter is loaded with the INIT value at the end of a PWM modulo cycle.

45.4.41 Capture Value 4 Register (SM0CVAL4 - SM3CVAL4)

45.4.41.1 Offset

Register	Offset
SM0CVAL4	50h
SM1CVAL4	B0h
SM2CVAL4	110h
SM3CVAL4	170h

45.4.41.2 Diagram



45.4.41.3 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.

45.4.42 Capture Value 4 Cycle Register (SM0CVAL4CYC - SM3CVAL4CYC)

45.4.42.1 Offset

Register	Offset
SM0CVAL4CYC	52h
SM1CVAL4CYC	B2h
SM2CVAL4CYC	112h
SM3CVAL4CYC	172h

45.4.42.2 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

45.4.42.3 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.

45.4.43 Capture Value 5 Register (SM0CVAL5 - SM3CVAL5)

45.4.43.1 Offset

Register	Offset
SM0CVAL5	54h
SM1CVAL5	B4h
SM2CVAL5	114h
SM3CVAL5	174h

45.4.43.2 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

45.4.43.3 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.

45.4.44 Capture Value 5 Cycle Register (SM0CVAL5CYC - SM3CVAL5CYC)

45.4.44.1 Offset

Register	Offset
SM0CVAL5CYC	56h
SM1CVAL5CYC	B6h
SM2CVAL5CYC	116h
SM3CVAL5CYC	176h

45.4.44.2 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

45.4.44.3 Fields

Field	Function
15-4 —	RESERVED

Table continues on the next page...

Field	Function
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.

45.4.45 Phase Delay Register (SM0PHASEDLY - SM3PHASEDLY)

45.4.45.1 Offset

Register	Offset
SM0PHASEDLY	58h
SM1PHASEDLY	B8h
SM2PHASEDLY	118h
SM3PHASEDLY	178h

45.4.45.2 Function

The 16-bit unsigned value in this buffered, read/write register defines the delay from the master sync signal of submodule 0 to the time that this submodule recognizes the master sync in PWM clock periods. CTRL2[INIT_SEL] must be set to 10b in order to select the master sync signal as the source for initialization when using this register. Setting this register with a non-zero value and using th master sync signal as the initilization source, allows the output of this submodule to be a fixed number of cycles delayed from submodule 0. For PWM operation, the buffered contents of this register are updated at the start of every PWM cycle. This register is not byte accessible.

NOTE

The PHASEDLY 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 PHASEDLY reads the value in a buffer and not necessarily the value the PWM generator is currently using. Also note, the value of this register should not be set to a value larger than the period defined in submodule 0.

45.4.45.3 Diagram

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

45.4.45.4 Fields

Field	Function
15-0 PHASEDLY	Initial Count Register Bits

45.4.46 Output Enable Register (OUTEN)

45.4.46.1 Offset

Register	Offset
OUTEN	180h

45.4.46.2 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

45.4.46.3 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.
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.

45.4.47 Mask Register (MASK)

45.4.47.1 Offset

Register	Offset
MASK	182h

45.4.47.2 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.

45.4.47.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R					MASKA				MASKB				MASKX			
W	UPDATE_MASK															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

45.4.47.4 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.

45.4.48 Software Controlled Output Register (SWCOUT)

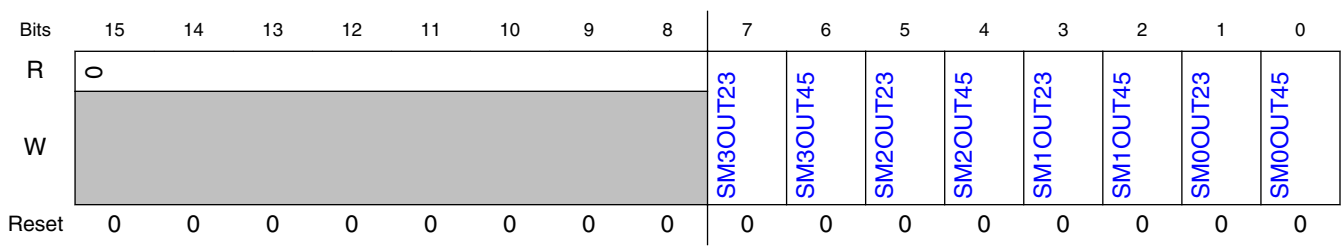
45.4.48.1 Offset

Register	Offset
SWCOUT	184h

45.4.48.2 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.

45.4.48.3 Diagram



45.4.48.4 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

Table continues on the next page...

Field	Function
	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.
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.

45.4.49 PWM Source Select Register (DTSRCSEL)

45.4.49.1 Offset

Register	Offset
DTSRCSEL	186h

45.4.49.2 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.

45.4.49.3 Diagram

Bits	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R W	SM3SEL23		SM3SEL45		SM2SEL23		SM2SEL45		SM1SEL23		SM1SEL45		SM0SEL23		SM0SEL45	
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

45.4.49.4 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

Table continues on the next page...

Field	Function
	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. 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.

45.4.50 Master Control Register (MCTRL)

45.4.50.1 Offset

Register	Offset
MCTRL	188h

45.4.50.2 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.

45.4.50.3 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

45.4.50.4 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. A reset clears this field. 0000b - PWM generator is disabled in the corresponding submodule. 0001b - PWM generator is enabled 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

Field	Function
	effect at the next PWM reload if CTRL[LDMOD] is clear or immediately if CTRL[LDMOD] is set. The VALx, FRACVALx,INIT, and CTRL[PRSC] registers of the corresponding submodule cannot be written while the 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. 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.

45.4.51 Master Control 2 Register (MCTRL2)

45.4.51.1 Offset

Register	Offset
MCTRL2	18Ah

45.4.51.2 Diagram

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

45.4.51.3 Fields

Field	Function
15-9 —	RESERVED
8 —	RESERVED

Table continues on the next page...

Field	Function
7-2 —	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 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.

45.4.52 Fault Control Register (FCTRL0)

45.4.52.1 Offset

Register	Offset
FCTRL0	18Ch

45.4.52.2 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.

45.4.52.3 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

45.4.52.4 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 setm 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.

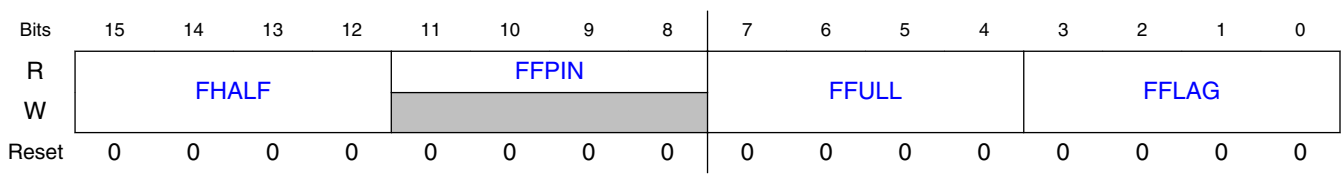
Field	Function
	0000b - FAULTx CPU interrupt requests disabled. 0001b - FAULTx CPU interrupt requests enabled.

45.4.53 Fault Status Register (FSTS0)

45.4.53.1 Offset

Register	Offset
FSTS0	18Eh

45.4.53.2 Diagram



45.4.53.3 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.

Table continues on the next page...

Field	Function
	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.

45.4.54 Fault Filter Register (FFILT0)

45.4.54.1 Offset

Register	Offset
FFILT0	190h

45.4.54.2 Function

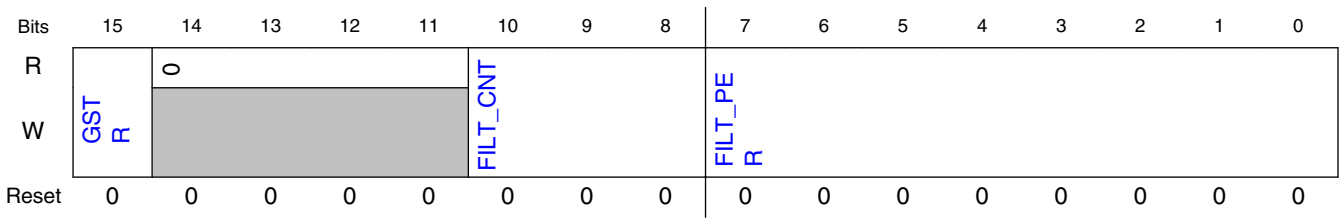
The settings in this register are shared among each of the fault input filters within the fault channel.

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].

45.4.54.3 Diagram



45.4.54.4 Fields

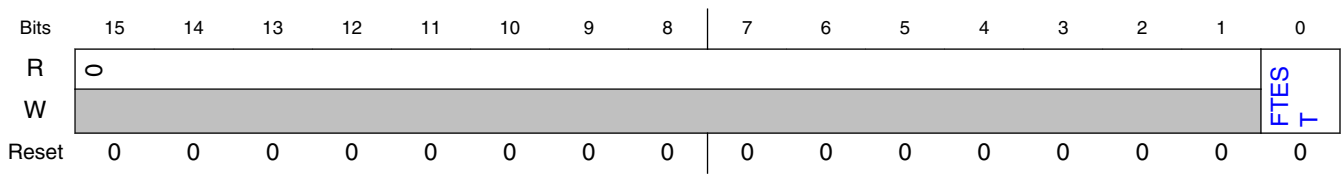
Field	Function
15 GSTR	Fault Glitch Stretch Enable This bit is used to enable the fault glitch stretching logic. This logic ensures that narrow fault glitches are stretched to be at least 2 IPBus clock cycles wide. In some cases a narrow fault input can cause problems due to the short PWM output shutdown/re-activation 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.

45.4.55 Fault Test Register (FTST0)

45.4.55.1 Offset

Register	Offset
FTST0	192h

45.4.55.2 Diagram



45.4.55.3 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

45.4.56 Fault Control 2 Register (FCTRL20)

45.4.56.1 Offset

Register	Offset
FCTRL20	194h

45.4.56.2 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

45.4.56.3 Fields

Field	Function
15-4 —	RESERVED
3-0 NOCOMB	No Combinational Path From Fault Input To PWM Output 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 enaough to get through the input filter and be latched in the fault logic. Setting these bits removes the combinational path and uses the filterred 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.

45.5 Functional Description

45.5.1 PWM Capabilities

This section describes some capabilities of the PWM module.

45.5.1.1 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 45-2.

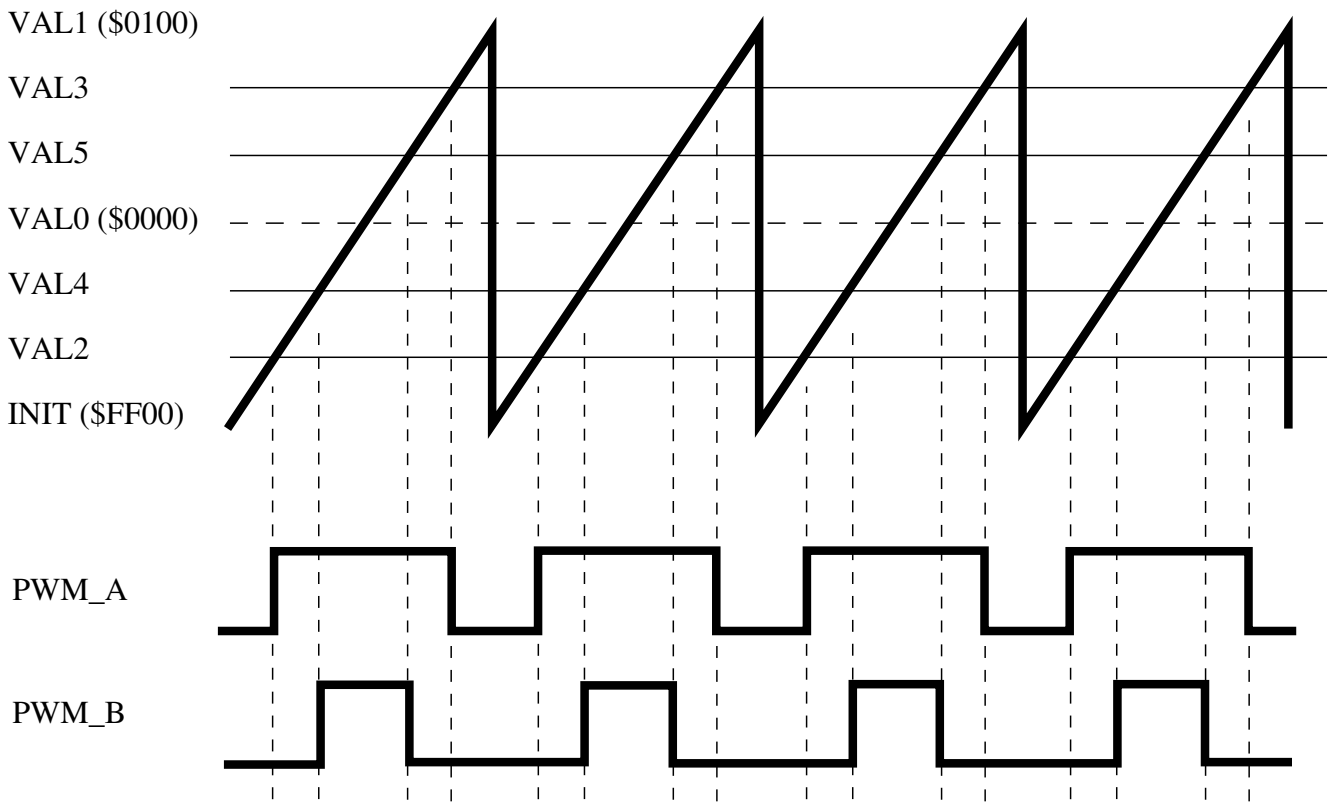


Figure 45-2. Center Aligned Example

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 45-2 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.

45.5.1.2 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.

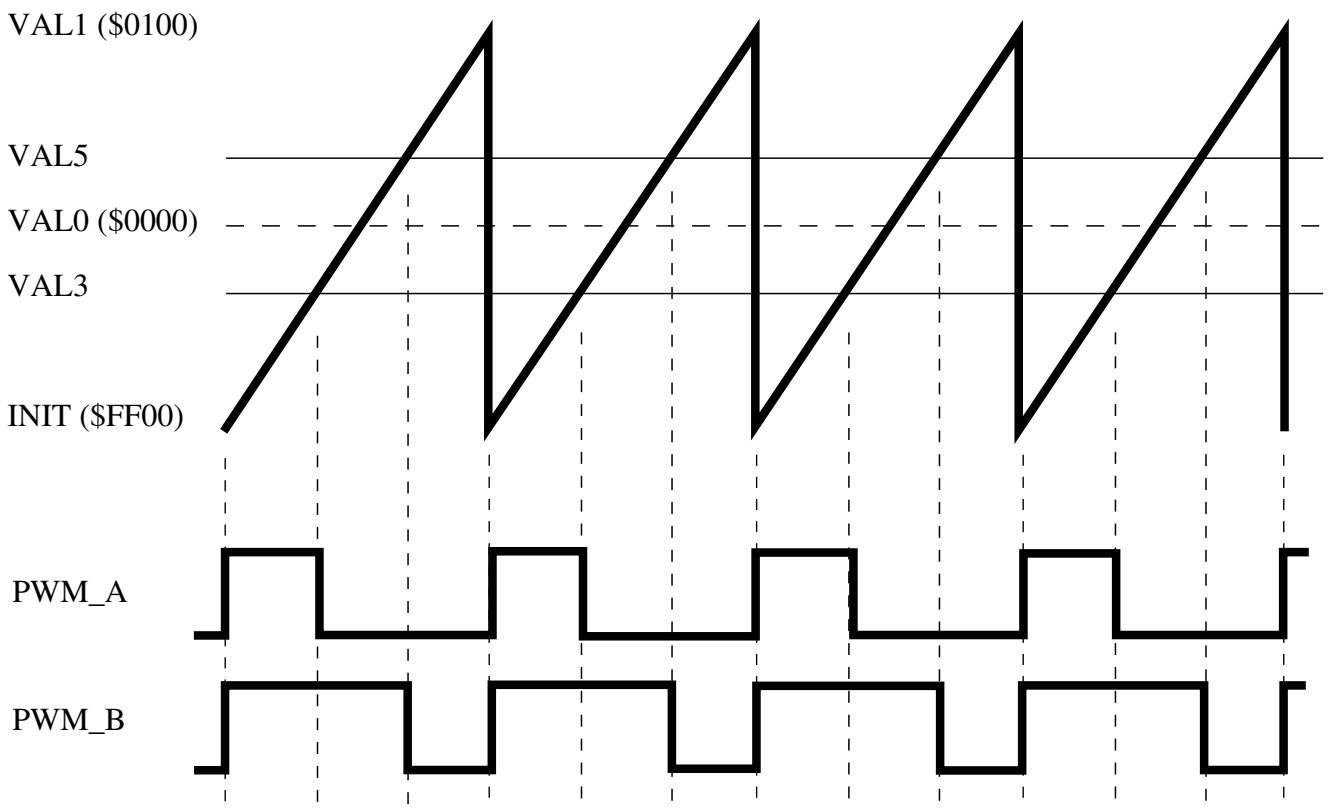


Figure 45-3. Edge Aligned Example (INIT=VAL2=VAL4)

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.

45.5.1.3 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.

If the outputs of submodules 1-3 need to be delayed from the output of submodule 0 (and from each other), instead of just creating a phase delay by adding an offset to the turn on and turn off times of the different submodules, another method is to use the PHASEDLY registers for submodules 1-3 to indicate their delay from the submodule 0 timing. This method can be used when the master sync signal from submodule 0 is selected as the initialization source (CTRL2[INIT_SEL]==b10). This method allows all of the submodules to be programmed with the same turn on and turn off times but submodules 1-3 can still be delayed in ime form submodule 0.

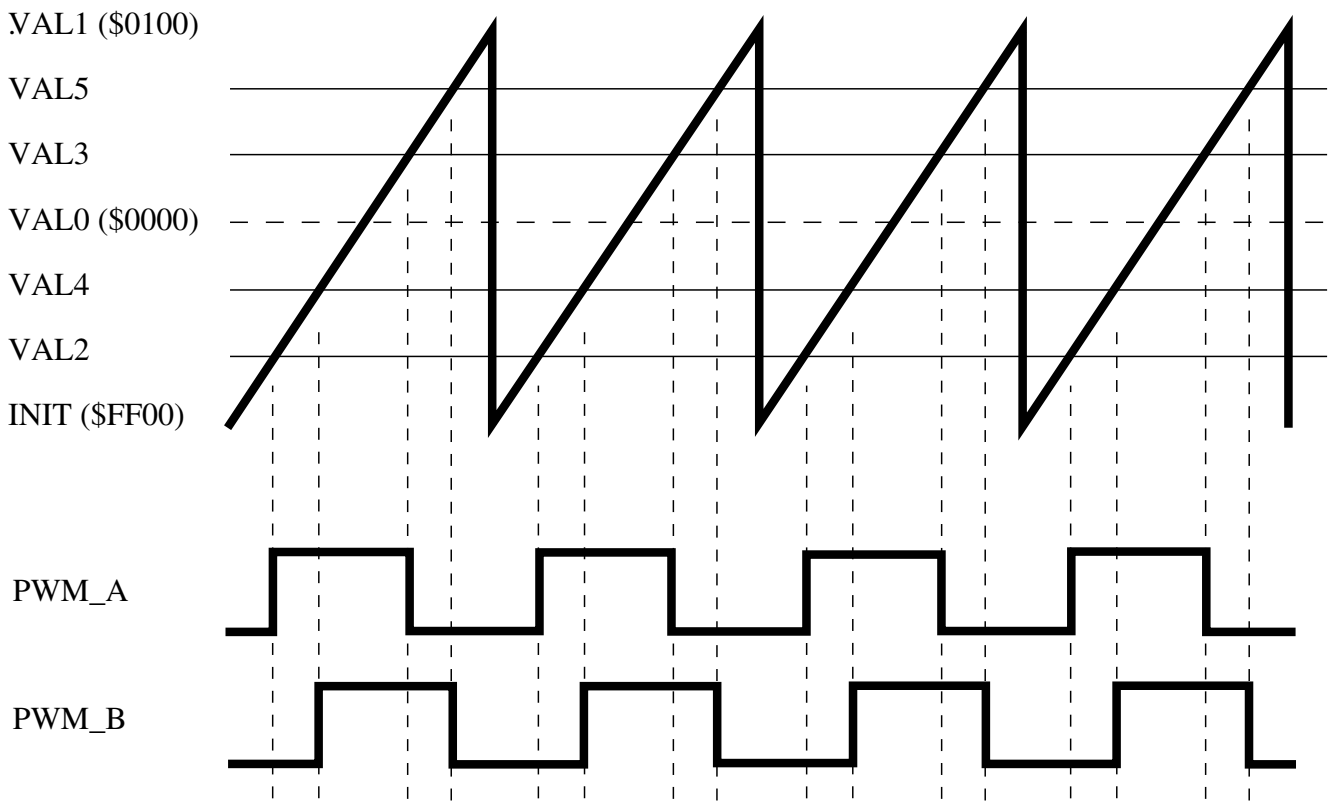


Figure 45-4. Phase Shifted Outputs Example

An additional benefit of phase shifted PWMs appears in Figure 45-5. 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.

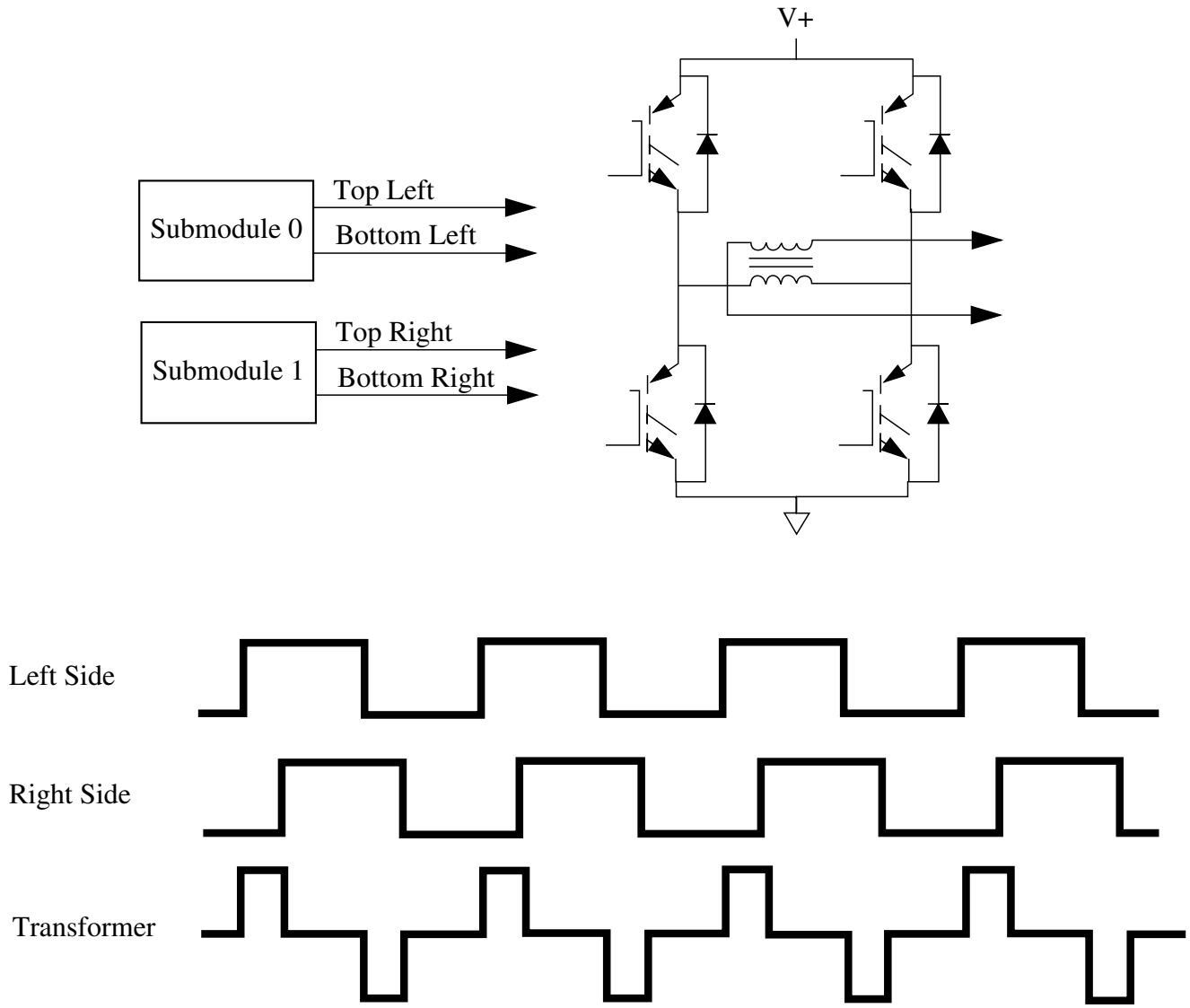


Figure 45-5. Phase Shifted PWMs Applied to a Transformer Primary

45.5.1.4 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.

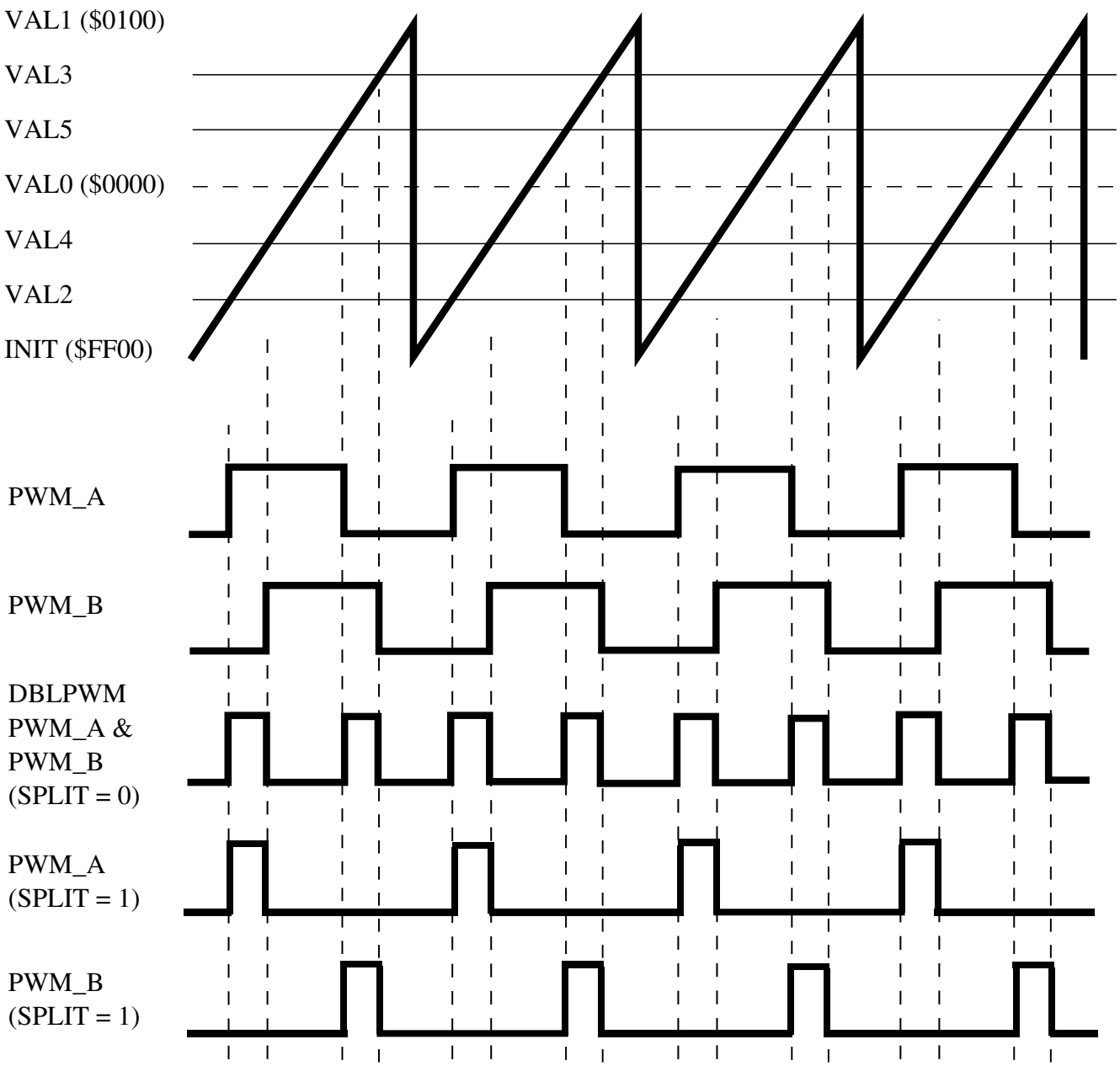


Figure 45-6. Double Switching Output Example

45.5.1.5 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 45-7 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.

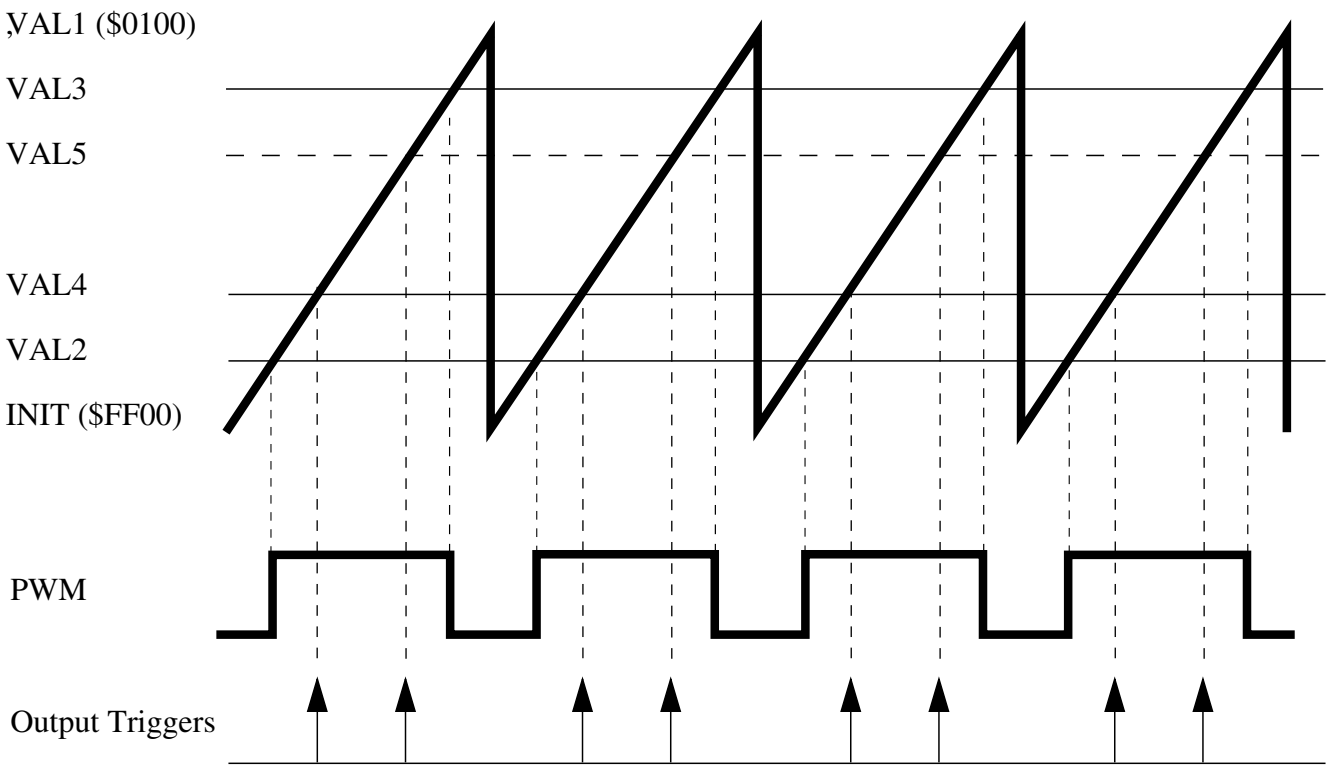


Figure 45-7. Multiple Output Trigger Generation in Hardware

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 45-8 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 45-8, *all* submodule comparators are shown being used for ADC trigger generation.

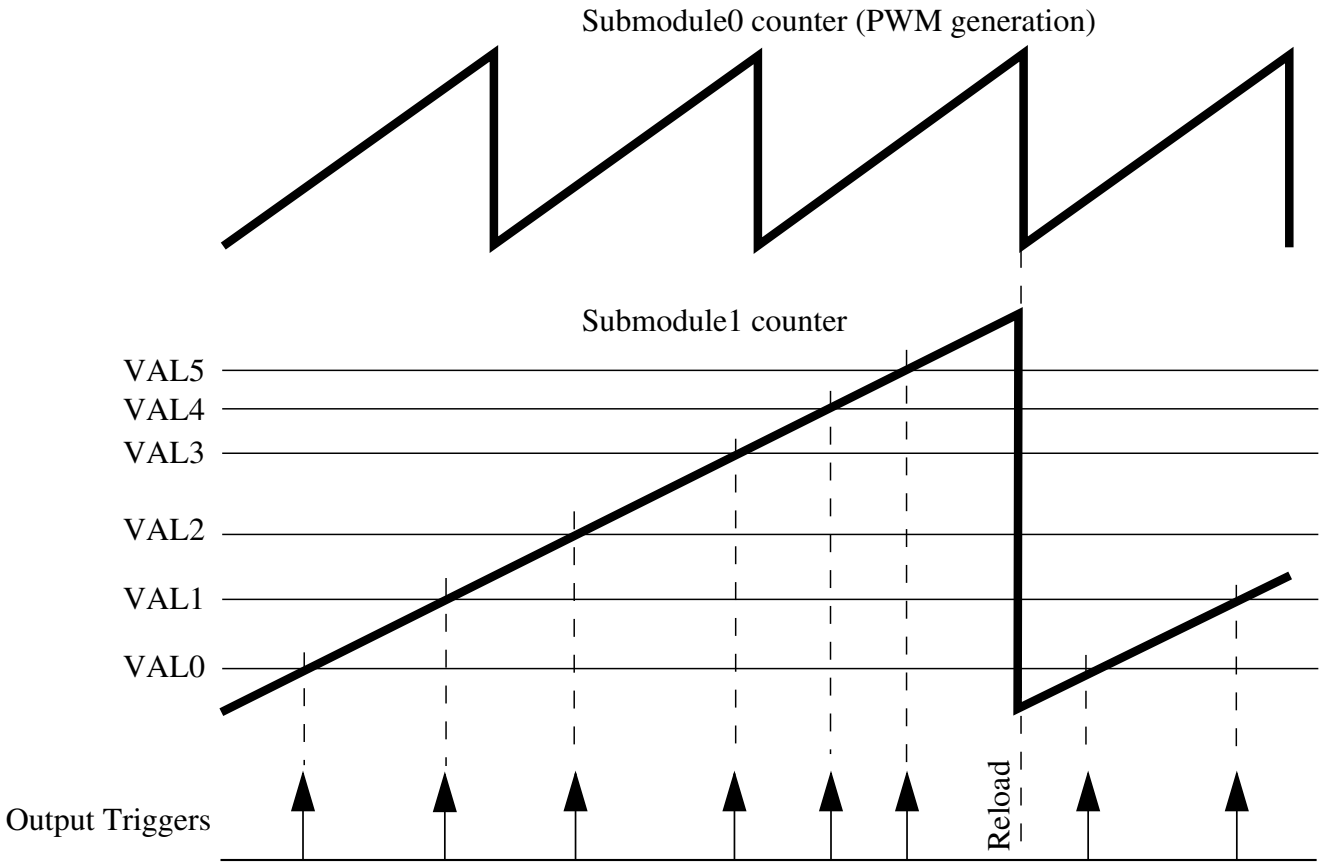


Figure 45-8. Multiple Output Triggers Over Several PWM Cycles

45.5.1.6 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.

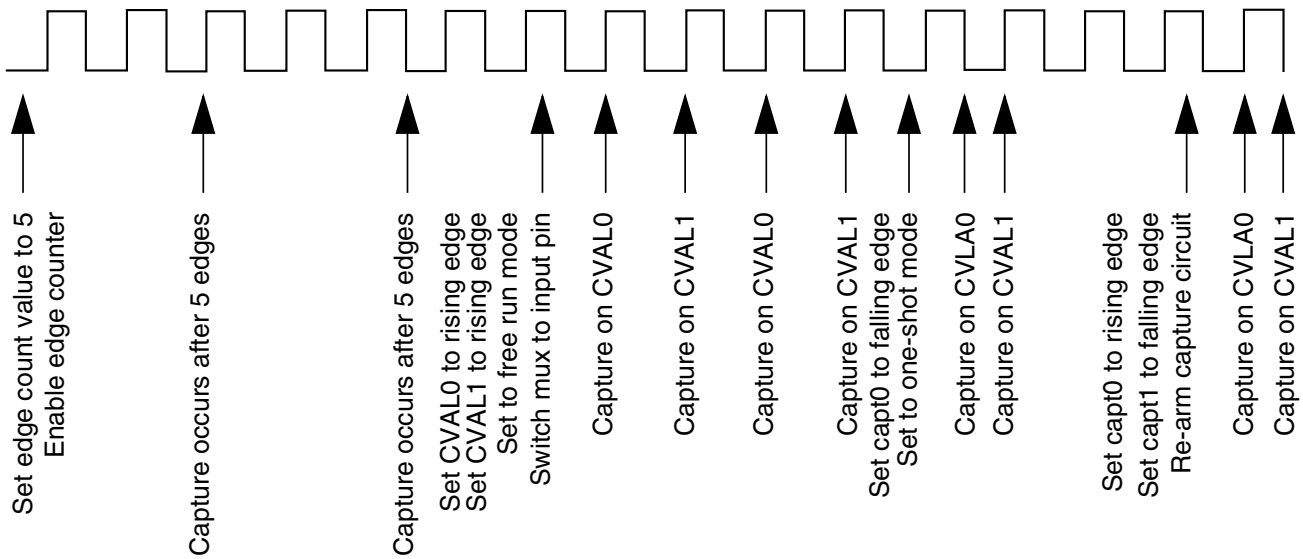


Figure 45-9. Capture Capabilities of the E-Capture Circuit

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.

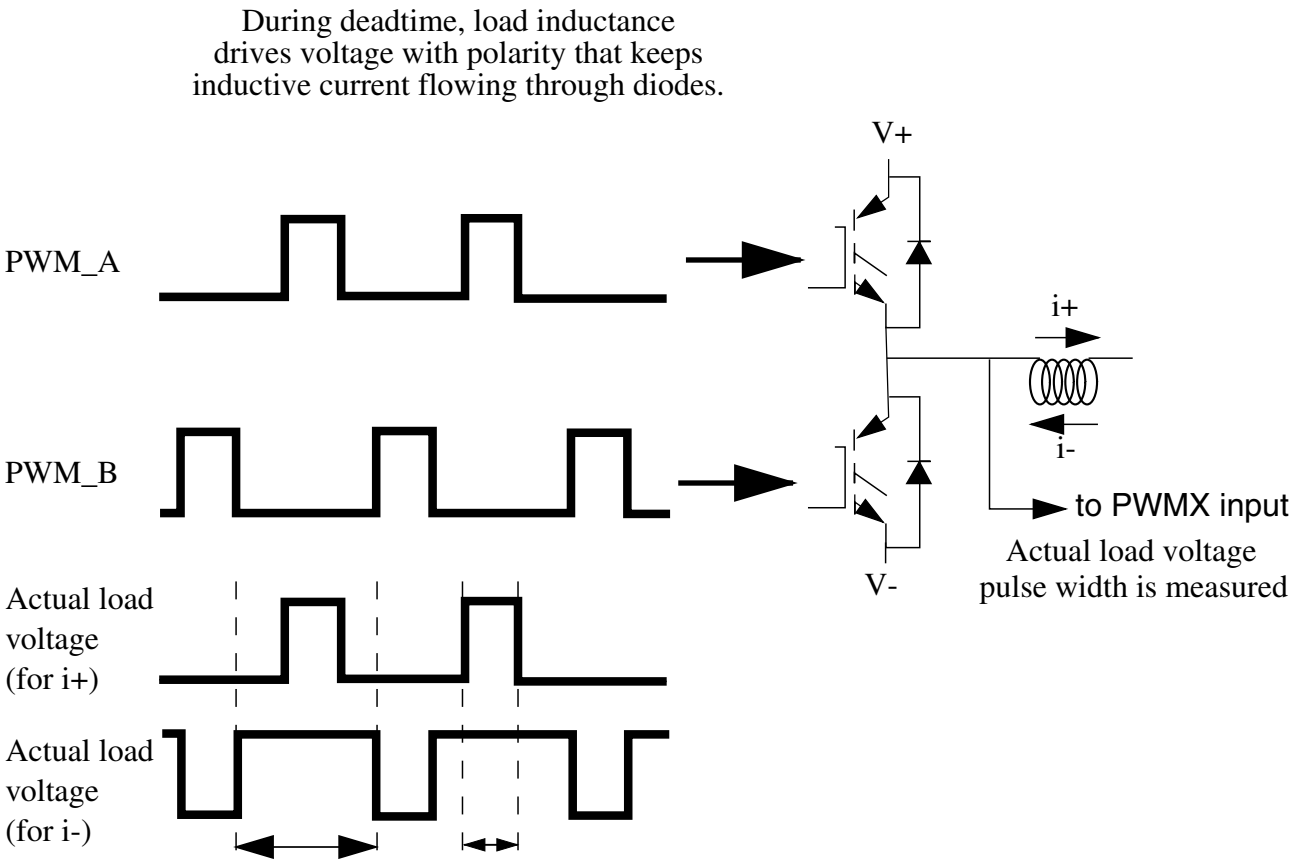


Figure 45-10. Output Pulse Width Measurement Possible with the E-Capture Circuit

45.5.1.7 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 45-11 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 45-11 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.

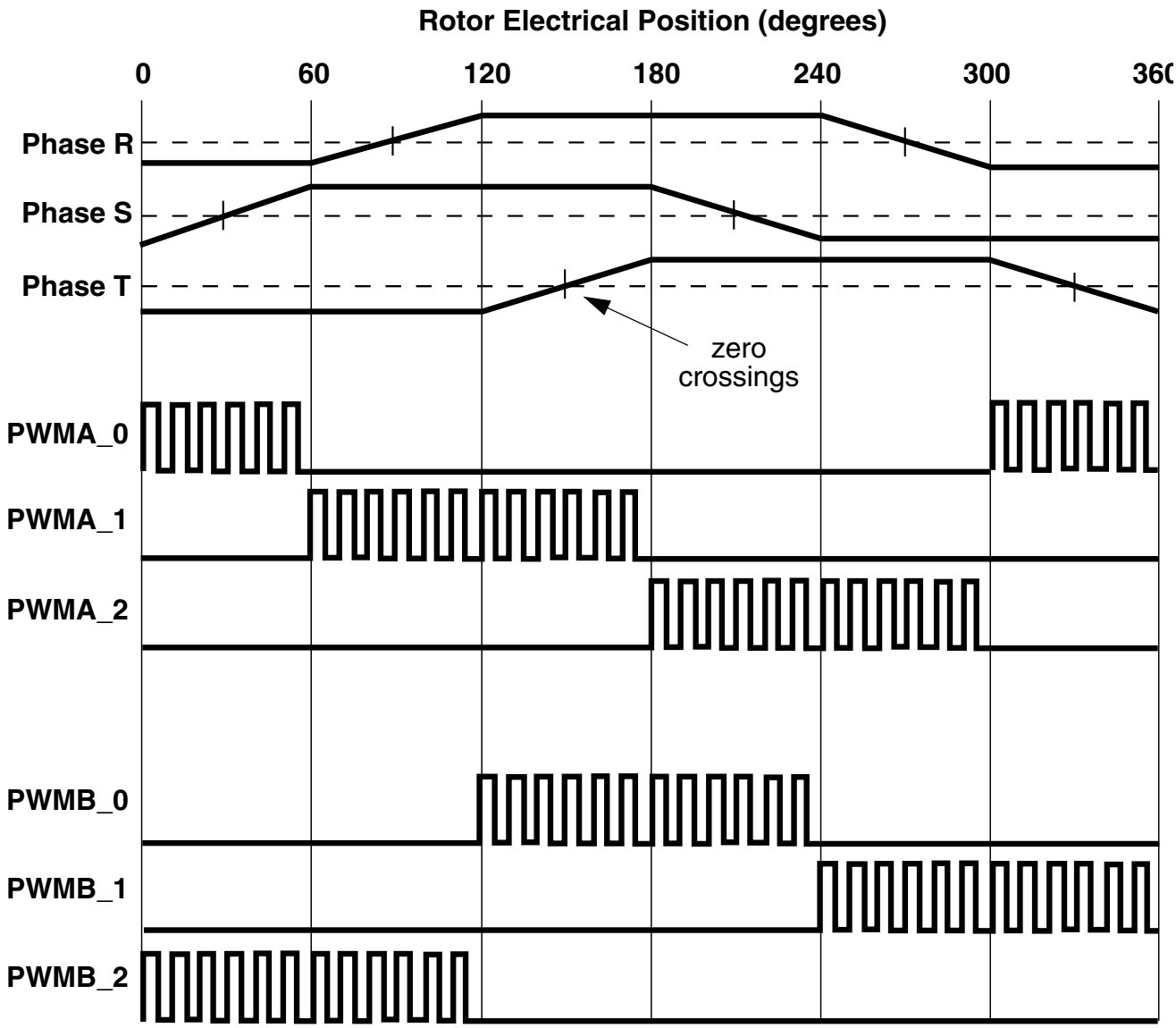


Figure 45-11. Sensorless BLDC Commutation Using the Force Out Function

45.5.2 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.

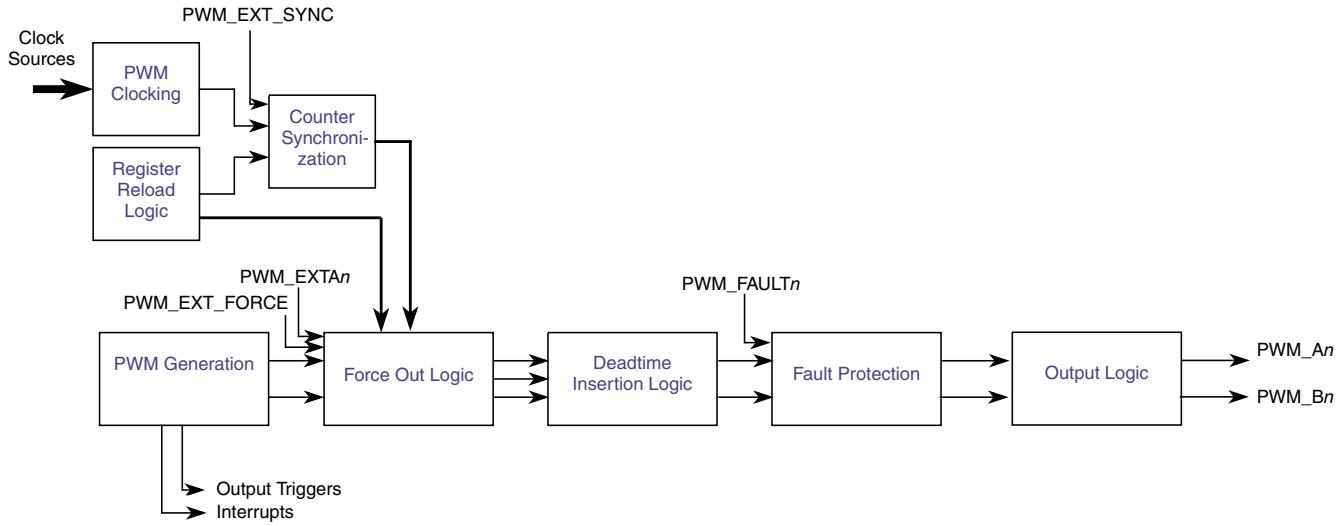


Figure 45-12. High-Level Output PWM Generation Block Diagram

45.5.2.1 PWM Clocking

Figure 45-13 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.

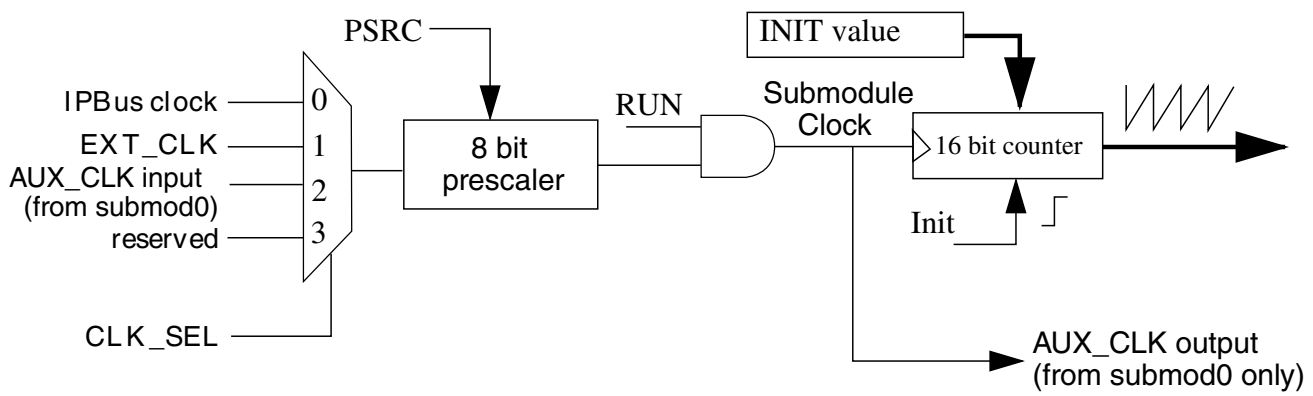


Figure 45-13. Clocking Block Diagram for Each PWM Submodule

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.

45.5.2.2 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 45-14 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.

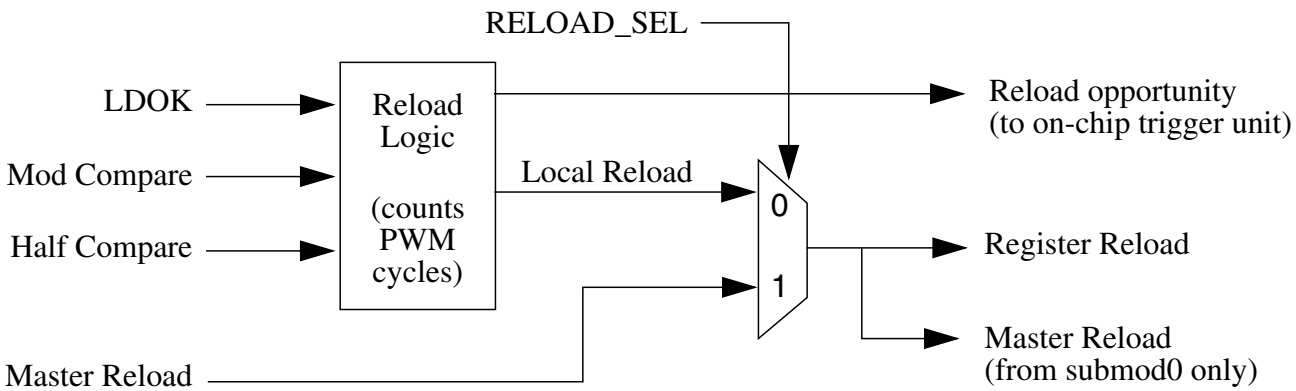


Figure 45-14. Register Reload Logic

45.5.2.3 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.

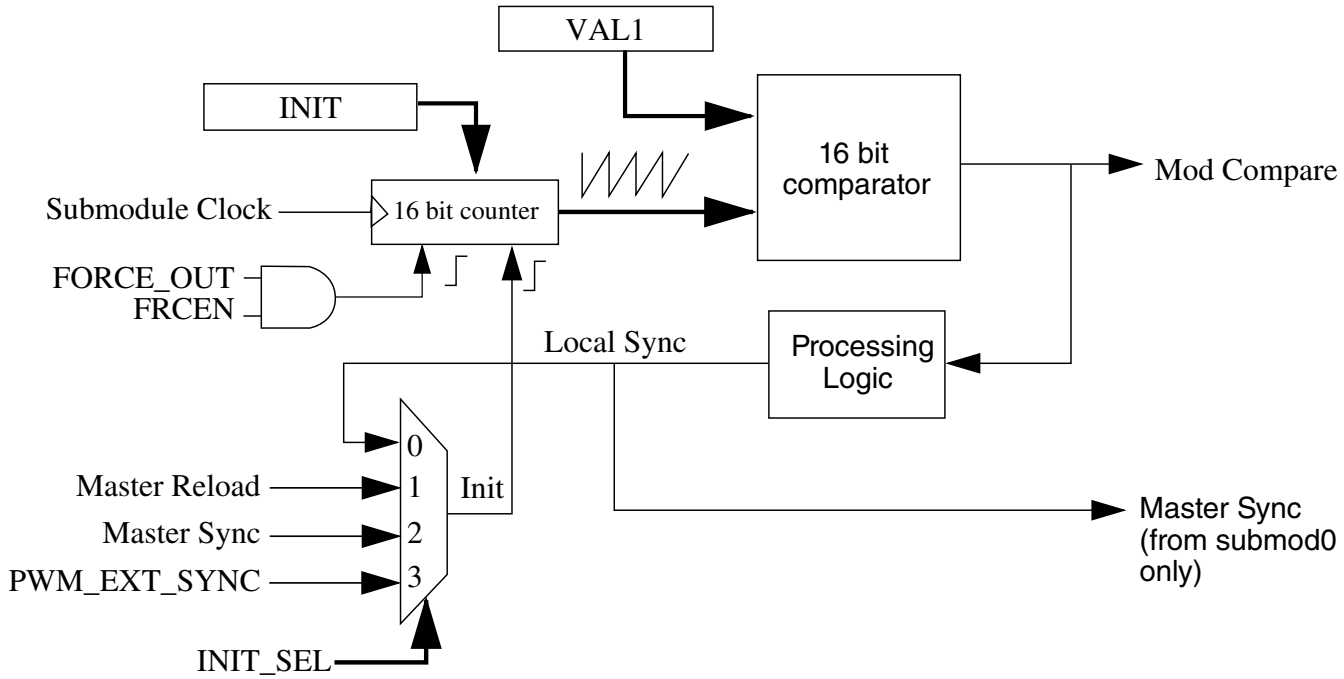


Figure 45-15. Submodule Timer Synchronization

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.

45.5.2.4 PWM Generation

Figure 45-16 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.

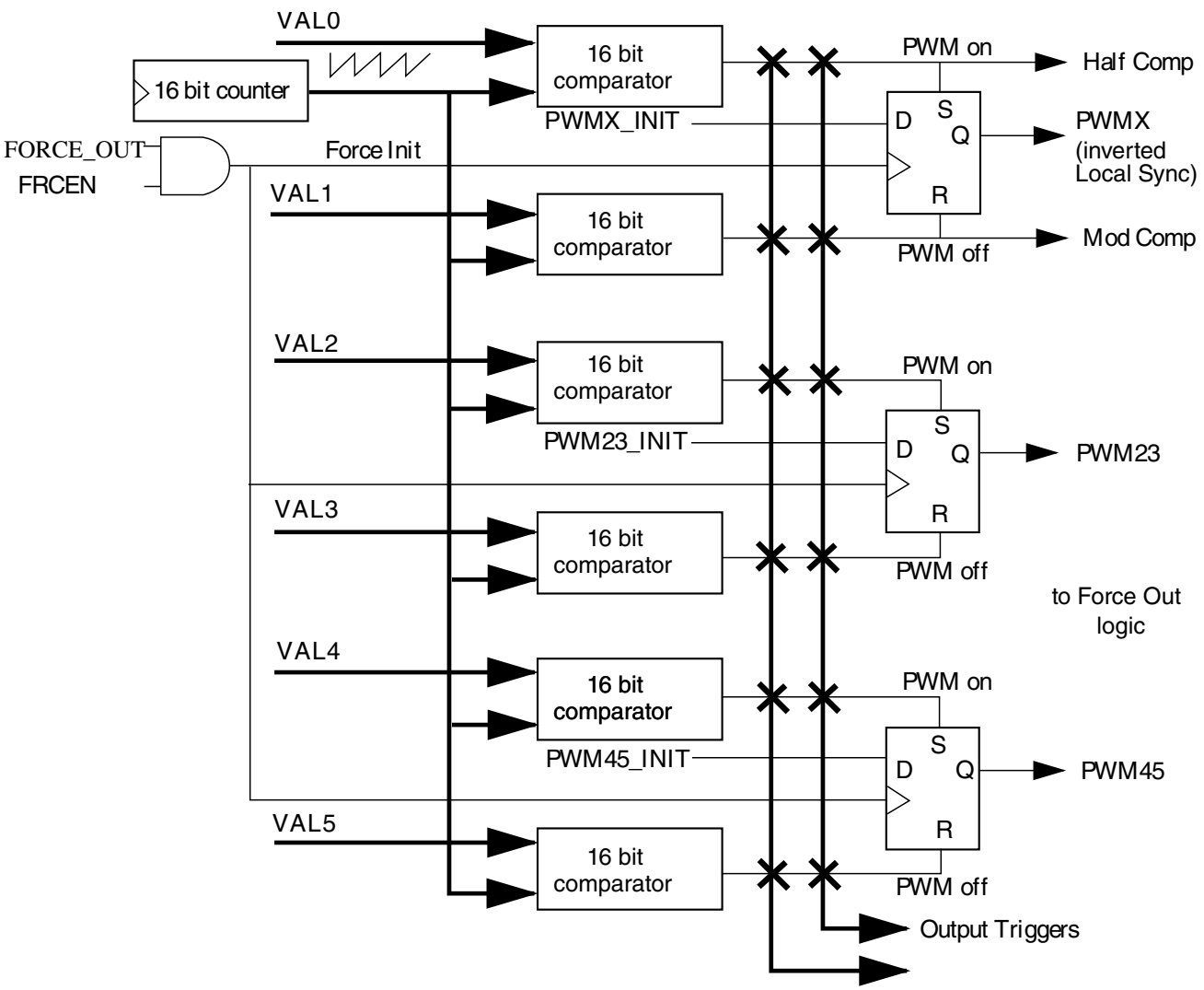


Figure 45-16. PWM Generation Hardware

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 45-16 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.

45.5.2.5 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.

45.5.2.6 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 45-17 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.

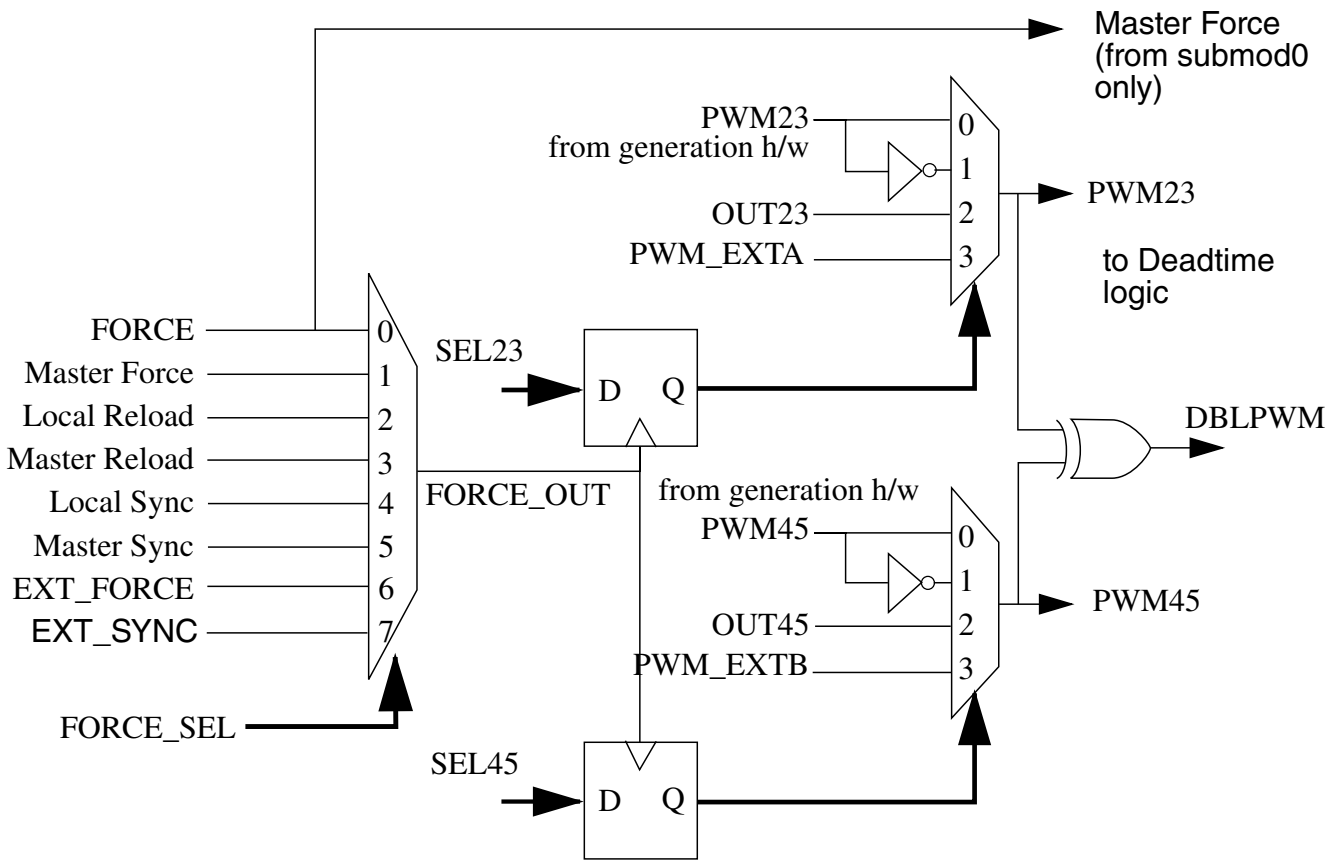


Figure 45-17. Force Out Logic

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.

45.5.2.7 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 45-18 in complementary channel operation. Which signal is connected to the output pin (PWM23 or PWM45) is determined by MCTRL[IPOL].

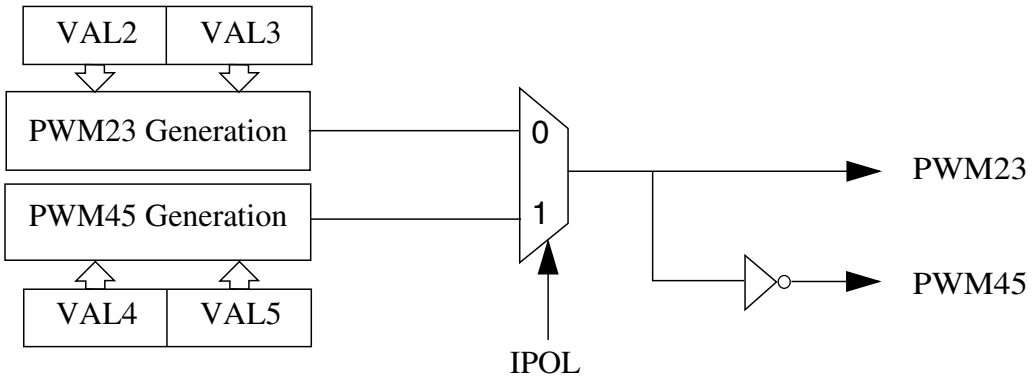


Figure 45-18. Complementary Channel Pair

The complementary channel operation is for driving top and bottom transistors in a motor drive circuit, such as the one in Figure 45-19.

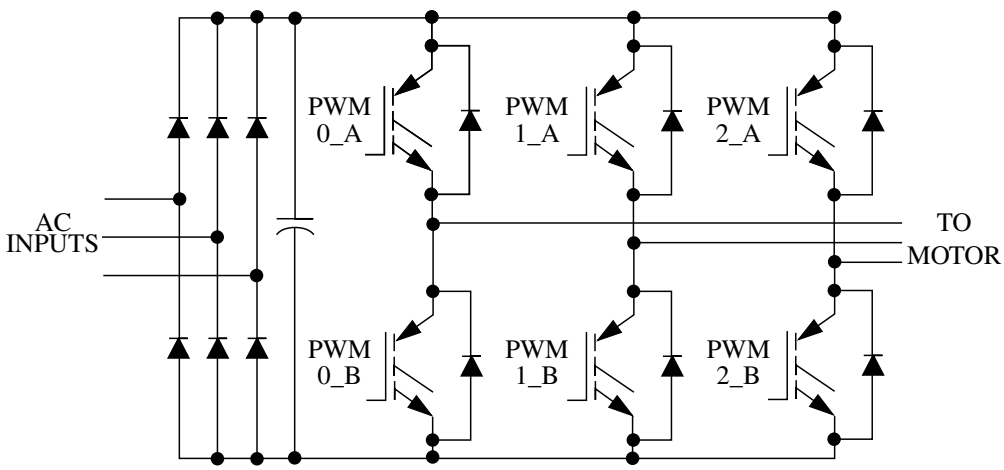


Figure 45-19. Typical 3 Phase AC Motor Drive

Complementary operation allows the use of the deadtime insertion feature.

45.5.2.8 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.

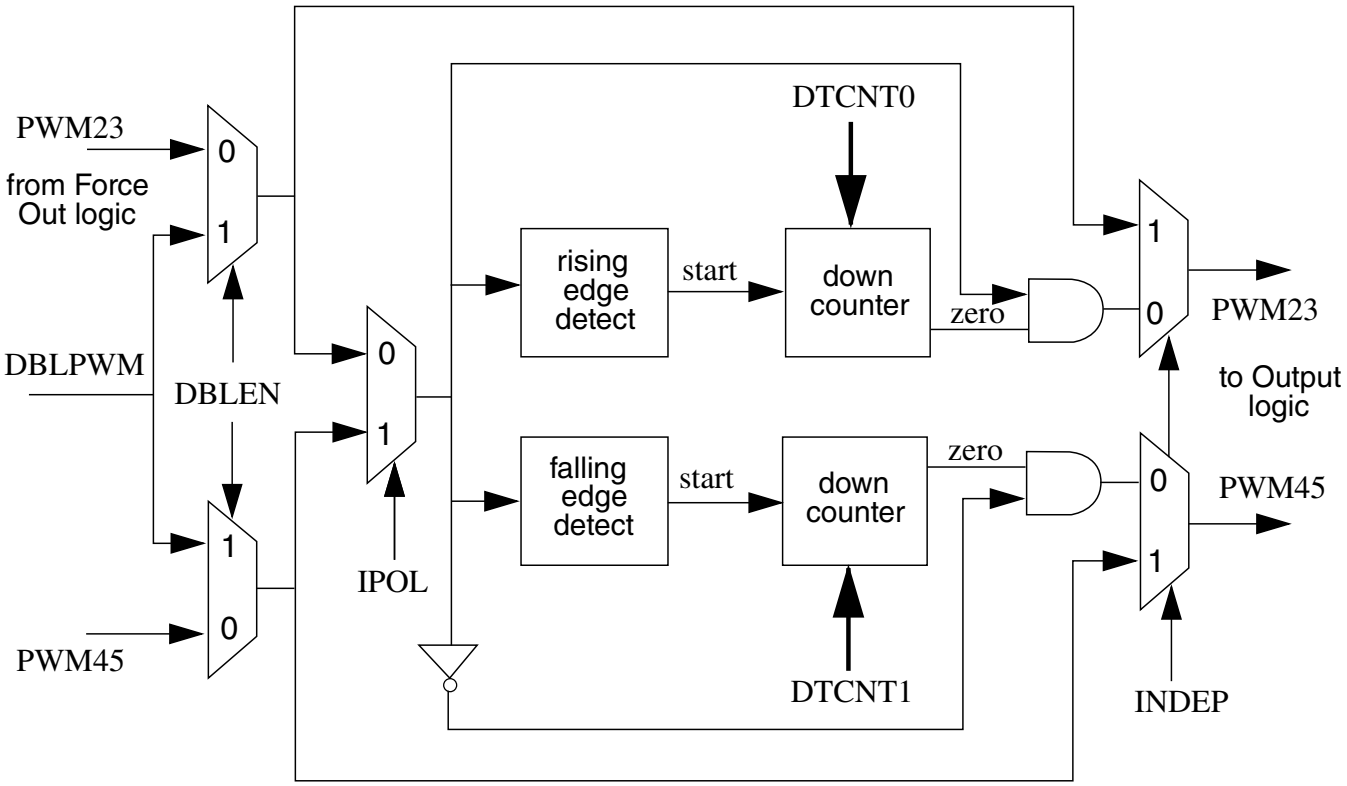


Figure 45-20. Deadtime Insertion Logic

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.

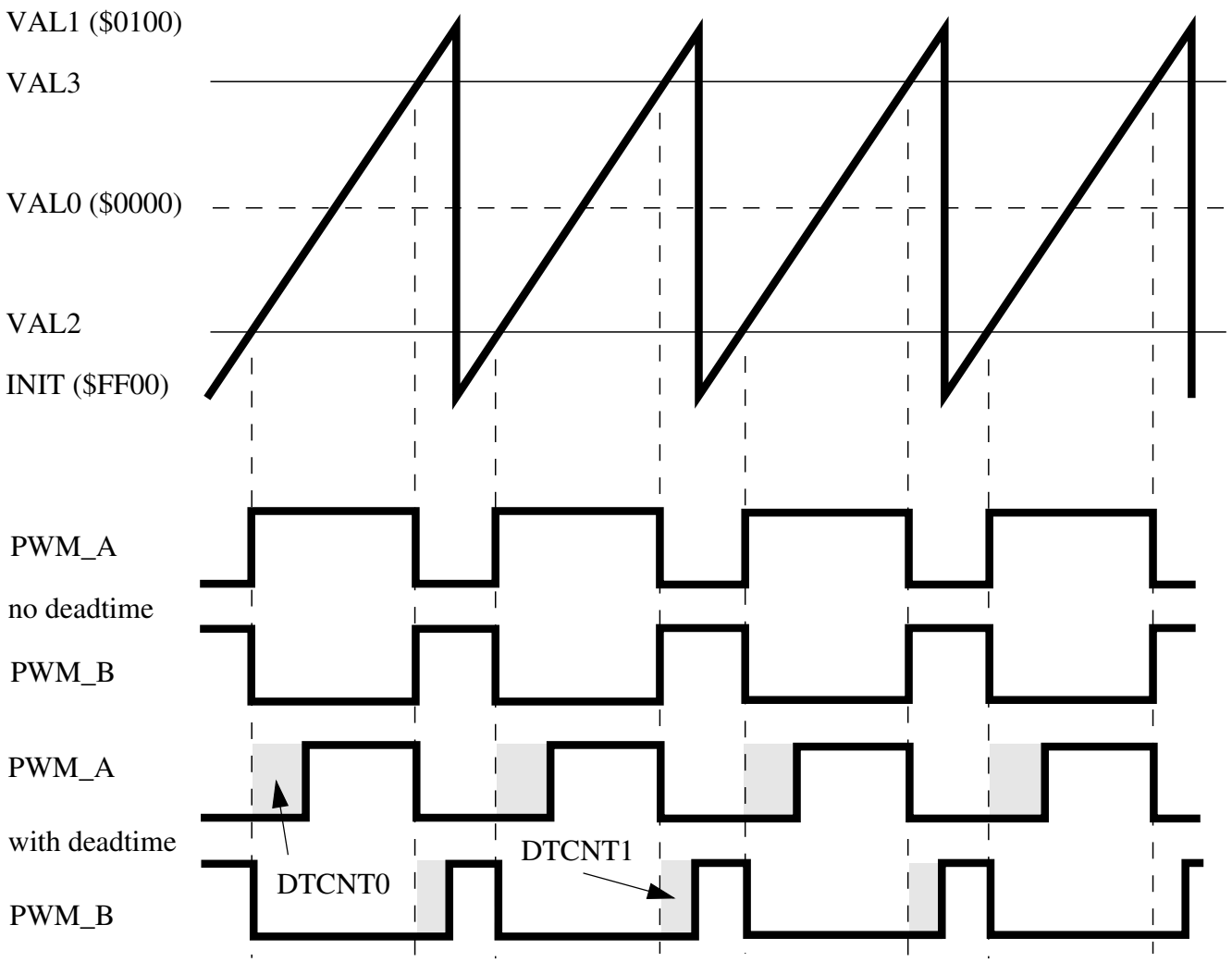


Figure 45-21. Deadtime Insertion

45.5.2.8.1 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.

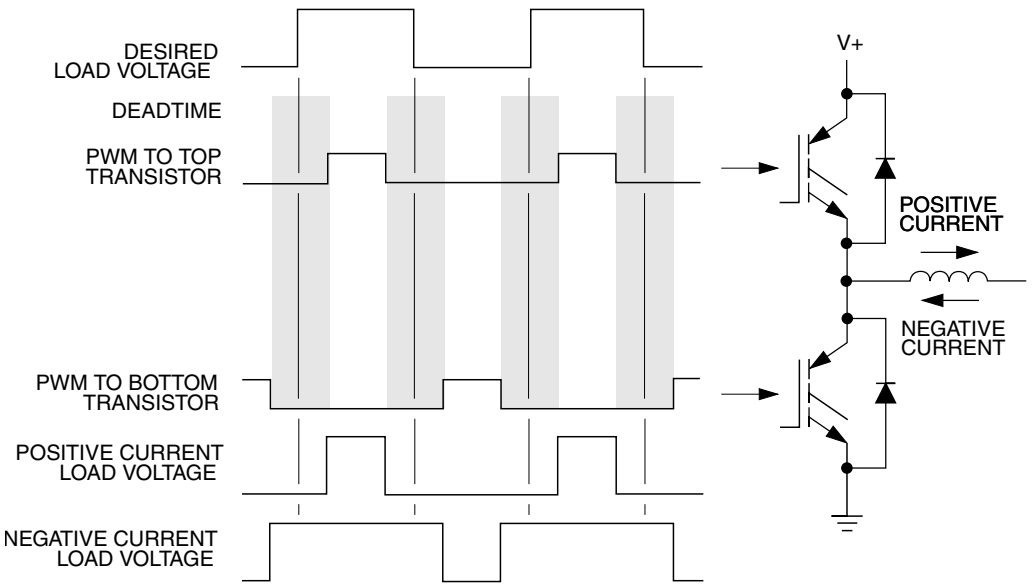


Figure 45-22. Deadtime Distortion

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.

45.5.2.8.2 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.

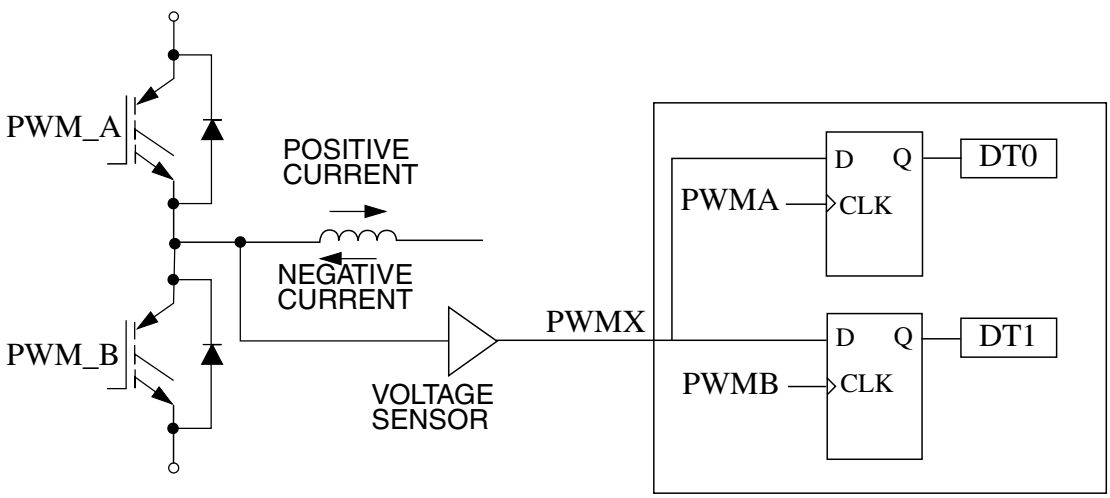


Figure 45-23. Current-status Sense Scheme for Deadtime Correction

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.**

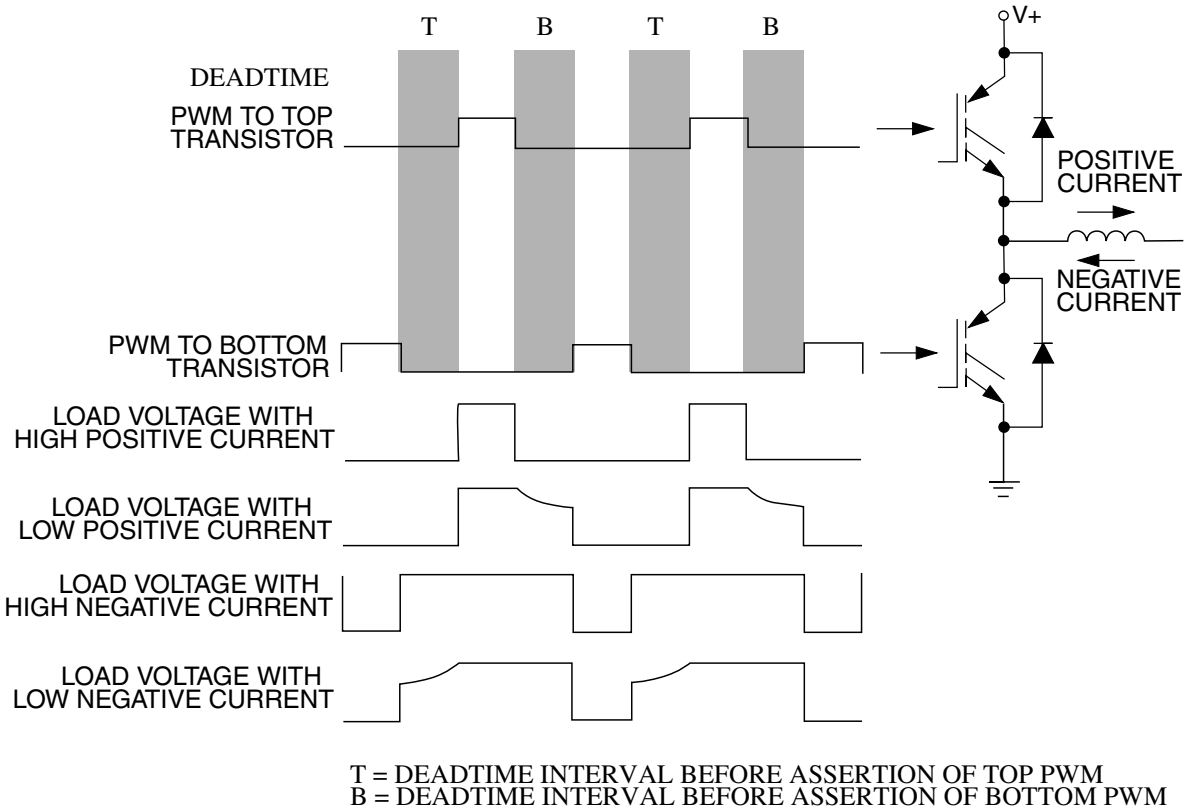


Figure 45-24. Output Voltage Waveforms

45.5.2.9 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.

45.5.2.9.1 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.

45.5.2.9.2 Fractional Delay Logic without NanoEdge Placement Block

For submodules that are not supported by the NanoEdge placer, the PWM can use dithering to simulate fine edge control. Enable this feature by setting the FRCTRL[FRAC1_EN], FRCTRL[FRAC23_EN], and FRCTRL[FRAC45_EN] bits. It is unnecessary to set FRCTRL[FRAC_PU]. The PWM period or the PWM edges will dither from the nearest whole number values to achieve an average value that is equivalent to the programmed fractional value. The added cycles are based on the accumulation of the fractional component. For example, if you want the PWM period to be 50.25 clock cycles, then program VAL1 with 0x0032 and FRACVAL1 with 0x4000. The PWM period will be 50 cycles long most of the time, but will occasionally be 51 cycles long to achieve a long-term average of 50.25 cycles.

In submodules that are not supported by a NanoEdge placer, the clock frequency is not required to be any specific value to achieve proper operation.

45.5.2.10 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.

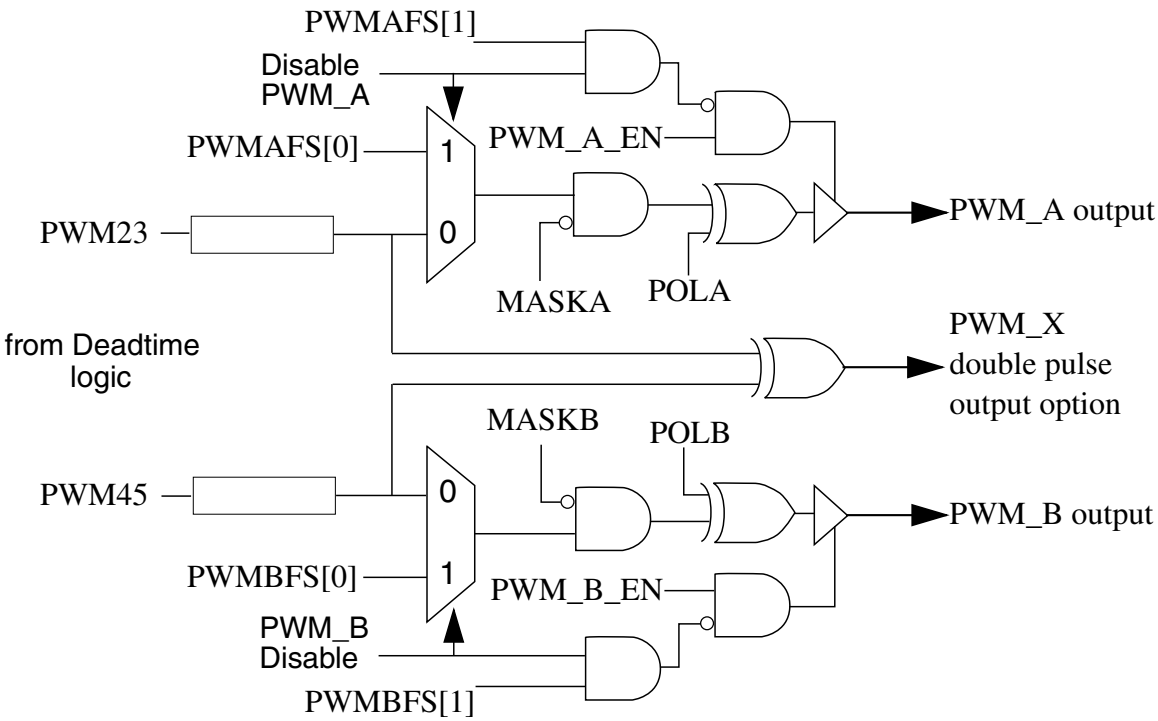


Figure 45-25. Output Logic

45.5.2.11 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.

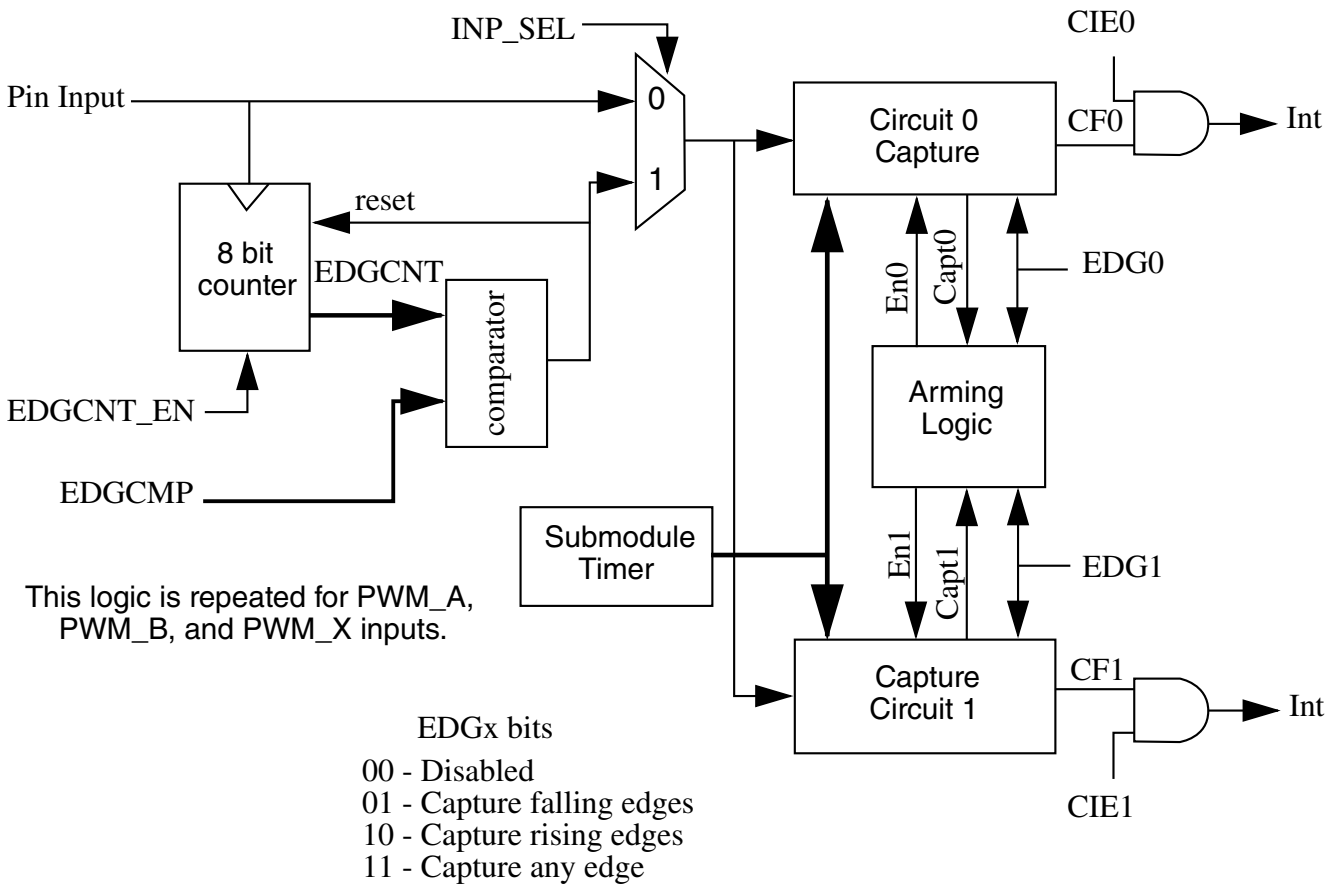


Figure 45-26. Enhanced Capture (E-Capture) Logic

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.

45.5.2.12 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.

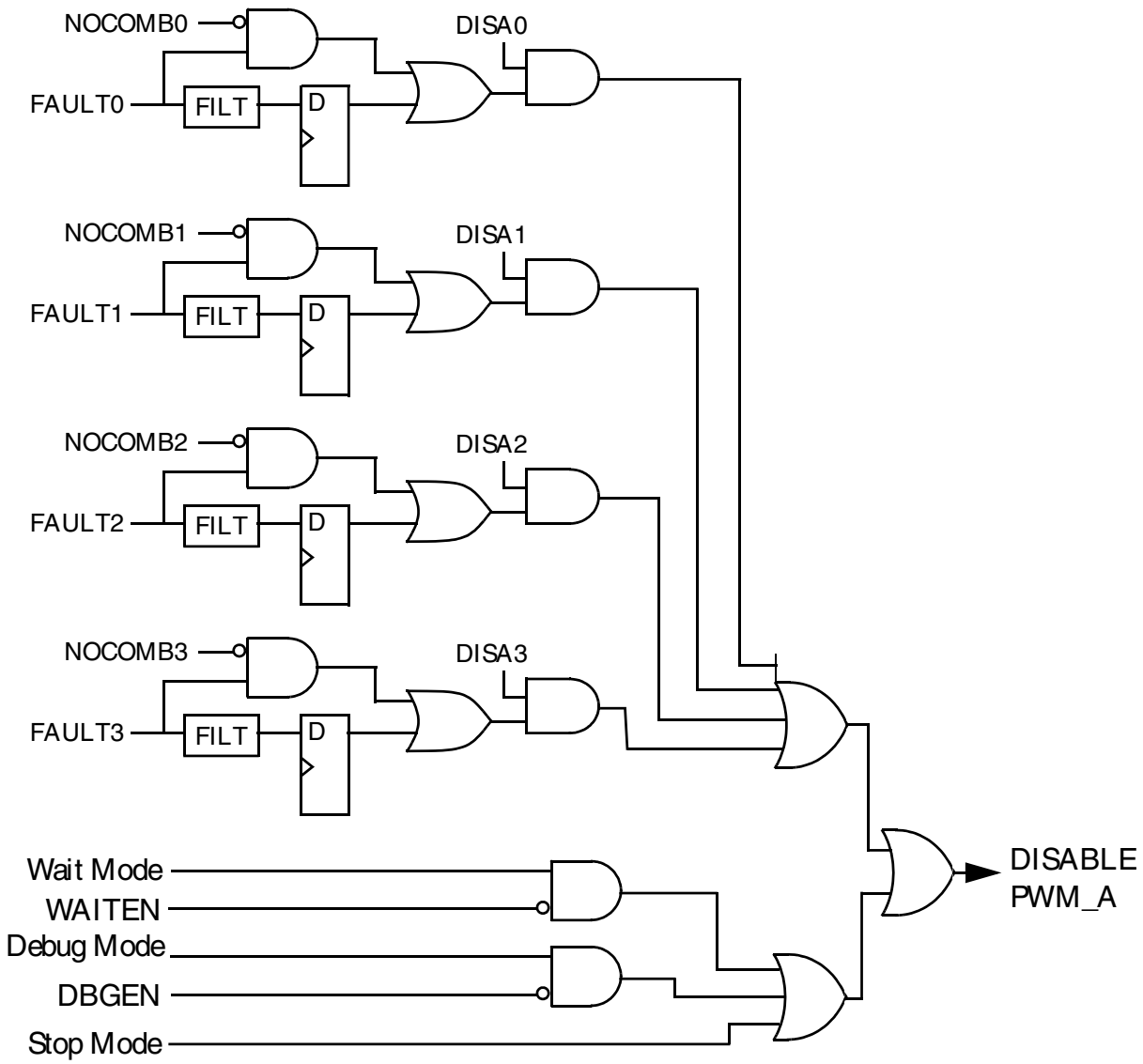


Figure 45-27. Fault Decoder for PWM_A

Table 45-3. Fault Mapping

PWM Pin	Controlling Register Bits
PWM_A	DISMAP0[DIS0A] and DISMAP1[DIS1A]
PWM_B	DISMAP0[DIS0B] and DISMAP1[DIS1B]
PWM_X	DISMAP0[DIS0X] and DISMAP1[DIS1X]

45.5.2.12.1 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.

45.5.2.12.2 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.

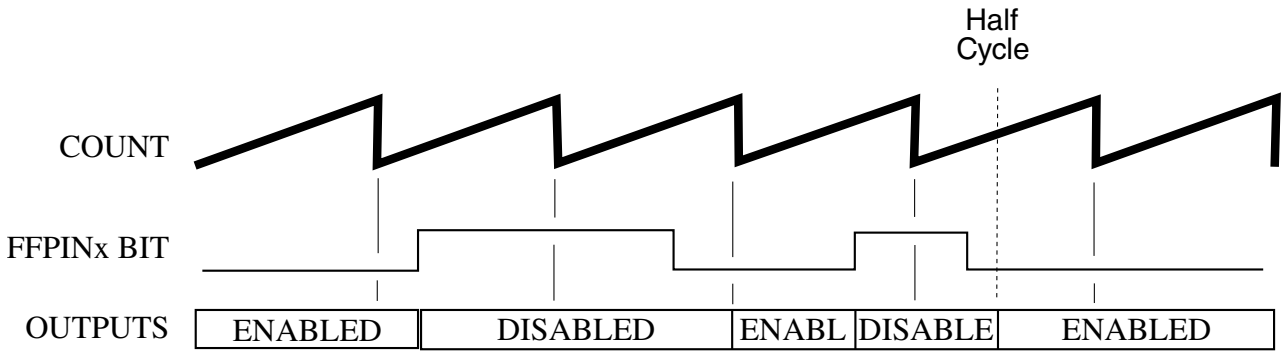


Figure 45-28. Automatic Fault Clearing

45.5.2.12.3 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.

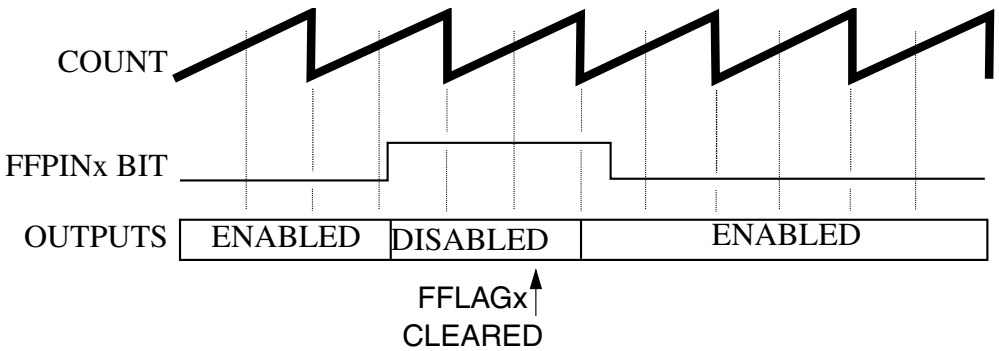


Figure 45-29. Manual Fault Clearing (FCTRL[FSAFE]=0)

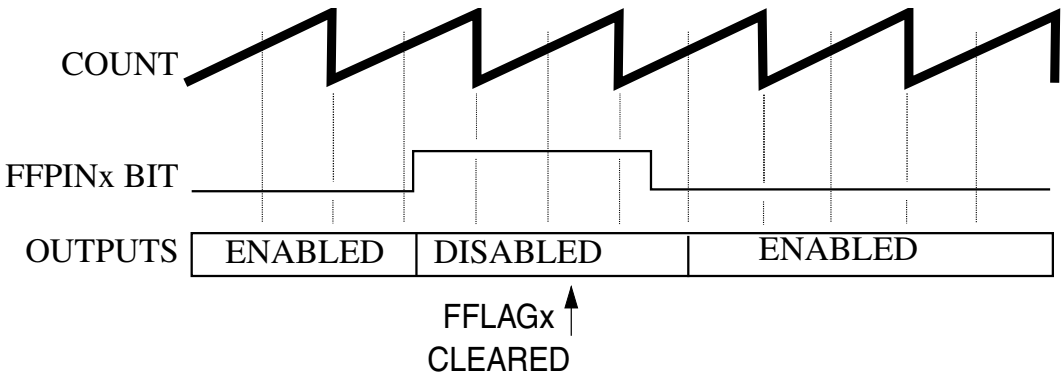


Figure 45-30. Manual Fault Clearing (FCTRL[FSAFE]=1)

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.

45.5.2.12.4 Fault Testing

FTST[FTEST] is used to simulate a fault condition on each of the fault inputs within that fault channel.

45.5.3 PWM Generator Loading

45.5.3.1 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 transfered to the inner set of registers immediately upon setting MCTRL[LDOK] if CTRL[LDMOD] is set. After loading, MCTRL[LDOK] is automatically cleared.

45.5.3.2 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.

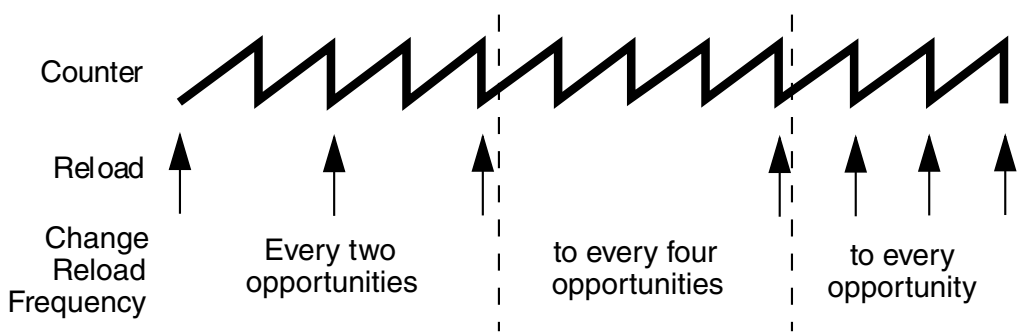


Figure 45-31. Full Cycle Reload Frequency Change

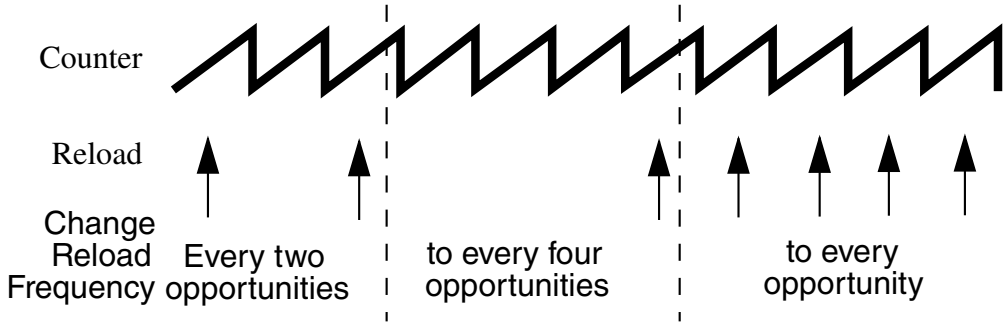


Figure 45-32. Half Cycle Reload Frequency Change

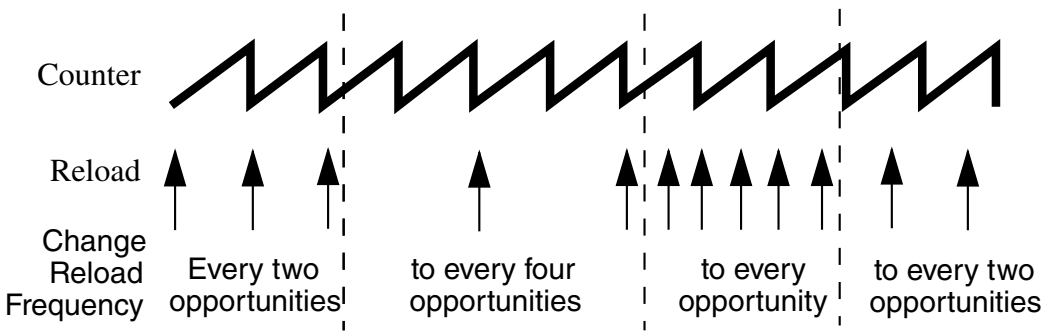


Figure 45-33. Full and Half Cycle Reload Frequency Change

45.5.3.3 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.

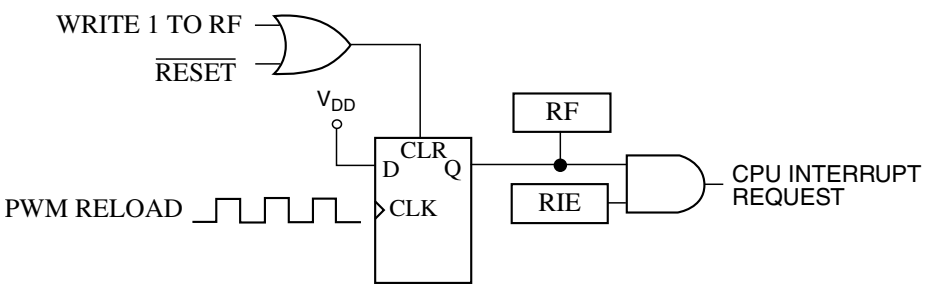


Figure 45-34. PWMF Reload Interrupt Request

45.5.3.4 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.

45.5.3.5 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

45.6 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.

45.7 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 45-4. 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], SM0STS[CFX0]	SM0INTEN[CFA1IE], SM0INTEN[CFA0IE], SM0INTEN[CFB1IE], SM0INTEN[CFB0IE], SM0INTEN[CFX1IE], SM0INTEN[CFX0IE]	Submodule 0 input capture interrupt	Input capture event has occurred
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

Table continues on the next page...

Table 45-4. Interrupt Summary (continued)

Core Interrupt	Interrupt Flag	Interrupt Enable	Name	Description
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	
PWM_FAULT	FSTS[FFLAG]	FCTRL[FIE]	Fault input interrupt	Fault condition has been detected

45.8 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 45-5. 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

Table continues on the next page...

Table 45-5. DMA Summary (continued)

DMA Request	DMA Enable	Name	Description
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
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

Table continues on the next page...

Table 45-5. DMA Summary (continued)

DMA Request	DMA Enable	Name	Description
	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

Chapter 46
Watchdog Timer (WDOG1-2)

46.1 Chip-specific WDOG 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 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.

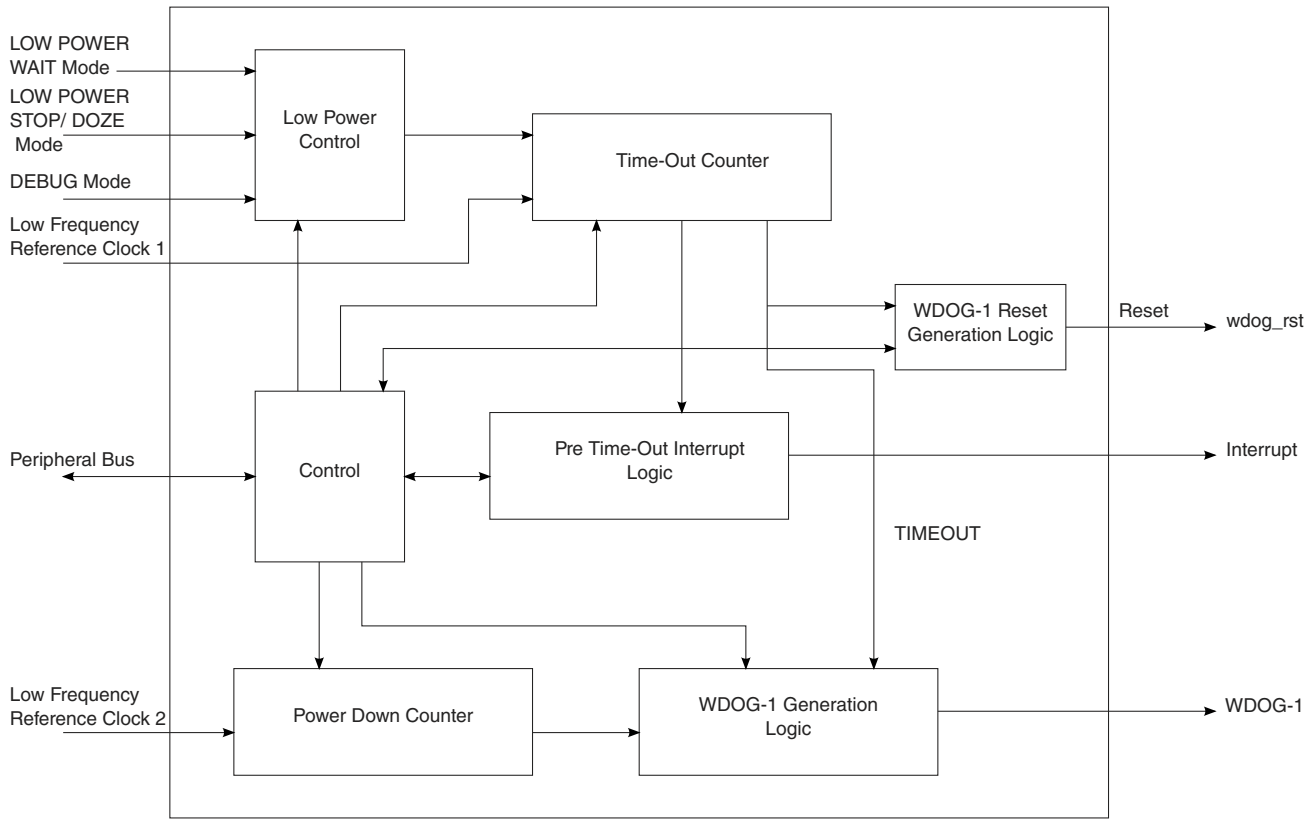


Figure 46-1. WDOG Diagram

NOTE

WDOG-1 means WDOG_B when it presents on a PAD.

46.2.1 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.

46.3 External signals

Table 46-2. 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.

46.4 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 46-3. 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.

46.5 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.

46.6 Functional description

This section provides a complete functional description of the block.

46.6.1 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.

46.6.1.1 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.

46.6.2 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.

46.6.3 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.

46.6.4 Low power modes

46.6.4.1 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.

46.6.4.2 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.

46.6.5 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.

46.6.6 Operations

46.6.6.1 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 46-3 shows the timing diagram of this signal due to a timeout condition.

46.6.6.2 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 46-4 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 46-2 shows the scenarios under which WDOG_B($\overline{\text{ipp\_wdog}}$) gets asserted.

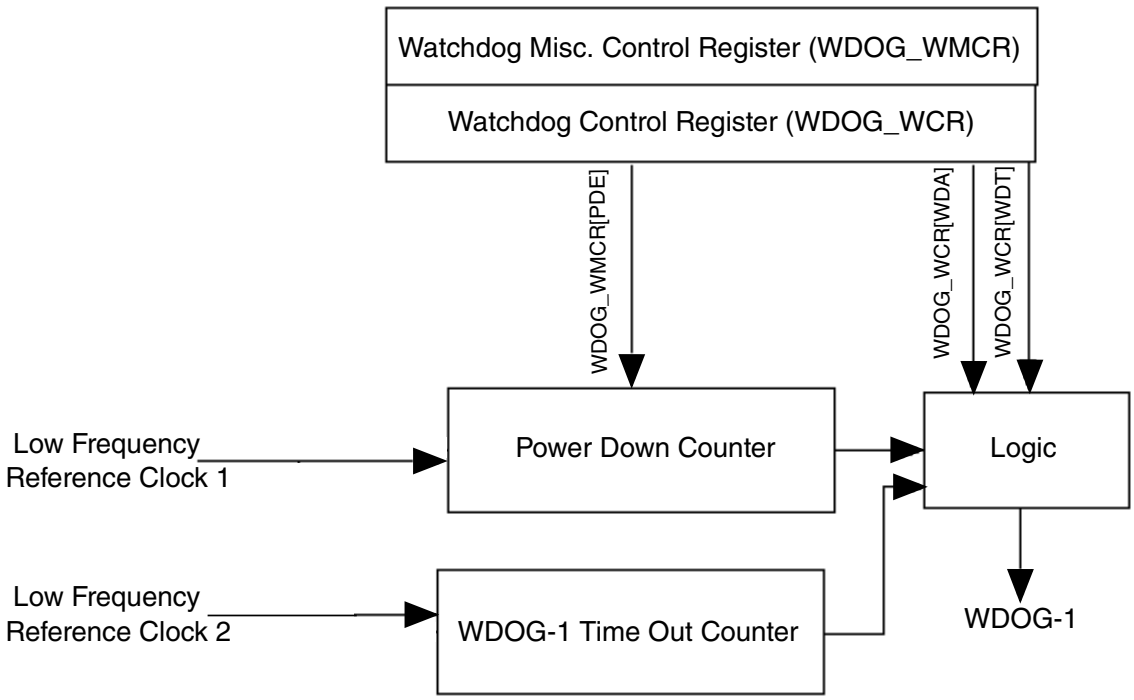


Figure 46-2. WDOG_B (ipp_wdog) generation

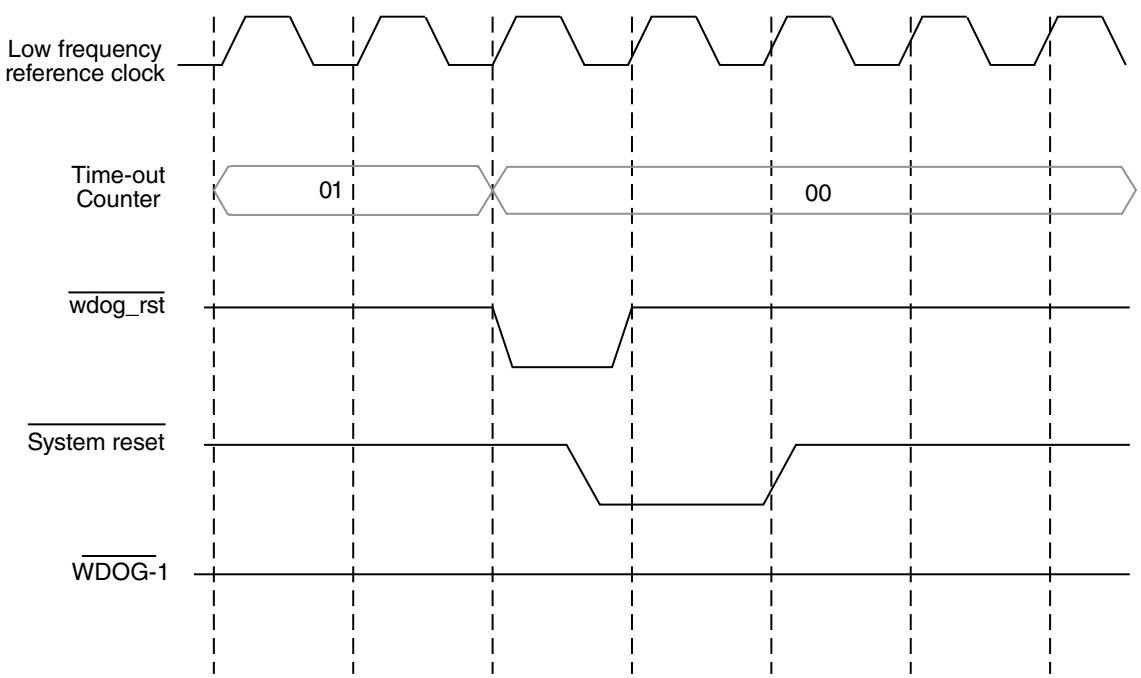


Figure 46-3. WDOG timeout condition/WDT bit is not set

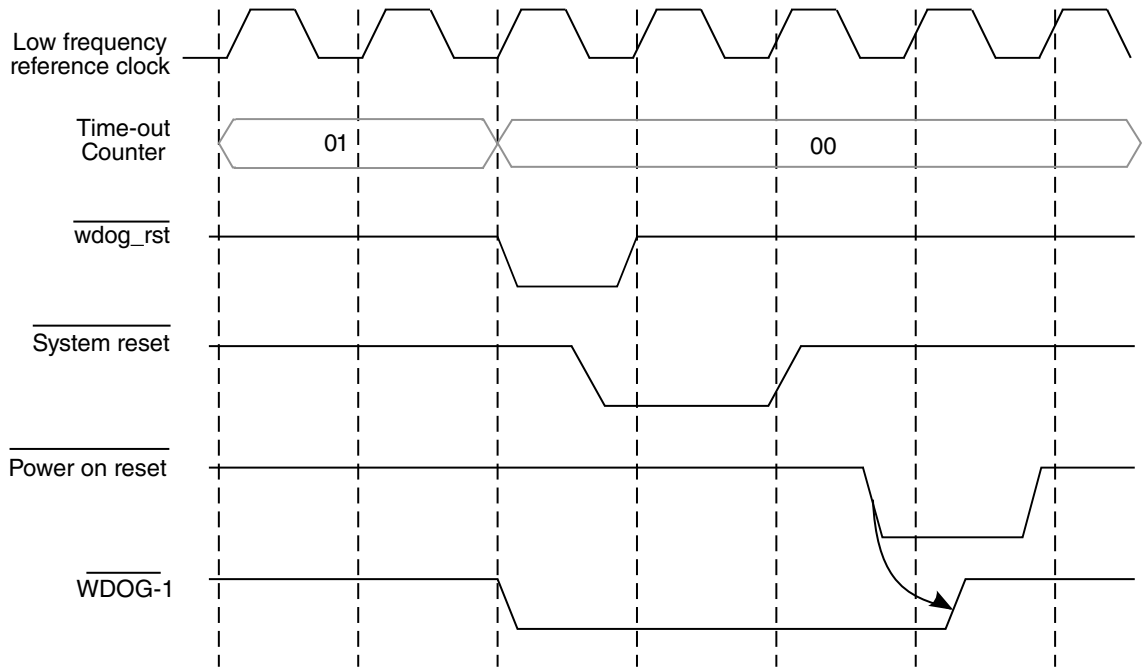


Figure 46-4. WDOG timeout condition/WDT bit is set

46.6.7 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.

46.6.8 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.

46.6.9 Flow Diagrams

A flow diagram of WDOG operation is shown below.

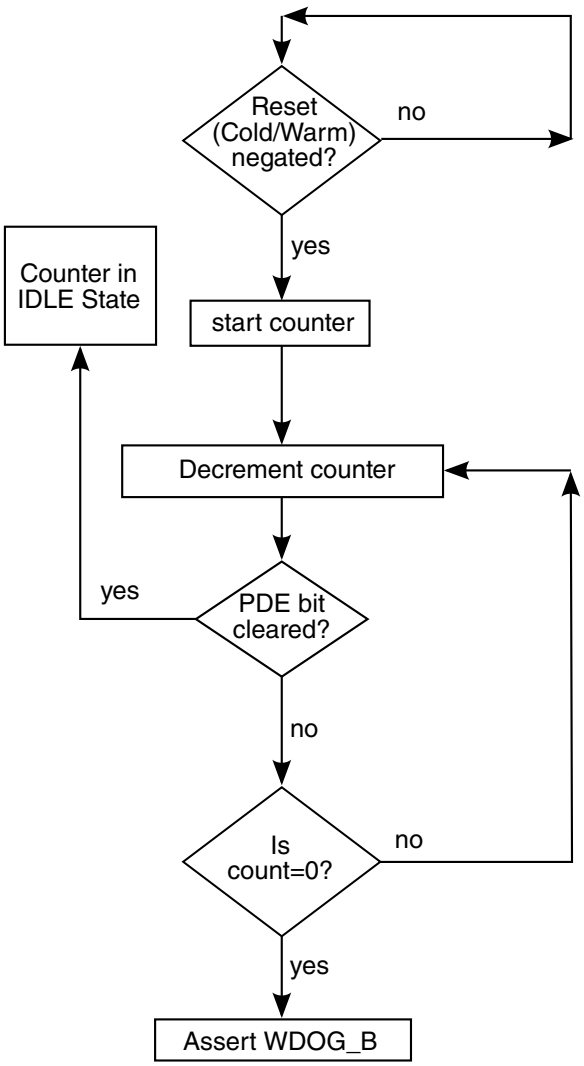


Figure 46-5. Power-Down Counter Flow Diagram

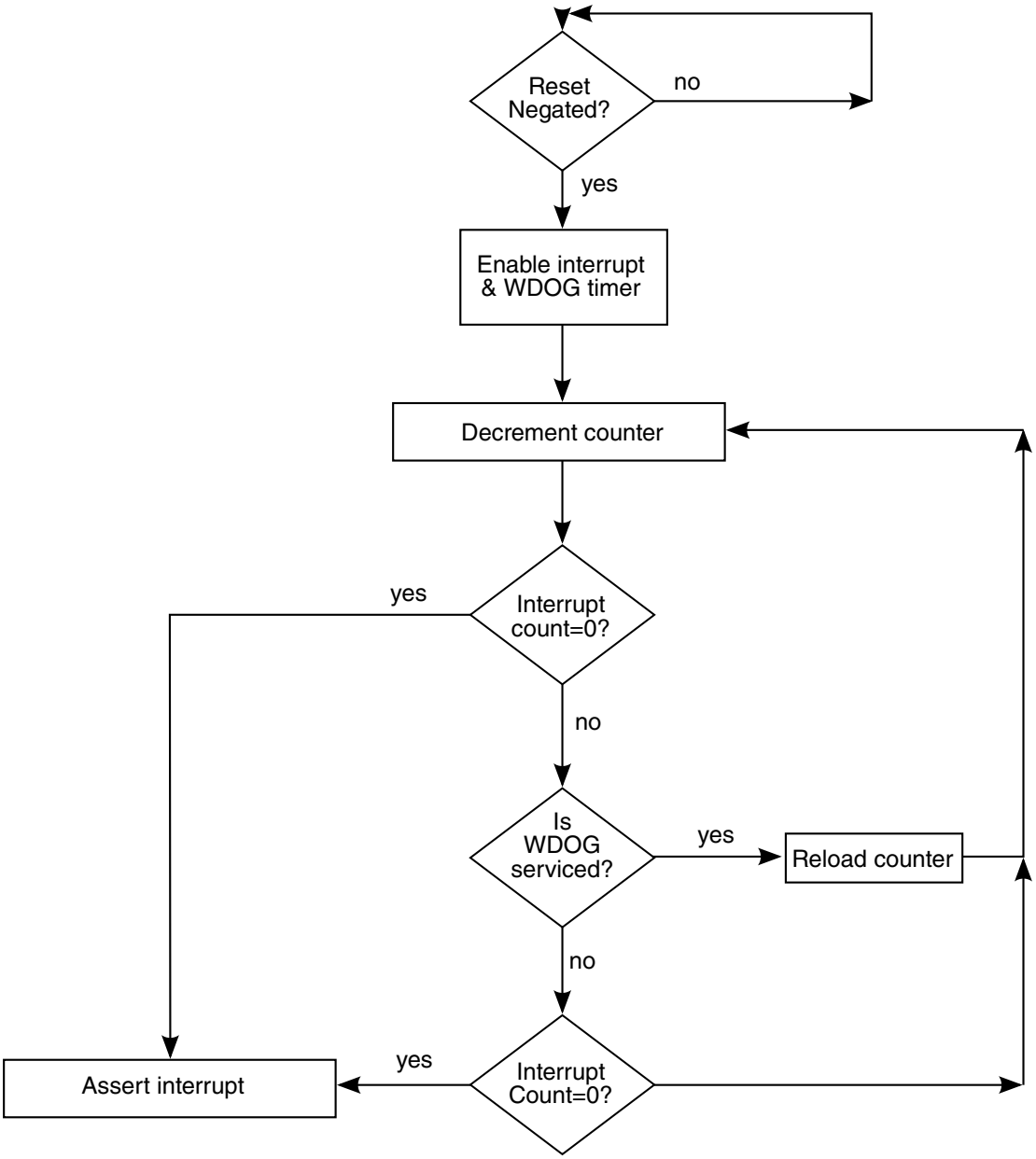


Figure 46-6. Interrupt Generation Flow Diagram

46.7 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.

46.8 WDOG Memory Map/Register Definition

46.8.1 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.

46.8.1.1 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

46.8.1.2 Watchdog Control (WCR)

46.8.1.2.1 Address

Register	Offset
WCR	0h

46.8.1.2.2 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.

46.8.1.2.3 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

46.8.1.2.4 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.

46.8.1.3 Watchdog Service (WSR)

46.8.1.3.1 Address

Register	Offset
WSR	2h

46.8.1.3.2 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.

46.8.1.3.3 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

46.8.1.3.4 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).

46.8.1.4 Watchdog Reset Status (WRSR)

46.8.1.4.1 Address

Register	Offset
WRSR	4h

46.8.1.4.2 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

46.8.1.4.3 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

46.8.1.4.4 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.

46.8.1.5 Watchdog Interrupt Control (WICR)

46.8.1.5.1 Address

Register	Offset
WICR	6h

46.8.1.5.2 Function

The WDOG_WICR controls the WDOG interrupt generation.

46.8.1.5.3 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

46.8.1.5.4 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.

46.8.1.6 Watchdog Miscellaneous Control (WMCR)

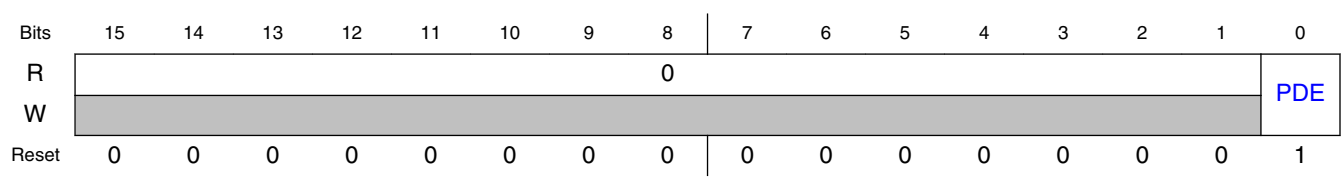
46.8.1.6.1 Address

Register	Offset
WMCR	8h

46.8.1.6.2 Function

WDOG_WMCR Controls the Power Down counter operation.

46.8.1.6.3 Diagram



46.8.1.6.4 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 47
RTWDOG (WDOG3)

47.1 Chip-specific RTWDOG 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

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.

47.2 Introduction

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.

47.2.1 Features

Features of the WDOG module include:

- Configurable clock source inputs independent from the bus clock
  - Bus clock (slow clock)
  - LPO clock
  - INTCLK (internal clock)
  - ERCLK (external reference 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 255 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 255 bus clocks after unlocking and before WDOG closing unlock window.

47.2.2 Block diagram

The following figure shows a block diagram of the WDOG module.

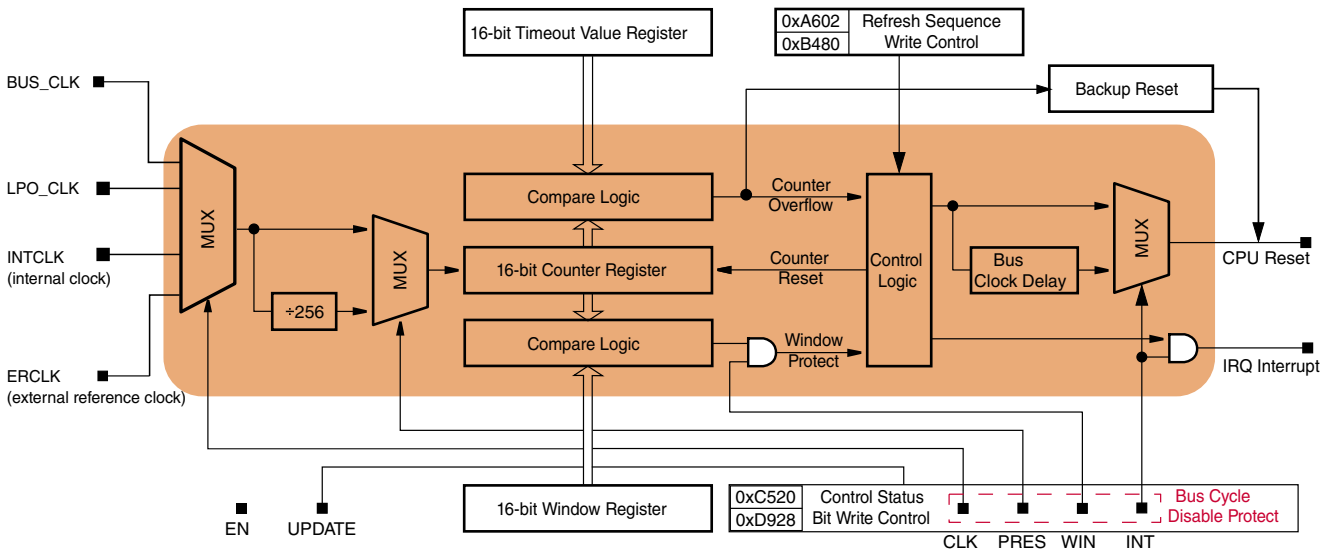


Figure 47-1. WDOG block diagram

47.3 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.

The timeout period, window mode, and clock source are all programmable but must be configured within 255 bus clocks after a reset.

47.3.1 Clock source

The watchdog counter has the following clock source options selected by programming CS[CLK]:

- bus clock
- Low-Power Oscillator clock (LPO_CLK) (This is the *default* source.)

- internal clock
- external clock

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.

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 47-2. Watchdog timeout availability (example)

Reference clock	Prescaler	Watchdog time-out availability
LPO_CLK	Pass through	~1 ms–65.5 s (if LPO_CLK = 1 kHz); (~1 ms–65.5 s)/128 (if LPO_CLK = 128 kHz). [1]
	÷256	~256 ms–16,777.2 s (if LPO_CLK = 1 kHz); ~2 ms–131.1 s (if LPO_CLK = 128 kHz).
Internal clock 8 MHz	Pass through	125 ns–8.1925 ms
	÷256	32 µs–2.09728 s
1 MHz (from bus or external)	Pass through	1 µs–65.54 ms
	÷256	256 µs–16.777 s
20 MHz (from bus or external)	Pass through	50 ns–3.277 ms
	÷256	12.8 µs–838.8 ms

1. The default timeout value after reset is approximately 32000 ms (if LPO_CLK = 1 kHz), or 32000/128 ms (if LPO_CLK = 128 kHz).

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 (255 bus clocks) ends. This delay ensures a smooth transition before restarting the counter with the new configuration.

47.3.2 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.

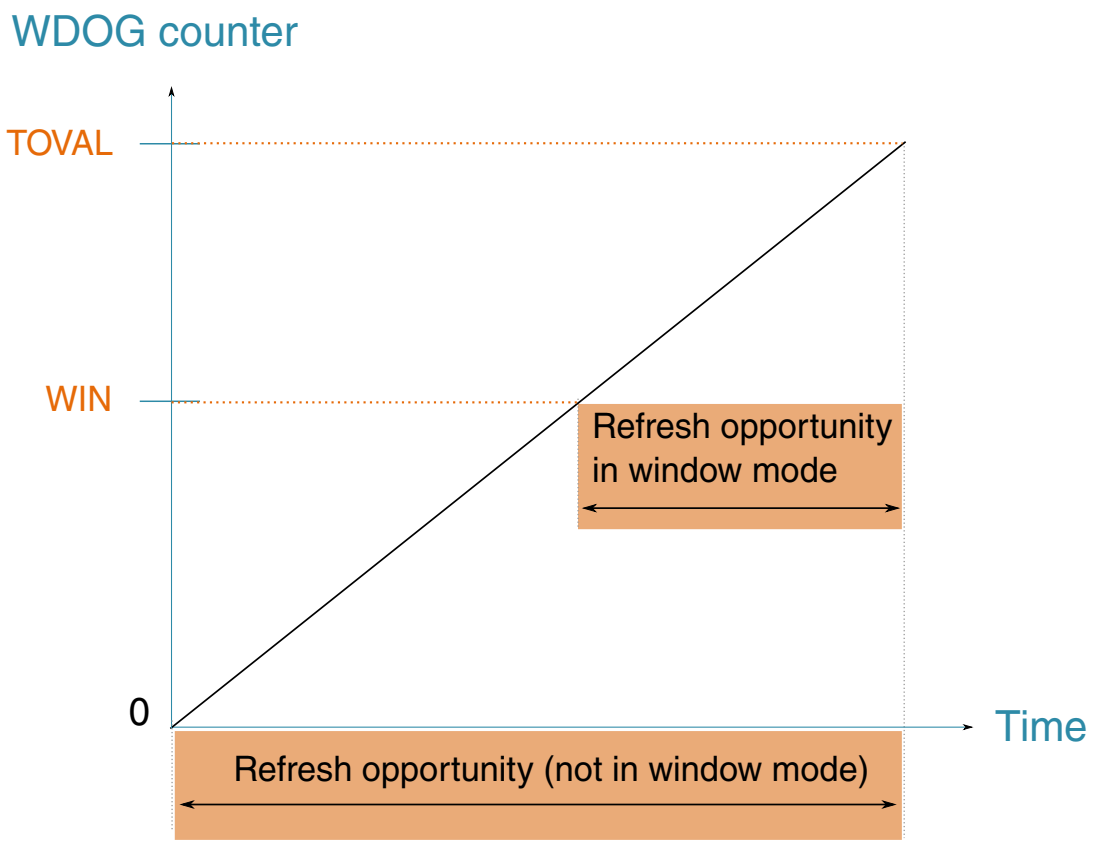


Figure 47-2. Refresh opportunity for the Watchdog counter

47.3.2.1 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.

47.3.2.2 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; applicable to Cortex-M core cases) 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.

47.3.3 Configuring the Watchdog

47.3.3.1 Configuring the Watchdog Once

All watchdog control bits, timeout value, and window value are write-once after reset *within 255 bus clocks*. 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.

47.3.3.2 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.

47.3.3.2.1 Unlocking the Watchdog

The unlock sequence is a write to the CNT register of 0xC520 followed by 0xD928 within 31 bus clocks at any time after the watchdog has been configured. On completing the unlock sequence, the user must reconfigure the watchdog within 255 bus clocks; otherwise, the watchdog closes the unlock window.

NOTE

Due to the 255 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, watchdog will not be able to wake the MCU from low power mode .

The example codes can be found at end of this chapter.

47.3.4 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 255 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 the forcing a reset.

47.3.5 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.

47.3.6 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.

47.3.7 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 kHz 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.

47.3.7.1 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.

47.3.7.2 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 LPO clock source, software can periodically read the CNT register to ensure the counter is being incremented.

47.4 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 enabled in the beginning of bootloader, and disabled 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.

47.4.1 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
```

47.4.2 Configure Watchdog

The watchdog can be configured once by set 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
```

47.4.3 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
```

47.5 Memory map and register definition

47.5.1 WDOG Register Descriptions

47.5.1.1 WDOG Memory Map

WDOG3 (RTWDOG) base address: 400B_C000h

Offset	Register	Width (In bits)	Access	Reset value
0h	Watchdog Control and Status (CS)	32	RW	00002180h
4h	Watchdog Counter (CNT)	32	RW	00000000h
8h	Watchdog Timeout Value (TOVAL)	32	RW	00007D00h
Ch	Watchdog Window (WIN)	32	RW	00000000h

47.5.1.2 Watchdog Control and Status (CS)

47.5.1.2.1 Address

Register	Offset
CS	0h

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.

47.5.1.2.2 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	WIN	FLG	CMD 32EN	PRES	ULK	RCS	CLK		EN	INT	UPDATE	TST		DBG	WAIT	STOP
W		w1c														
Reset	0	0	1	0	0	0	0	1	1	0	0	0	0	0	0	0

47.5.1.2.3 Fields

Field	Function
31-16 —	Reserved

Table continues on the next page...

Field	Function
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 Clock source section. 00b - Bus clock 01b - LPO clock 10b - INTCLK (internal clock) 11b - ERCLK (external reference clock)
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 255 bus clocks. 0b - Watchdog interrupts are disabled. Watchdog resets are not delayed. 1b - Watchdog interrupts are enabled. Watchdog resets are delayed by 255 bus clocks from the interrupt vector fetch.
5	Allow updates

Table continues on the next page...

Field	Function
UPDATE	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 255 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.

47.5.1.3 Watchdog Counter (CNT)

47.5.1.3.1 Address

Register	Offset
CNT	4h

47.5.1.3.2 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.

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	CNTHIGH								CNTLOW							
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

47.5.1.3.4 Fields

Field	Function
31-16 —	Reserved
15-8 CNTHIGH	High byte of the Watchdog Counter
7-0 CNTLOW	Low byte of the Watchdog Counter

47.5.1.4 Watchdog Timeout Value (TOVAL)

47.5.1.4.1 Address

Register	Offset
TOVAL	8h

47.5.1.4.2 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 Watchdog Timeout Value Register; otherwise, the watchdog always generates a reset.

47.5.1.4.3 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	TOVALHIGH								TOVALLOW							
W																
Reset	0	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0

47.5.1.4.4 Fields

Field	Function
31-16 —	Reserved
15-8 TOVALHIGH	High byte of the timeout value
7-0 TOVALLOW	Low byte of the timeout value

47.5.1.5 Watchdog Window (WIN)

47.5.1.5.1 Address

Register	Offset
WIN	Ch

47.5.1.5.2 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.

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															
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

47.5.1.5.4 Fields

Field	Function
31-16 —	Reserved
15-8 WINHIGH	High byte of Watchdog Window
7-0 WINLOW	Low byte of Watchdog Window

Chapter 48
External Watchdog Monitor (EWM)

48.1 Chip-specific EWM information

Table 48-1. Reference links to related information

Topic	Related module(s)	Reference
System memory map	-	System Memory Map
Clocking	CCM	Clock Management
	Low Power Clock Source	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

For low power clock source, see lpo_clk[3:0] in the EWM Signal Descriptions table.

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.

48.2 Introduction

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 assert 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.

48.2.1 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.

48.2.2 Modes of Operation

This section describes the module's operating modes.

48.2.2.1 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.

48.2.2.2 Wait Mode

The EWM module treats the stop and wait modes as the same. EWM functionality remains the same in both of these modes.

48.2.2.3 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.

48.2.3 Block Diagram

This figure shows the EWM block diagram.

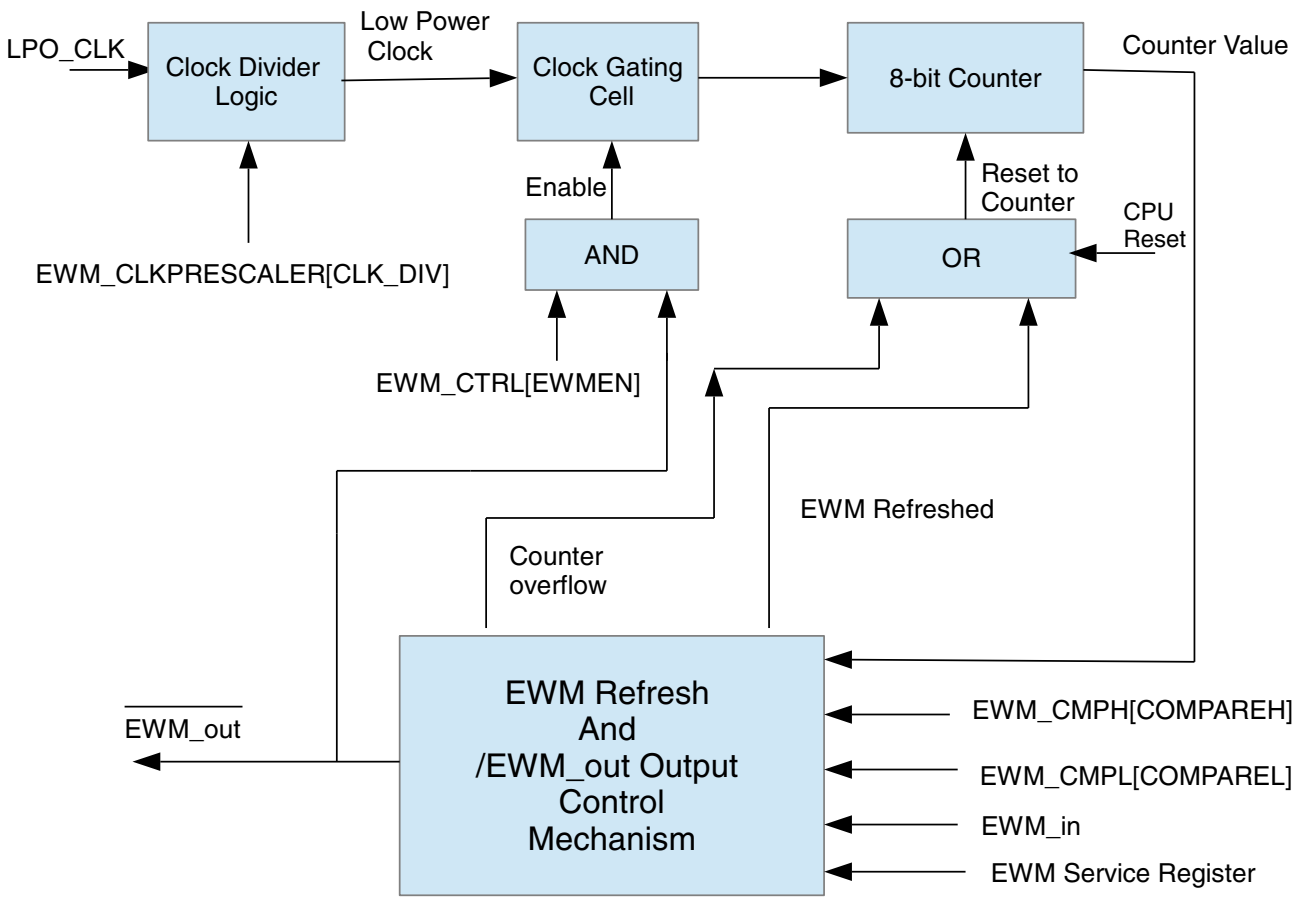


Figure 48-1. EWM Block Diagram

48.3 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 48-2. 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

48.4 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.

48.4.1 EWM register descriptions

48.4.1.1 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

48.4.1.2 Control Register (CTRL)

48.4.1.2.1 Offset

Register	Offset
CTRL	0h

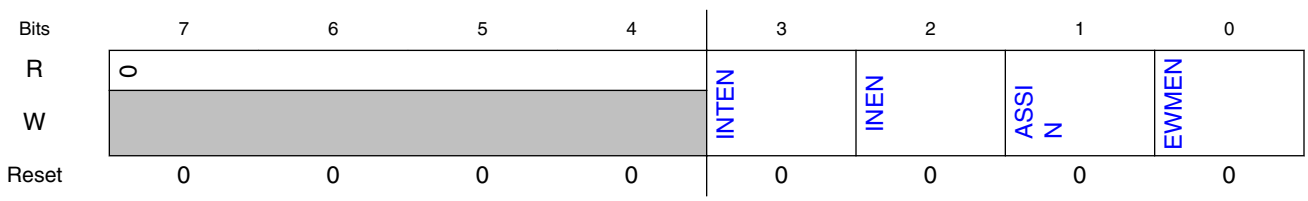
48.4.1.2.2 Function

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.

48.4.1.2.3 Diagram



48.4.1.2.4 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.
2 INEN	Input Enable. This bit when set, enables the EWM_in port.
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.
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.

48.4.1.3 Service Register (SERV)

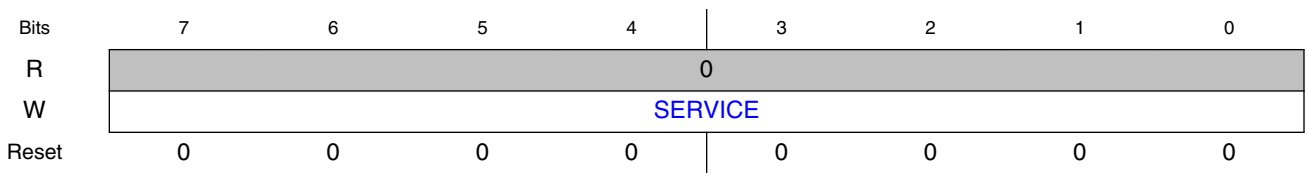
48.4.1.3.1 Offset

Register	Offset
SERV	1h

48.4.1.3.2 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.

48.4.1.3.3 Diagram



48.4.1.3.4 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*

48.4.1.4 Compare Low Register (CMPL)

48.4.1.4.1 Offset

Register	Offset
CMPL	2h

48.4.1.4.2 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.

48.4.1.4.3 Diagram

Bits	7	6	5	4	3	2	1	0
R W	COMPAREL							
Reset	0	0	0	0	0	0	0	0

48.4.1.4.4 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.

48.4.1.5 Compare High Register (CMPH)

48.4.1.5.1 Offset

Register	Offset
CMPH	3h

48.4.1.5.2 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.

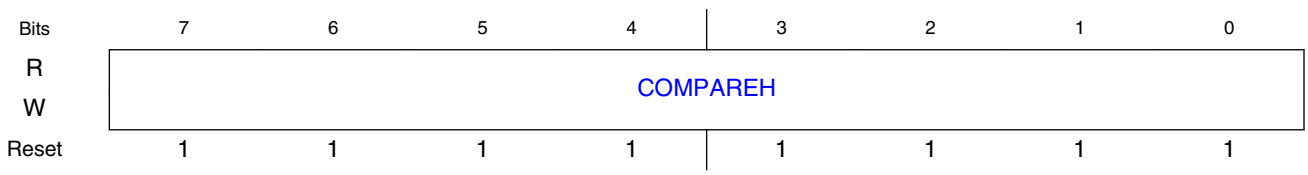
NOTE

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.

48.4.1.5.3 Diagram



48.4.1.5.4 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.

48.4.1.6 Clock Control Register (CLKCTRL)

48.4.1.6.1 Offset

Register	Offset
CLKCTRL	4h

48.4.1.6.2 Function

This CLKCTRL register is reset to 0x00 after a CPU reset.

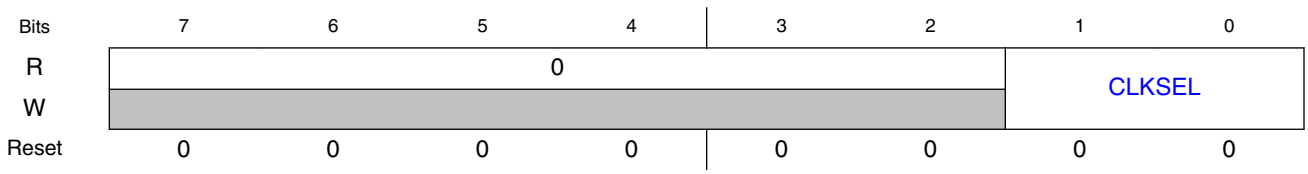
NOTE

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.

48.4.1.6.3 Diagram



48.4.1.6.4 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.

48.4.1.7 Clock Prescaler Register (CLKPRESCALER)

48.4.1.7.1 Offset

Register	Offset
CLKPRESCALER	5h

48.4.1.7.2 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.

48.4.1.7.3 Diagram

Bits	7	6	5	4	3	2	1	0
R W	CLK_DIV							
Reset	0	0	0	0	0	0	0	0

48.4.1.7.4 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)

48.5 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

48.5.1 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.

48.5.2 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.

48.5.3 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.

48.5.4 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.

48.5.5 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.

48.5.6 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 48-3. 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.

48.5.7 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.

48.5.8 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].

48.5.9 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.

Chapter 49
On Chip Cross Triggers Overview

49.1 Overview

This chip integrates an on-chip cross trigger network. The following diagram shows the cross trigger network of this device.

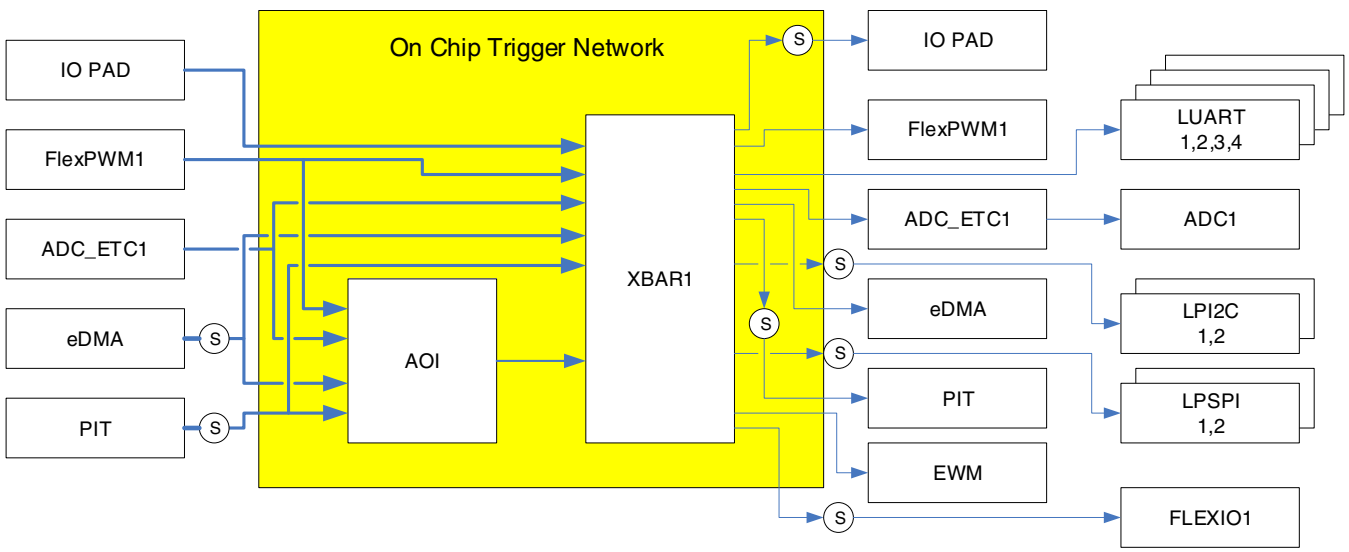


Figure 49-1. On-Chip Cross Trigger Network

49.1.1 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.

49.1.2 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 50
Inter-Peripheral Crossbar Switch A (XBARA)

50.1 Chip-specific XBAR information

On this device, for XBAR1 (see the XBARA chapter), the number of inputs is 28, and the number of outputs is 31.

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.

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

50.2 Introduction

50.2.1 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.

50.2.2 Features

The XBAR module design includes these distinctive features:

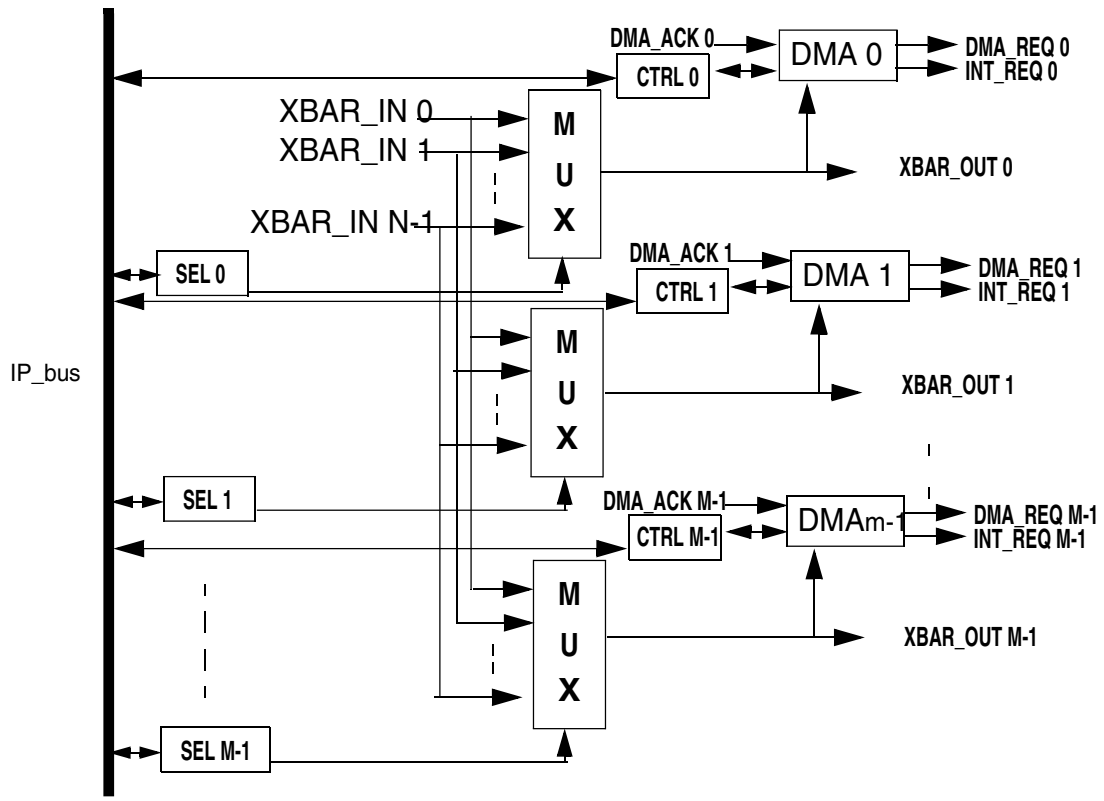
- M identical N-input muxes with individual select fields.
- Edge detection with associated interrupt or DMA request generation for a subset of mux outputs.
- Memory mapped registers with IPBus interface for select and control fields.
- Register write protection input signal.

50.2.3 Modes of Operation

The XBAR module design operates in only a single mode of operation: Functional Mode. The various counting modes are detailed in Functional Mode.

50.2.4 Block Diagram

The block diagram for XBAR is shown in Figure 50-1.



Note: Control field support can be restricted to a subset of outputs. CTRLn represents the memory mapped control fields in the XBAR_CTRL registers for each corresponding XBAR_OUT output. Control fields provide edge status, edge selection, and interrupt/DMA generation controls.

Figure 50-1. XBAR Block Diagram

50.3 Signal Descriptions

The following table summarizes the module's external signals.

Table 50-2. 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: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].

50.3.1 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].

50.3.2 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.

50.3.3 DMA_REQ[n] - DMA Request Output(s)

DMA_REQ[n] is a DMA request to the DMA controller.

50.3.4 DMA_ACK[n] - DMA Acknowledge Input(s)

DMA_ACK[n] is a DMA acknowledge input from the DMA controller.

50.3.5 INT_REQ[n] - Interrupt Request Output(s)

INT_REQ[n] is an interrupt request output to the interrupt controller.

50.4 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 Chip Configuration 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
4009_8000	Crossbar A Select Register 0 (XBARA_SEL0)	16	R/W	0000h	50.4.1/1868
4009_8002	Crossbar A Select Register 1 (XBARA_SEL1)	16	R/W	0000h	50.4.2/1868
4009_8004	Crossbar A Select Register 2 (XBARA_SEL2)	16	R/W	0000h	50.4.3/1869
4009_8006	Crossbar A Select Register 3 (XBARA_SEL3)	16	R/W	0000h	50.4.4/1869
4009_8008	Crossbar A Select Register 4 (XBARA_SEL4)	16	R/W	0000h	50.4.5/1870
4009_800A	Crossbar A Select Register 5 (XBARA_SEL5)	16	R/W	0000h	50.4.6/1870
4009_800C	Crossbar A Select Register 6 (XBARA_SEL6)	16	R/W	0000h	50.4.7/1871
4009_800E	Crossbar A Select Register 7 (XBARA_SEL7)	16	R/W	0000h	50.4.8/1871
4009_8010	Crossbar A Select Register 8 (XBARA_SEL8)	16	R/W	0000h	50.4.9/1872
4009_8012	Crossbar A Select Register 9 (XBARA_SEL9)	16	R/W	0000h	50.4.10/ 1872
4009_8014	Crossbar A Select Register 10 (XBARA_SEL10)	16	R/W	0000h	50.4.11/ 1873
4009_8016	Crossbar A Select Register 11 (XBARA_SEL11)	16	R/W	0000h	50.4.12/ 1873
4009_8018	Crossbar A Select Register 12 (XBARA_SEL12)	16	R/W	0000h	50.4.13/ 1874
4009_801A	Crossbar A Select Register 13 (XBARA_SEL13)	16	R/W	0000h	50.4.14/ 1874
4009_801C	Crossbar A Select Register 14 (XBARA_SEL14)	16	R/W	0000h	50.4.15/ 1875
4009_801E	Crossbar A Select Register 15 (XBARA_SEL15)	16	R/W	0000h	50.4.16/ 1875
4009_8020	Crossbar A Select Register 16 (XBARA_SEL16)	16	R/W	0000h	50.4.17/ 1876
4009_8022	Crossbar A Select Register 17 (XBARA_SEL17)	16	R/W	0000h	50.4.18/ 1876
4009_8024	Crossbar A Select Register 18 (XBARA_SEL18)	16	R/W	0000h	50.4.19/ 1877
4009_8026	Crossbar A Select Register 19 (XBARA_SEL19)	16	R/W	0000h	50.4.20/ 1877

Table continues on the next page...

XBARA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4009_8028	Crossbar A Select Register 20 (XBARA_SEL20)	16	R/W	0000h	50.4.21/ 1878
4009_802A	Crossbar A Select Register 21 (XBARA_SEL21)	16	R/W	0000h	50.4.22/ 1878
4009_802C	Crossbar A Select Register 22 (XBARA_SEL22)	16	R/W	0000h	50.4.23/ 1879
4009_802E	Crossbar A Select Register 23 (XBARA_SEL23)	16	R/W	0000h	50.4.24/ 1879
4009_8030	Crossbar A Select Register 24 (XBARA_SEL24)	16	R/W	0000h	50.4.25/ 1880
4009_8032	Crossbar A Select Register 25 (XBARA_SEL25)	16	R/W	0000h	50.4.26/ 1880
4009_8034	Crossbar A Select Register 26 (XBARA_SEL26)	16	R/W	0000h	50.4.27/ 1881
4009_8036	Crossbar A Select Register 27 (XBARA_SEL27)	16	R/W	0000h	50.4.28/ 1881
4009_8038	Crossbar A Select Register 28 (XBARA_SEL28)	16	R/W	0000h	50.4.29/ 1882
4009_803A	Crossbar A Select Register 29 (XBARA_SEL29)	16	R/W	0000h	50.4.30/ 1882
4009_803C	Crossbar A Select Register 30 (XBARA_SEL30)	16	R/W	0000h	50.4.31/ 1883
4009_803E	Crossbar A Select Register 31 (XBARA_SEL31)	16	R/W	0000h	50.4.32/ 1883
4009_8040	Crossbar A Select Register 32 (XBARA_SEL32)	16	R/W	0000h	50.4.33/ 1884
4009_8042	Crossbar A Select Register 33 (XBARA_SEL33)	16	R/W	0000h	50.4.34/ 1884
4009_8044	Crossbar A Select Register 34 (XBARA_SEL34)	16	R/W	0000h	50.4.35/ 1885
4009_8046	Crossbar A Select Register 35 (XBARA_SEL35)	16	R/W	0000h	50.4.36/ 1885
4009_8048	Crossbar A Select Register 36 (XBARA_SEL36)	16	R/W	0000h	50.4.37/ 1886
4009_804A	Crossbar A Select Register 37 (XBARA_SEL37)	16	R/W	0000h	50.4.38/ 1886
4009_804C	Crossbar A Select Register 38 (XBARA_SEL38)	16	R/W	0000h	50.4.39/ 1887
4009_804E	Crossbar A Select Register 39 (XBARA_SEL39)	16	R/W	0000h	50.4.40/ 1887
4009_8050	Crossbar A Select Register 40 (XBARA_SEL40)	16	R/W	0000h	50.4.41/ 1888
4009_8052	Crossbar A Select Register 41 (XBARA_SEL41)	16	R/W	0000h	50.4.42/ 1888

Table continues on the next page...

XBARA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4009_8054	Crossbar A Select Register 42 (XBARA_SEL42)	16	R/W	0000h	50.4.43/ 1889
4009_8056	Crossbar A Select Register 43 (XBARA_SEL43)	16	R/W	0000h	50.4.44/ 1889
4009_8058	Crossbar A Select Register 44 (XBARA_SEL44)	16	R/W	0000h	50.4.45/ 1890
4009_805A	Crossbar A Select Register 45 (XBARA_SEL45)	16	R/W	0000h	50.4.46/ 1890
4009_805C	Crossbar A Select Register 46 (XBARA_SEL46)	16	R/W	0000h	50.4.47/ 1891
4009_805E	Crossbar A Select Register 47 (XBARA_SEL47)	16	R/W	0000h	50.4.48/ 1891
4009_8060	Crossbar A Select Register 48 (XBARA_SEL48)	16	R/W	0000h	50.4.49/ 1892
4009_8062	Crossbar A Select Register 49 (XBARA_SEL49)	16	R/W	0000h	50.4.50/ 1892
4009_8064	Crossbar A Select Register 50 (XBARA_SEL50)	16	R/W	0000h	50.4.51/ 1893
4009_8066	Crossbar A Select Register 51 (XBARA_SEL51)	16	R/W	0000h	50.4.52/ 1893
4009_8068	Crossbar A Select Register 52 (XBARA_SEL52)	16	R/W	0000h	50.4.53/ 1894
4009_806A	Crossbar A Select Register 53 (XBARA_SEL53)	16	R/W	0000h	50.4.54/ 1894
4009_806C	Crossbar A Select Register 54 (XBARA_SEL54)	16	R/W	0000h	50.4.55/ 1895
4009_806E	Crossbar A Select Register 55 (XBARA_SEL55)	16	R/W	0000h	50.4.56/ 1895
4009_8070	Crossbar A Select Register 56 (XBARA_SEL56)	16	R/W	0000h	50.4.57/ 1896
4009_8072	Crossbar A Select Register 57 (XBARA_SEL57)	16	R/W	0000h	50.4.58/ 1896
4009_8074	Crossbar A Select Register 58 (XBARA_SEL58)	16	R/W	0000h	50.4.59/ 1897
4009_8076	Crossbar A Select Register 59 (XBARA_SEL59)	16	R/W	0000h	50.4.60/ 1897
4009_8078	Crossbar A Select Register 60 (XBARA_SEL60)	16	R/W	0000h	50.4.61/ 1898
4009_807A	Crossbar A Select Register 61 (XBARA_SEL61)	16	R/W	0000h	50.4.62/ 1898
4009_807C	Crossbar A Select Register 62 (XBARA_SEL62)	16	R/W	0000h	50.4.63/ 1899
4009_807E	Crossbar A Select Register 63 (XBARA_SEL63)	16	R/W	0000h	50.4.64/ 1899

Table continues on the next page...

XBARA memory map (continued)

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4009_8080	Crossbar A Select Register 64 (XBARA_SEL64)	16	R/W	0000h	50.4.65/ 1900
4009_8082	Crossbar A Select Register 65 (XBARA_SEL65)	16	R/W	0000h	50.4.66/ 1900
4009_8084	Crossbar A Control Register 0 (XBARA_CTRL0)	16	R/W	0000h	50.4.67/ 1901
4009_8086	Crossbar A Control Register 1 (XBARA_CTRL1)	16	R/W	0000h	50.4.68/ 1903

50.4.1 Crossbar A Select Register 0 (XBARA_SEL0)

Address: 4009_8000h base + 0h offset = 4009_8000h

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

XBARA_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)

50.4.2 Crossbar A Select Register 1 (XBARA_SEL1)

Address: 4009_8000h base + 2h offset = 4009_8002h

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_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)

50.4.3 Crossbar A Select Register 2 (XBARA_SEL2)

Address: 4009_8000h base + 4h offset = 4009_8004h

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_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)

50.4.4 Crossbar A Select Register 3 (XBARA_SEL3)

Address: 4009_8000h base + 6h offset = 4009_8006h

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_SEL3 field descriptions

Field	Description
15 Reserved	This field is reserved. This read-only field is reserved and always has the value 0.

Table continues on the next page...

XBARA_SEL3 field descriptions (continued)

Field	Description
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)

50.4.5 Crossbar A Select Register 4 (XBARA_SEL4)

Address: 4009_8000h base + 8h offset = 4009_8008h

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_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)

50.4.6 Crossbar A Select Register 5 (XBARA_SEL5)

Address: 4009_8000h base + Ah offset = 4009_800Ah

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_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)

Table continues on the next page...

XBARA_SEL5 field descriptions (continued)

Field	Description
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)

50.4.7 Crossbar A Select Register 6 (XBARA_SEL6)

Address: 4009_8000h base + Ch offset = 4009_800Ch

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_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)

50.4.8 Crossbar A Select Register 7 (XBARA_SEL7)

Address: 4009_8000h base + Eh offset = 4009_800Eh

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_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.

Table continues on the next page...

XBARA_SEL7 field descriptions (continued)

Field	Description
SEL14	Input (XBARA_INn) to be muxed to XBARA_OUT14 (refer to Functional Description section for input/ output assignment)

50.4.9 Crossbar A Select Register 8 (XBARA_SEL8)

Address: 4009_8000h base + 10h offset = 4009_8010h

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_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)

50.4.10 Crossbar A Select Register 9 (XBARA_SEL9)

Address: 4009_8000h base + 12h offset = 4009_8012h

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

XBARA_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)

50.4.11 Crossbar A Select Register 10 (XBARA_SEL10)

Address: 4009_8000h base + 14h offset = 4009_8014h

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

XBARA_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)

50.4.12 Crossbar A Select Register 11 (XBARA_SEL11)

Address: 4009_8000h base + 16h offset = 4009_8016h

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

XBARA_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)

50.4.13 Crossbar A Select Register 12 (XBARA_SEL12)

Address: 4009_8000h base + 18h offset = 4009_8018h

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

XBARA_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)

50.4.14 Crossbar A Select Register 13 (XBARA_SEL13)

Address: 4009_8000h base + 1Ah offset = 4009_801Ah

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

XBARA_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)

50.4.15 Crossbar A Select Register 14 (XBARA_SEL14)

Address: 4009_8000h base + 1Ch offset = 4009_801Ch

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

XBARA_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)

50.4.16 Crossbar A Select Register 15 (XBARA_SEL15)

Address: 4009_8000h base + 1Eh offset = 4009_801Eh

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

XBARA_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)

50.4.17 Crossbar A Select Register 16 (XBARA_SEL16)

Address: 4009_8000h base + 20h offset = 4009_8020h

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

XBARA_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)

50.4.18 Crossbar A Select Register 17 (XBARA_SEL17)

Address: 4009_8000h base + 22h offset = 4009_8022h

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

XBARA_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)

50.4.19 Crossbar A Select Register 18 (XBARA_SEL18)

Address: 4009_8000h base + 24h offset = 4009_8024h

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

XBARA_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)

50.4.20 Crossbar A Select Register 19 (XBARA_SEL19)

Address: 4009_8000h base + 26h offset = 4009_8026h

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

XBARA_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)

50.4.21 Crossbar A Select Register 20 (XBARA_SEL20)

Address: 4009_8000h base + 28h offset = 4009_8028h

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

XBARA_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)

50.4.22 Crossbar A Select Register 21 (XBARA_SEL21)

Address: 4009_8000h base + 2Ah offset = 4009_802Ah

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

XBARA_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)

50.4.23 Crossbar A Select Register 22 (XBARA_SEL22)

Address: 4009_8000h base + 2Ch offset = 4009_802Ch

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

XBARA_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)

50.4.24 Crossbar A Select Register 23 (XBARA_SEL23)

Address: 4009_8000h base + 2Eh offset = 4009_802Eh

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

XBARA_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)

50.4.25 Crossbar A Select Register 24 (XBARA_SEL24)

Address: 4009_8000h base + 30h offset = 4009_8030h

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

XBARA_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)

50.4.26 Crossbar A Select Register 25 (XBARA_SEL25)

Address: 4009_8000h base + 32h offset = 4009_8032h

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

XBARA_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)

50.4.27 Crossbar A Select Register 26 (XBARA_SEL26)

Address: 4009_8000h base + 34h offset = 4009_8034h

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

XBARA_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)

50.4.28 Crossbar A Select Register 27 (XBARA_SEL27)

Address: 4009_8000h base + 36h offset = 4009_8036h

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

XBARA_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)

50.4.29 Crossbar A Select Register 28 (XBARA_SEL28)

Address: 4009_8000h base + 38h offset = 4009_8038h

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

XBARA_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)

50.4.30 Crossbar A Select Register 29 (XBARA_SEL29)

Address: 4009_8000h base + 3Ah offset = 4009_803Ah

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

XBARA_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)

50.4.31 Crossbar A Select Register 30 (XBARA_SEL30)

Address: 4009_8000h base + 3Ch offset = 4009_803Ch

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

XBARA_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)

50.4.32 Crossbar A Select Register 31 (XBARA_SEL31)

Address: 4009_8000h base + 3Eh offset = 4009_803Eh

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

XBARA_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)

50.4.33 Crossbar A Select Register 32 (XBARA_SEL32)

Address: 4009_8000h base + 40h offset = 4009_8040h

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

XBARA_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)

50.4.34 Crossbar A Select Register 33 (XBARA_SEL33)

Address: 4009_8000h base + 42h offset = 4009_8042h

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

XBARA_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)

50.4.35 Crossbar A Select Register 34 (XBARA_SEL34)

Address: 4009_8000h base + 44h offset = 4009_8044h

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

XBARA_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)

50.4.36 Crossbar A Select Register 35 (XBARA_SEL35)

Address: 4009_8000h base + 46h offset = 4009_8046h

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

XBARA_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)

50.4.37 Crossbar A Select Register 36 (XBARA_SEL36)

Address: 4009_8000h base + 48h offset = 4009_8048h

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

XBARA_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)

50.4.38 Crossbar A Select Register 37 (XBARA_SEL37)

Address: 4009_8000h base + 4Ah offset = 4009_804Ah

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

XBARA_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)

50.4.39 Crossbar A Select Register 38 (XBARA_SEL38)

Address: 4009_8000h base + 4Ch offset = 4009_804Ch

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

XBARA_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)

50.4.40 Crossbar A Select Register 39 (XBARA_SEL39)

Address: 4009_8000h base + 4Eh offset = 4009_804Eh

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

XBARA_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)

50.4.41 Crossbar A Select Register 40 (XBARA_SEL40)

Address: 4009_8000h base + 50h offset = 4009_8050h

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

XBARA_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)

50.4.42 Crossbar A Select Register 41 (XBARA_SEL41)

Address: 4009_8000h base + 52h offset = 4009_8052h

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

XBARA_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)

50.4.43 Crossbar A Select Register 42 (XBARA_SEL42)

Address: 4009_8000h base + 54h offset = 4009_8054h

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

XBARA_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)

50.4.44 Crossbar A Select Register 43 (XBARA_SEL43)

Address: 4009_8000h base + 56h offset = 4009_8056h

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

XBARA_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)

50.4.45 Crossbar A Select Register 44 (XBARA_SEL44)

Address: 4009_8000h base + 58h offset = 4009_8058h

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

XBARA_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)

50.4.46 Crossbar A Select Register 45 (XBARA_SEL45)

Address: 4009_8000h base + 5Ah offset = 4009_805Ah

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

XBARA_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)

50.4.47 Crossbar A Select Register 46 (XBARA_SEL46)

Address: 4009_8000h base + 5Ch offset = 4009_805Ch

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

XBARA_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)

50.4.48 Crossbar A Select Register 47 (XBARA_SEL47)

Address: 4009_8000h base + 5Eh offset = 4009_805Eh

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

XBARA_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)

50.4.49 Crossbar A Select Register 48 (XBARA_SEL48)

Address: 4009_8000h base + 60h offset = 4009_8060h

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

XBARA_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)

50.4.50 Crossbar A Select Register 49 (XBARA_SEL49)

Address: 4009_8000h base + 62h offset = 4009_8062h

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

XBARA_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)

50.4.51 Crossbar A Select Register 50 (XBARA_SEL50)

Address: 4009_8000h base + 64h offset = 4009_8064h

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

XBARA_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)

50.4.52 Crossbar A Select Register 51 (XBARA_SEL51)

Address: 4009_8000h base + 66h offset = 4009_8066h

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

XBARA_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)

50.4.53 Crossbar A Select Register 52 (XBARA_SEL52)

Address: 4009_8000h base + 68h offset = 4009_8068h

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

XBARA_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)

50.4.54 Crossbar A Select Register 53 (XBARA_SEL53)

Address: 4009_8000h base + 6Ah offset = 4009_806Ah

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

XBARA_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)

50.4.55 Crossbar A Select Register 54 (XBARA_SEL54)

Address: 4009_8000h base + 6Ch offset = 4009_806Ch

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

XBARA_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)

50.4.56 Crossbar A Select Register 55 (XBARA_SEL55)

Address: 4009_8000h base + 6Eh offset = 4009_806Eh

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

XBARA_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)

50.4.57 Crossbar A Select Register 56 (XBARA_SEL56)

Address: 4009_8000h base + 70h offset = 4009_8070h

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

XBARA_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)

50.4.58 Crossbar A Select Register 57 (XBARA_SEL57)

Address: 4009_8000h base + 72h offset = 4009_8072h

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

XBARA_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)

50.4.59 Crossbar A Select Register 58 (XBARA_SEL58)

Address: 4009_8000h base + 74h offset = 4009_8074h

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

XBARA_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)

50.4.60 Crossbar A Select Register 59 (XBARA_SEL59)

Address: 4009_8000h base + 76h offset = 4009_8076h

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

XBARA_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)

50.4.61 Crossbar A Select Register 60 (XBARA_SEL60)

Address: 4009_8000h base + 78h offset = 4009_8078h

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

XBARA_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)

50.4.62 Crossbar A Select Register 61 (XBARA_SEL61)

Address: 4009_8000h base + 7Ah offset = 4009_807Ah

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

XBARA_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)

50.4.63 Crossbar A Select Register 62 (XBARA_SEL62)

Address: 4009_8000h base + 7Ch offset = 4009_807Ch

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

XBARA_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)

50.4.64 Crossbar A Select Register 63 (XBARA_SEL63)

Address: 4009_8000h base + 7Eh offset = 4009_807Eh

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

XBARA_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)

50.4.65 Crossbar A Select Register 64 (XBARA_SEL64)

Address: 4009_8000h base + 80h offset = 4009_8080h

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

XBARA_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)

50.4.66 Crossbar A Select Register 65 (XBARA_SEL65)

Address: 4009_8000h base + 82h offset = 4009_8082h

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

XBARA_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)

50.4.67 Crossbar A Control Register 0 (XBARA_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: 4009_8000h base + 84h offset = 4009_8084h

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

XBARA_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

Table continues on the next page...

XBARA_CTRL0 field descriptions (continued)

Field	Description
	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. 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.

Table continues on the next page...

XBARA_CTRL0 field descriptions (continued)

Field	Description
	Restriction: IEN0 and DEN0 should not both be set to 1. 0 DMA disabled 1 DMA enabled

50.4.68 Crossbar A Control Register 1 (XBARA_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: 4009_8000h base + 86h offset = 4009_8086h

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

XBARA_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

Table continues on the next page...

XBARA_CTRL1 field descriptions (continued)

Field	Description
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. 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

Table continues on the next page...

XBARA_CTRL1 field descriptions (continued)

Field	Description
	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

50.5 Functional Description

50.5.1 General

The XBAR module has only one mode of operation, functional mode.

50.5.2 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.

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 (DEN and IEN). 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.

50.6 Resets

The XBAR module can be reset by only a hard reset, which forces all registers to their reset state.

50.7 Clocks

All sequential functionality is controlled by the Bus Clock.

50.8 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 EDGEDENn 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.

DENn and IENn should not be set to 1 at the same time for the same output XBAR_OUT[n].

Chapter 51
And-Or-Inverter (AOI)

51.1 Chip-specific AOI information

Table 51-1. 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

51.2 Introduction

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.

51.2.1 Overview

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.

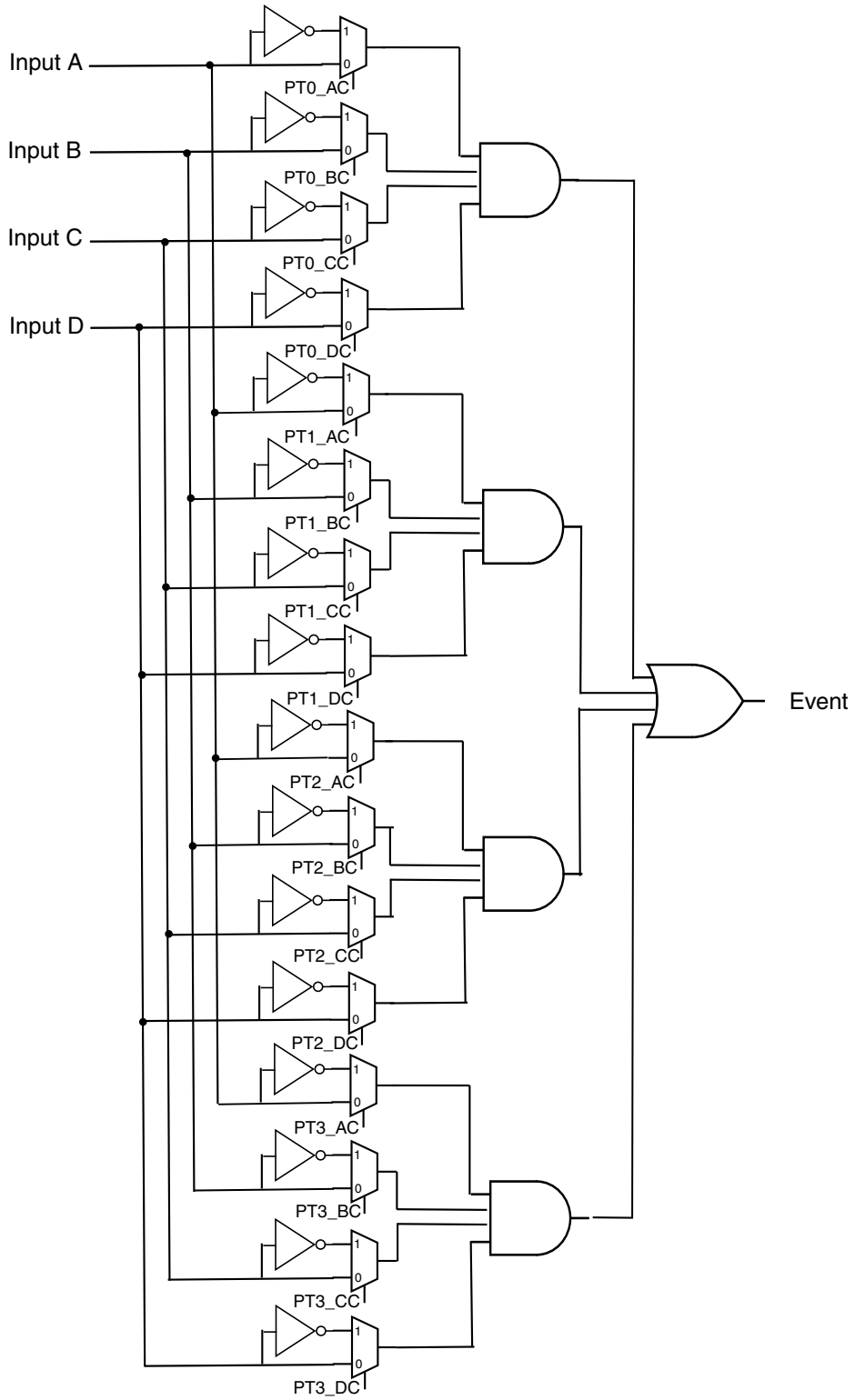


Figure 51-1. Simplified AOI Block Diagram

51.2.2 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

51.2.3 Modes of Operation

The AOI module does not support any special modes of operation. As shown in Figure 51-1, 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*.

51.3 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.

51.4 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.

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 termuses 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.

AOI memory map

Absolute address (hex)	Register name	Width (in bits)	Access	Reset value	Section/ page
4009_4000	Boolean Function Term 0 and 1 Configuration Register for EVENTn (AOI_BFCRT010)	16	R/W	0000h	51.4.1/1912
4009_4002	Boolean Function Term 2 and 3 Configuration Register for EVENTn (AOI_BFCRT230)	16	R/W	0000h	51.4.2/1913
4009_4004	Boolean Function Term 0 and 1 Configuration Register for EVENTn (AOI_BFCRT011)	16	R/W	0000h	51.4.1/1912
4009_4006	Boolean Function Term 2 and 3 Configuration Register for EVENTn (AOI_BFCRT231)	16	R/W	0000h	51.4.2/1913
4009_4008	Boolean Function Term 0 and 1 Configuration Register for EVENTn (AOI_BFCRT012)	16	R/W	0000h	51.4.1/1912
4009_400A	Boolean Function Term 2 and 3 Configuration Register for EVENTn (AOI_BFCRT232)	16	R/W	0000h	51.4.2/1913
4009_400C	Boolean Function Term 0 and 1 Configuration Register for EVENTn (AOI_BFCRT013)	16	R/W	0000h	51.4.1/1912
4009_400E	Boolean Function Term 2 and 3 Configuration Register for EVENTn (AOI_BFCRT233)	16	R/W	0000h	51.4.2/1913

51.4.1 Boolean Function Term 0 and 1 Configuration Register for EVENTn (AOI_BFCRT01*n*)

Address: 4009_4000h base + 0h offset + (4d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	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

AOI_BFCRT01*n* field descriptions

Field	Description
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. 00 Force the A input in this product term to a logical zero 01 Pass the A input in this product term 10 Complement the A input in this product term 11 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. 00 Force the B input in this product term to a logical zero 01 Pass the B input in this product term 10 Complement the B input in this product term 11 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. 00 Force the C input in this product term to a logical zero 01 Pass the C input in this product term 10 Complement the C input in this product term 11 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. 00 Force the D input in this product term to a logical zero 01 Pass the D input in this product term 10 Complement the D input in this product term 11 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. 00 Force the A input in this product term to a logical zero 01 Pass the A input in this product term

Table continues on the next page...

AOI_BFCRT01*n* field descriptions (continued)

Field	Description
	10 Complement the A input in this product term 11 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. 00 Force the B input in this product term to a logical zero 01 Pass the B input in this product term 10 Complement the B input in this product term 11 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. 00 Force the C input in this product term to a logical zero 01 Pass the C input in this product term 10 Complement the C input in this product term 11 Force the C input in this product term to a logical one
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. 00 Force the D input in this product term to a logical zero 01 Pass the D input in this product term 10 Complement the D input in this product term 11 Force the D input in this product term to a logical one

51.4.2 Boolean Function Term 2 and 3 Configuration Register for EVENTn (AOI_BFCRT23*n*)

Address: 4009_4000h base + 2h offset + (4d × i), where i=0d to 3d

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Read Write	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

AOI_BFCRT23*n* field descriptions

Field	Description
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. 00 Force the A input in this product term to a logical zero 01 Pass the A input in this product term 10 Complement the A input in this product term 11 Force the A input in this product term to a logical one

Table continues on the next page...

AOI_BFCRT23*n* field descriptions (continued)

Field	Description
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. 00 Force the B input in this product term to a logical zero 01 Pass the B input in this product term 10 Complement the B input in this product term 11 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. 00 Force the C input in this product term to a logical zero 01 Pass the C input in this product term 10 Complement the C input in this product term 11 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. 00 Force the D input in this product term to a logical zero 01 Pass the D input in this product term 10 Complement the D input in this product term 11 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. 00 Force the A input in this product term to a logical zero 01 Pass the A input in this product term 10 Complement the A input in this product term 11 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. 00 Force the B input in this product term to a logical zero 01 Pass the B input in this product term 10 Complement the B input in this product term 11 Force the B input in this product term to a logical one
3–2 PT3_CC	Product term 3, C input configuration This 2-bit field defines the Boolean evaluation associated with the selected input C in product term 3. 00 Force the C input in this product term to a logical zero 01 Pass the C input in this product term 10 Complement the C input in this product term 11 Force the C input in this product term to a logical one
PT3_DC	Product term 3, D input configuration This 2-bit field defines the Boolean evaluation associated with the selected input D in product term 3.

Table continues on the next page...

AOI_BFCRT23*n* field descriptions (continued)

Field	Description
	00 Force the D input in this product term to a logical zero 01 Pass the D input in this product term 10 Complement the D input in this product term 11 Force the D input in this product term to a logical one

51.5 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 51-1, 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.

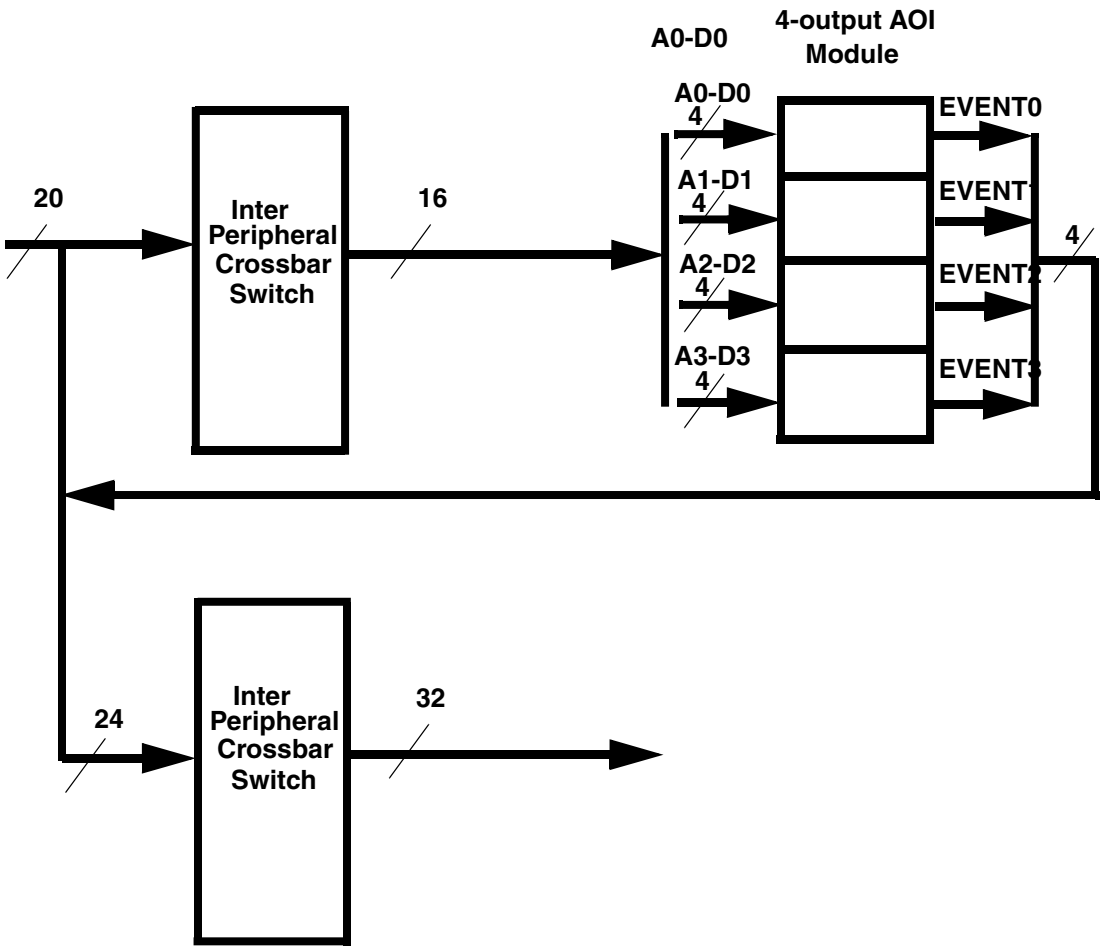


Figure 51-2. Integration Example of AOI with two Inter-Peripheral Crossbar Switches

51.5.1 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] & A |  PT0_AC[1] & ~A)// product term 0
                     & (PT0_BC[0] &  B | PTO_BC[1] & ~B)
                     & (PT0_CC[0] &  C | PTO_CC[1] & ~C)
                     & (PT0_DC[0] &  D | PTO_DC[1] & ~D)

    | (PT1_AC[0] & A |  PT1_AC[1] & ~A)// product term 1
                     & (PT1_BC[0] &  B | PT1_BC[1] & ~B)
                     & (PT1_CC[0] &  C | PT1_CC[1] & ~C)
                     & (PT1_DC[0] &  D | PT1_DC[1] & ~D)

    | (PT2_AC[0] & A |  PT2_AC[1] & ~A)// product term 2
                     & (PT2_BC[0] &  B | PT2_BC[1] & ~B)
                     & (PT2_CC[0] &  C | PT2_CC[1] & ~C)
                     & (PT2_DC[0] &  D | PT2_DC[1] & ~D)

    | (PT3_AC[0] & A |  PT3_AC[1] & ~A)// product term 3
                     & (PT3_BC[0] &  B | PT3_BC[1] & ~B)
                     & (PT3_CC[0] &  C | PT3_CC[1] & ~C)
                     & (PT3_DC[0] &  D | PT3_DC[1] & ~D)
```

where the bits of the combined {BFECRT01n,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 {BFECRT01n,BFCRT23n} registers for several simple boolean expressions as shown in Table 51-2.

Table 51-2. IEVENT_BFECRn 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.

51.5.2 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.

Chapter 52
Analog Overview

52.1 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
- ADC External Trigger Control (ADC_ETC) - Enables multiple users to share a ADC module in a Time-Division-Multiplexing (TDM) method

52.1.1 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

52.1.2 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 is configured with a Back-to-Back (B2B) and Interrupt Enable (IE) settings and each external trigger can be configured with a fixed priority.

Chapter 53
Analog-to-Digital Converter (ADC)

53.1 Chip-specific ADC 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

NOTE

V_{REFH} / V_{REFL} are bonded to VDDA_ADC_3P3 / GND, on this device.

53.2 Overview

The analog-to-digital converter (ADC) is a successive approximation ADC designed for operation within an integrated microcontroller system-on-chip.

53.2.1 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 four sources
- Asynchronous clock source for lower noise operation with option to output the clock
- Selectable asynchronous hardware conversion trigger with hardware channel select
- Selectable voltage reference, internal, external, or alternate
- 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

53.2.2 ADC I/F block diagram

The following diagram represents the ADC I/F block.

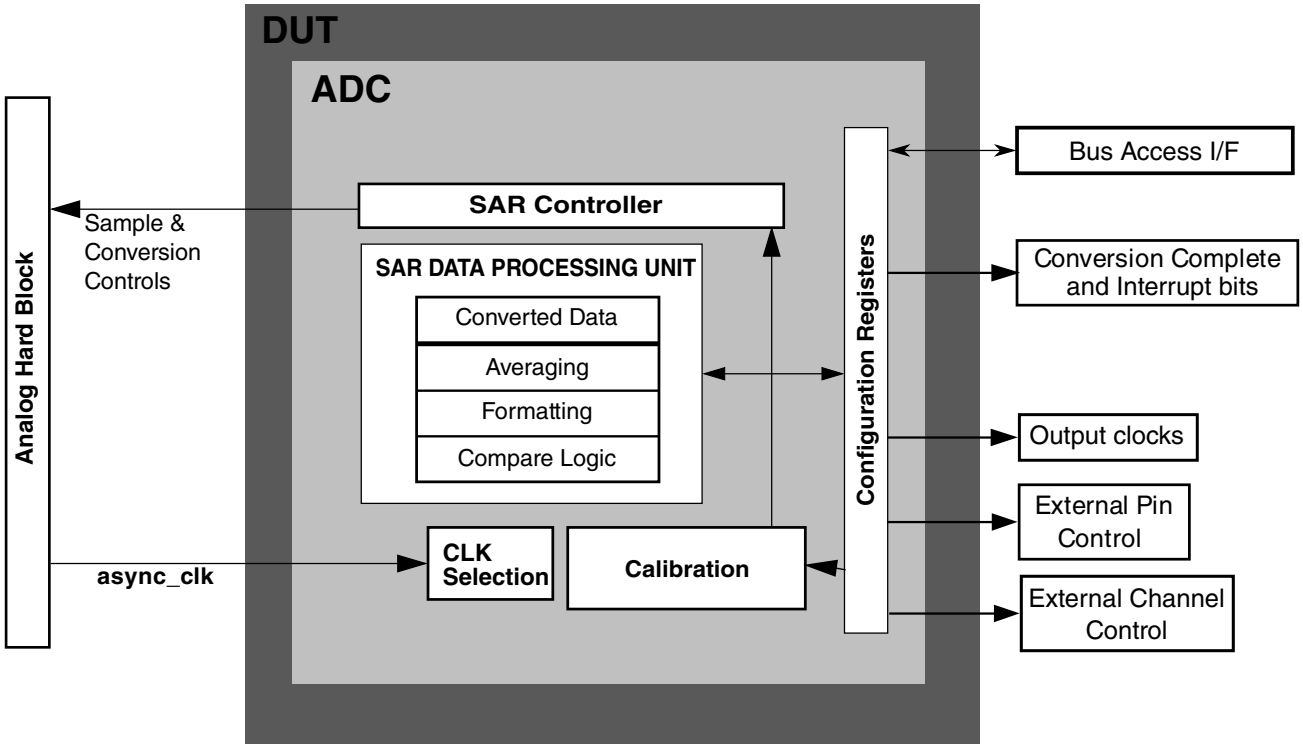


Figure 53-1. ADC I/F block diagram

53.2.3 ADC block diagram

The following figure shows a top-level block diagram of the ADC module.

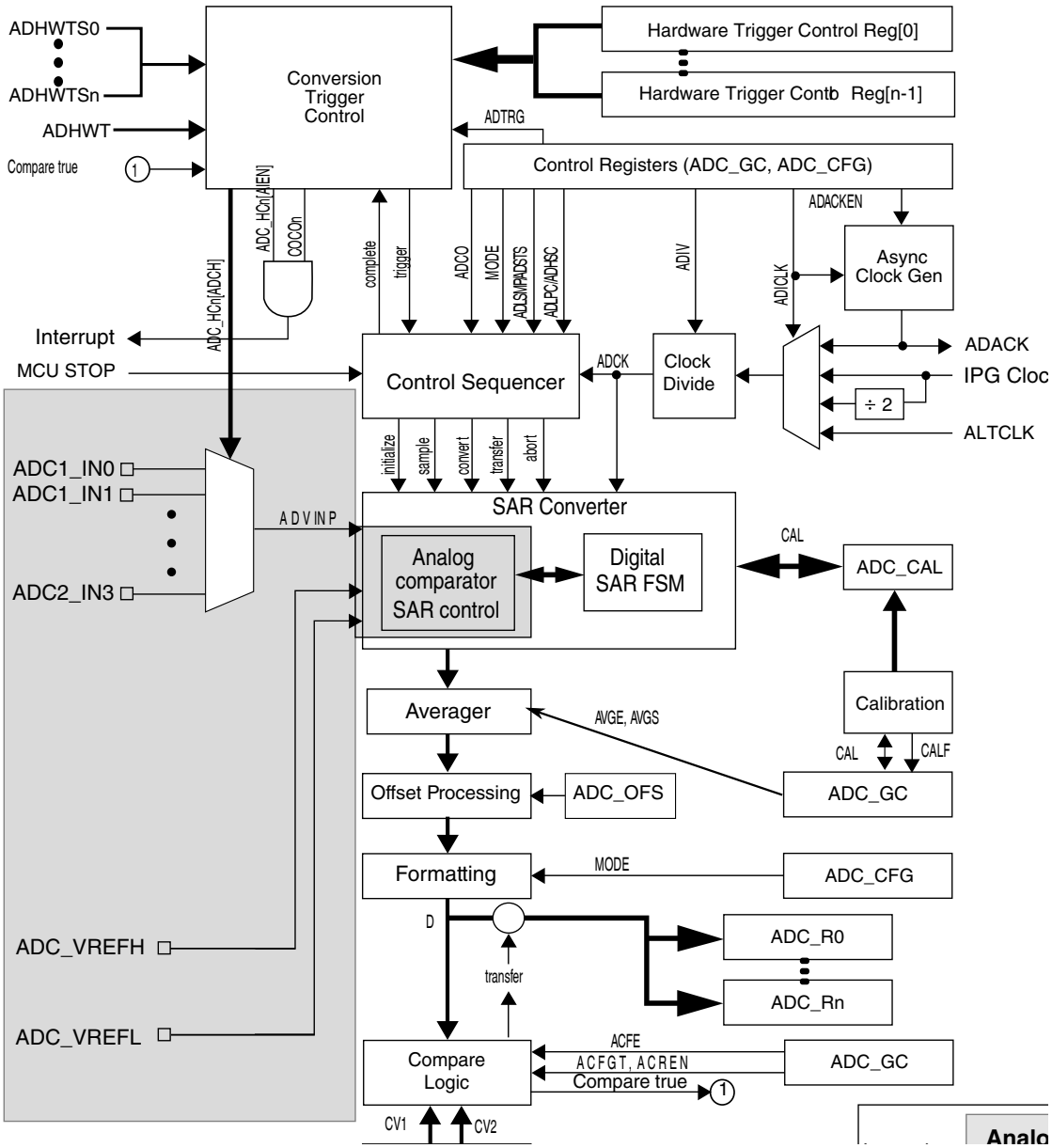


Figure 53-2. ADC block diagram

53.2.4 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.

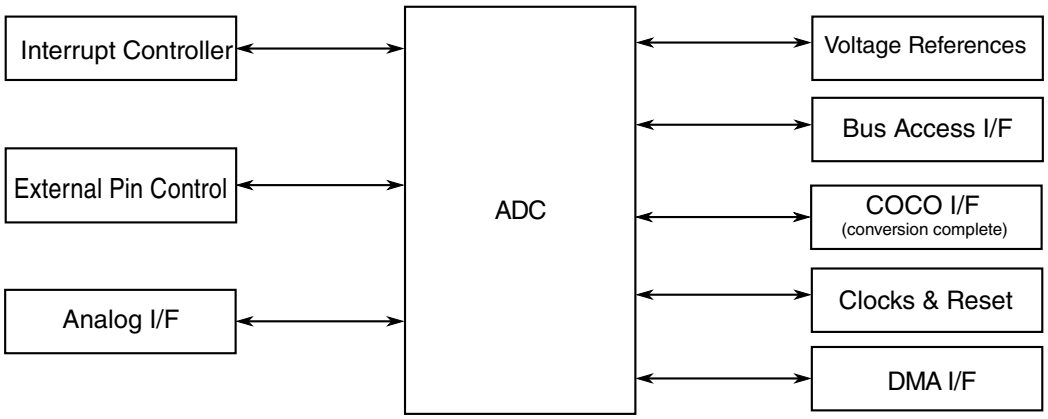


Figure 53-3. ADC module interface

53.2.5 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.

53.3 External Signals

The following table describes the external signals of ADC:

Table 53-2. ADC External Signals

Signal	Description	Pad	Mode	Direction
ADC1_IN0	Analog channel 1 input 0	GPIO_AD_00	-	I
ADC1_IN1	Analog channel 1 input 1	GPIO_AD_01	-	I
ADC1_IN2	Analog channel 1 input 2	GPIO_AD_02	-	I
ADC1_IN3	Analog channel 1 input 3	GPIO_AD_03	-	I
ADC1_IN4	Analog channel 1 input 4	GPIO_AD_04	-	I
ADC1_IN5	Analog channel 1 input 5	GPIO_AD_05	-	I
ADC1_IN6	Analog channel 1 input 6	GPIO_AD_06	-	I
ADC1_IN7	Analog channel 1 input 7	GPIO_AD_07	-	I

Table continues on the next page...

Table 53-2. ADC External Signals (continued)

Signal	Description	Pad	Mode	Direction
ADC1_IN8	Analog channel 1 input 8	GPIO_AD_08	-	I
ADC1_IN9	Analog channel 1 input 9	GPIO_AD_09	-	I
ADC1_IN10	Analog channel 1 input 10	GPIO_AD_10	-	I
ADC1_IN11	Analog channel 1 input 11	GPIO_AD_11	-	I
ADC1_IN12	Analog channel 1 input 12	GPIO_AD_12	-	I
ADC1_IN13	Analog channel 1 input 13	GPIO_AD_13	-	I
ADC1_IN14	Analog channel 1 input 14	GPIO_AD_14	-	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.

53.4 Clocks

The following table describes the clock sources for ADC.

Please see Clock Controller Module (CCM) for clock setting, configuration and gating information.

Table 53-3. ADC Clocks

Clock name	Clock Root	Description
ipg_clk	ipg_clk_root	Peripheral clock

53.5 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.

53.5.1 Clock Select and Divide Control

The ADC digital module has three clock sources:

- IPG clock

- Alternate Clock (ALTCLK) is connected to the signal as described in the chip-specific ADC information.
- 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
- ALTCLK
- ADACK

From the four clocks listed above, one is chosen depending on the configuration of ADCLK[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	Alternate clock (ALTCLK)
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.
- ALTCLK, as defined for this chip.
- 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.

53.5.2 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].

53.5.3 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).

53.5.4 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.

53.5.4.1 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.

53.5.4.2 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.

53.5.4.3 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.

53.5.4.4 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.

53.5.4.5 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)

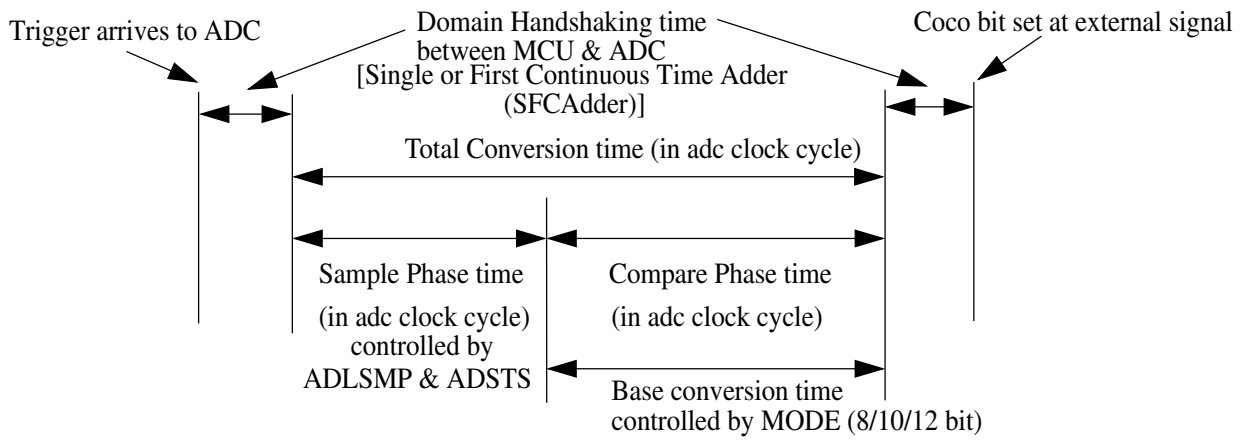


Figure 53-4. ADC conversion time details

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 1934. Refer to Table 53-4 through Table 53-7 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 53-4. 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 53-4. 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 53-5. 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 53-6. 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 53-7. 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.

53.5.4.6 Conversion Time Examples

The following examples uses Equation 1 on page 1934 and the information provided in tables Table 53-4 through Table 53-7.

53.5.4.6.1 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 1934 and the information provided in Table 53-8 through Table 53-10. The table below list the variables of Equation 1 on page 1934.

Table 53-8. 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 53-8. So for Bus clock equal to 40 Mhz and ADCK equal to 40 Mhz the resulting conversion time is 0.85 us.

53.5.4.6.2 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 53-4 through Table 53-7. The table below lists the variables of equation.

Table 53-9. 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 53-9. 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.

53.5.4.6.3 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 53-4 to Table 53-7. The table below list the variables of equation.

Table 53-10. 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 53-10. So for Bus clock equal to 40Mhz and ADCK equal to 40Mhz the resulting conversion time is 650 ns.

53.5.4.7 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).

53.5.5 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 53-11. 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.

53.5.6 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.

53.5.7 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.

53.5.8 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. The use of ALTCLK as the conversion clock source in wait is dependent on the definition of ALTCLK for the specific MCU.

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.

53.5.9 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.

53.5.9.1 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.

53.5.9.2 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.

53.6 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.

53.6.1 ADC Module Initialization Example

This section describes the initialization sequence along with pseudo-code.

53.6.1.1 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.

53.6.1.2 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.
```

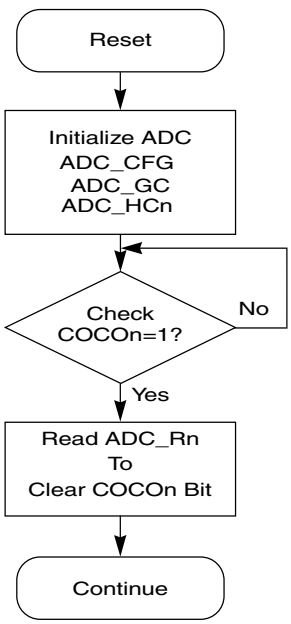


Figure 53-5. Initialization Flowchart for Example

53.7 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.

53.7.1 Sources of Error

Several sources of error exist for A/D conversions. These are discussed in the following sections.

53.7.1.1 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.

53.7.1.2 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).

53.7.1.3 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.

53.7.1.4 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.

53.7.1.5 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.

53.7.1.6 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.

53.8 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	53.8.1/1950
400C_4004	Control register for hardware triggers (ADC1_HC1)	32	R/W	0000_001Fh	53.8.2/1951
400C_4008	Control register for hardware triggers (ADC1_HC2)	32	R/W	0000_001Fh	53.8.2/1951
400C_400C	Control register for hardware triggers (ADC1_HC3)	32	R/W	0000_001Fh	53.8.2/1951
400C_4010	Control register for hardware triggers (ADC1_HC4)	32	R/W	0000_001Fh	53.8.2/1951

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	53.8.2/1951
400C_4018	Control register for hardware triggers (ADC1_HC6)	32	R/W	0000_001Fh	53.8.2/1951
400C_401C	Control register for hardware triggers (ADC1_HC7)	32	R/W	0000_001Fh	53.8.2/1951
400C_4020	Status register for HW triggers (ADC1_HS)	32	R (reads 0)	0000_0000h	53.8.3/1953
400C_4024	Data result register for HW triggers (ADC1_R0)	32	R	0000_0000h	53.8.4/1955
400C_4028	Data result register for HW triggers (ADC1_R1)	32	R	0000_0000h	53.8.5/1956
400C_402C	Data result register for HW triggers (ADC1_R2)	32	R	0000_0000h	53.8.5/1956
400C_4030	Data result register for HW triggers (ADC1_R3)	32	R	0000_0000h	53.8.5/1956
400C_4034	Data result register for HW triggers (ADC1_R4)	32	R	0000_0000h	53.8.5/1956
400C_4038	Data result register for HW triggers (ADC1_R5)	32	R	0000_0000h	53.8.5/1956
400C_403C	Data result register for HW triggers (ADC1_R6)	32	R	0000_0000h	53.8.5/1956
400C_4040	Data result register for HW triggers (ADC1_R7)	32	R	0000_0000h	53.8.5/1956
400C_4044	Configuration register (ADC1_CFG)	32	R/W	0000_0200h	53.8.6/1957
400C_4048	General control register (ADC1_GC)	32	R/W	0000_0000h	53.8.7/1960
400C_404C	General status register (ADC1_GS)	32	R/W	0000_0000h	53.8.8/1962
400C_4050	Compare value register (ADC1_CV)	32	R/W	0000_0000h	53.8.9/1963
400C_4054	Offset correction value register (ADC1_OFS)	32	R/W	0000_0000h	53.8.10/ 1964
400C_4058	Calibration value register (ADC1_CAL)	32	R/W	0000_0000h	53.8.11/ 1964

53.8.1 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: 400C_4000h base + 0h offset = 400C_4000h

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

Table continues on the next page...

ADCx_HC0 field descriptions (continued)

Field	Description
	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.

53.8.2 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: 400C_4000h base + 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_HCn field descriptions

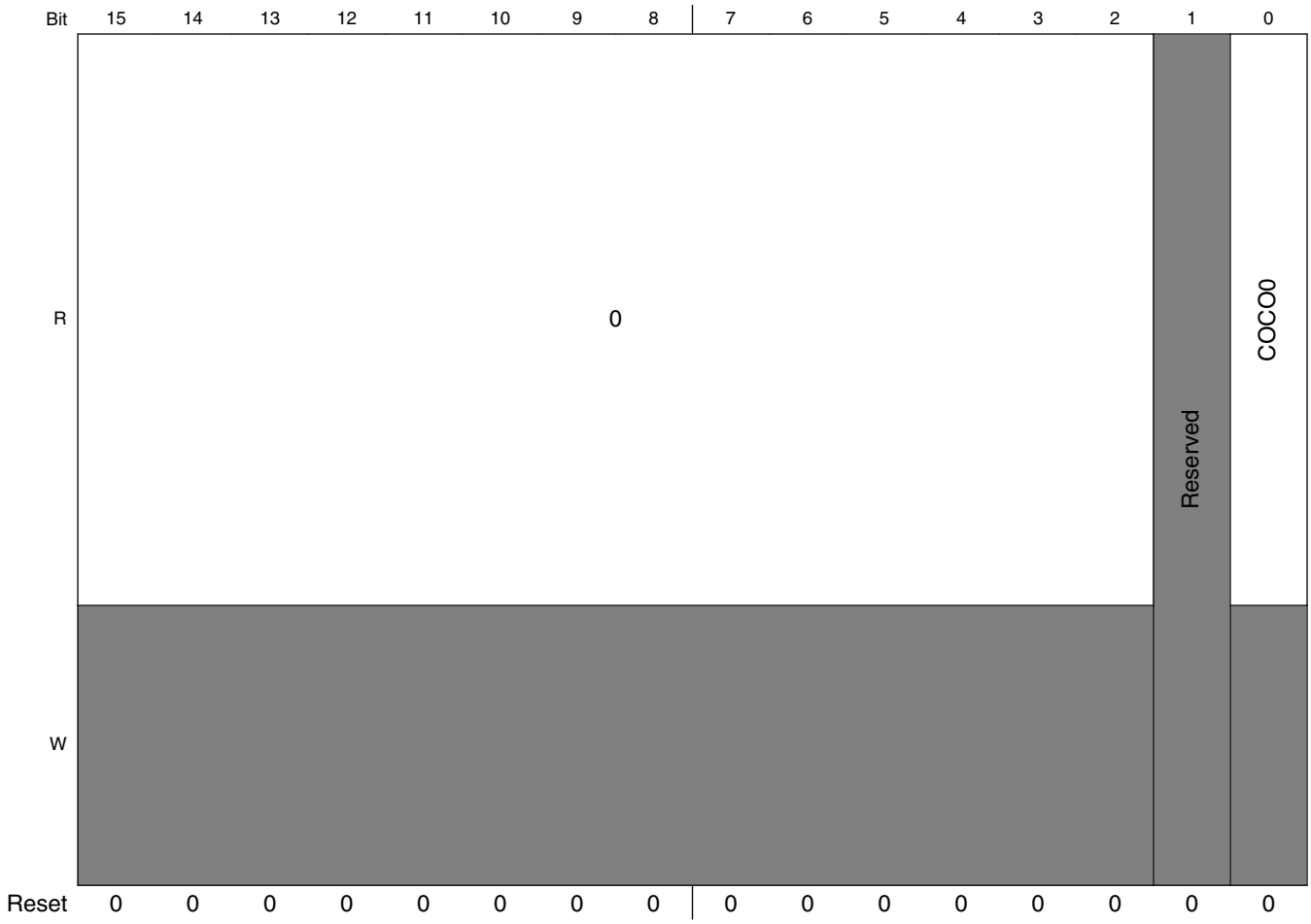
Field	Description
31–8 Reserved	This read-only field is reserved and always has the value 0.
7 AIEN	Conversion Complete Interrupt Enable/Disable Control An interrupt is generated whenever ADC_HS[COCO0]=1(conversion ADC_HC0 completed), provided the corresponding interrupt is enabled. 1 Conversion complete 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.

53.8.3 Status register for HW triggers (ADC*x*_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: 400C_4000h base + 20h offset = 400C_4020h

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



ADCx_HS field descriptions

Field	Description
31–2 Reserved	This read-only field is reserved and always has the value 0.
1 -	This field is reserved. Reserved
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.

53.8.4 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 53-12. 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: 400C_4000h base + 24h offset = 400C_4024h

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																				CDATA											
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_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)

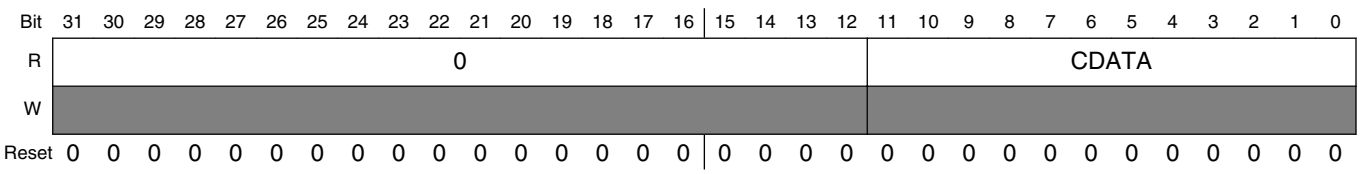
53.8.5 Data result register for HW triggers (ADC*x*_R*n*)

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 53-13. 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: 400C_4000h base + 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)

53.8.6 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: 400C_4000h base + 44h offset = 400C_4044h

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...

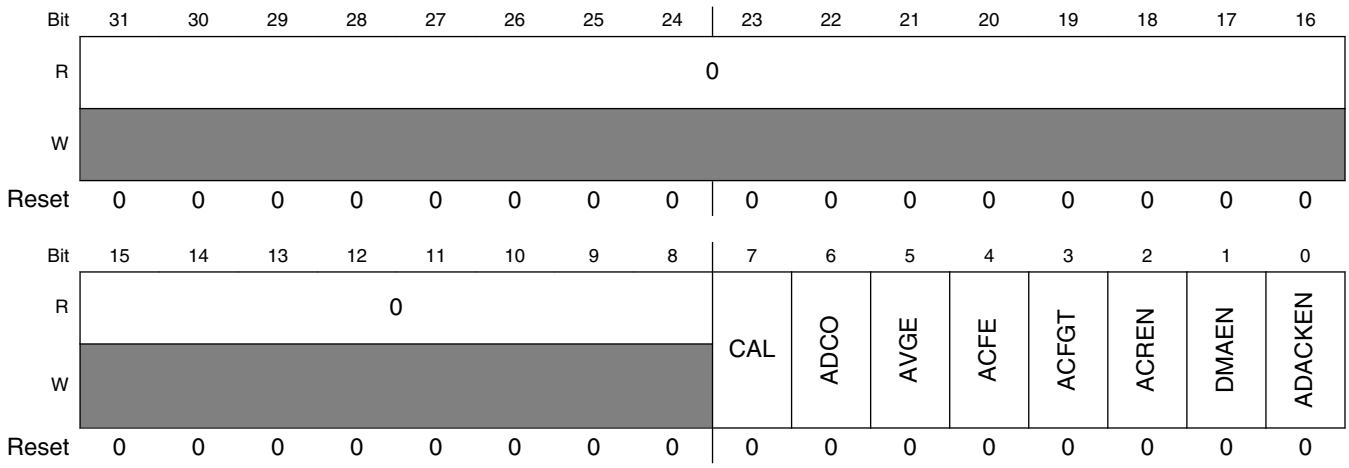
ADCx_CFG field descriptions (continued)

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 Alternate clock (ALTCLK) 11 Asynchronous clock (ADACK)

53.8.7 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: 400C_4000h base + 48h offset = 400C_4048h



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...

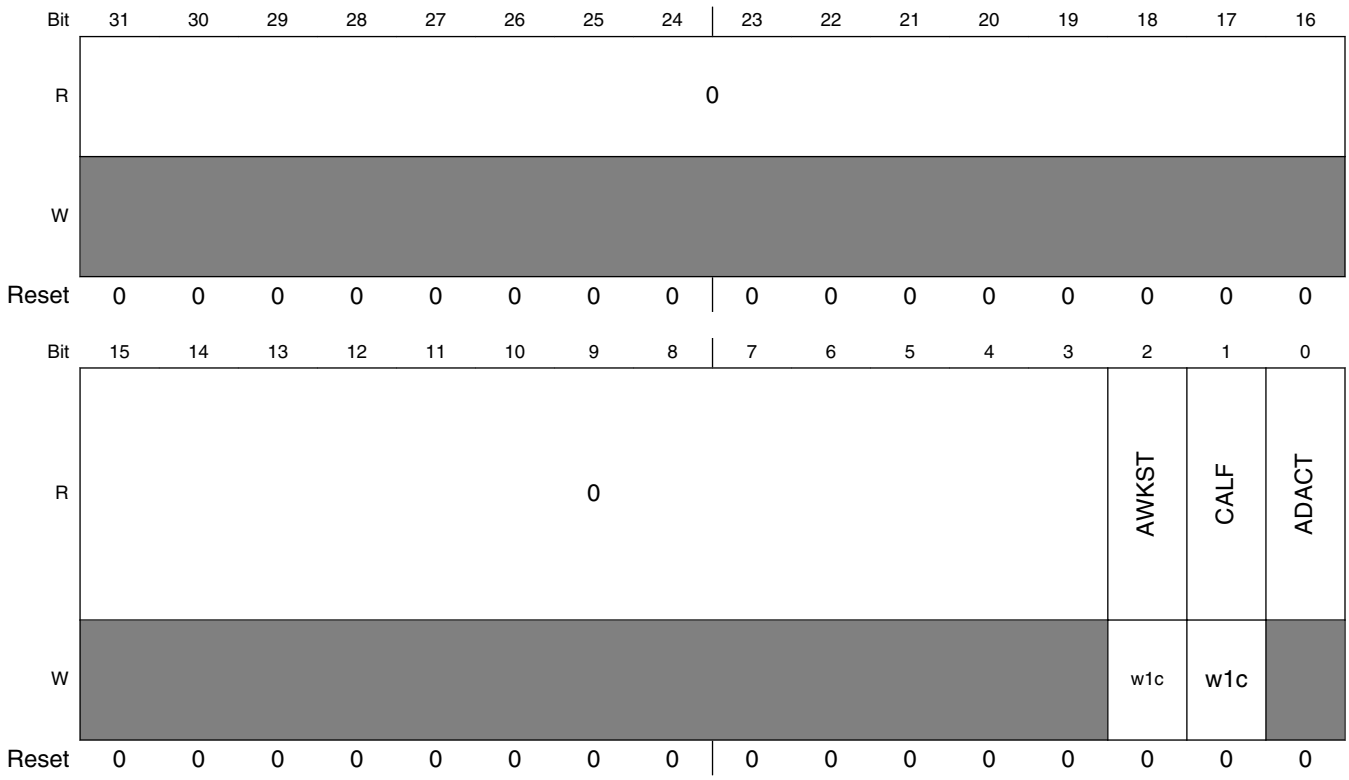
ADCx_GC field descriptions (continued)

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

53.8.8 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: 400C_4000h base + 4Ch offset = 400C_404Ch



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.

53.8.9 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: 400C_4000h base + 50h offset = 400C_4050h

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

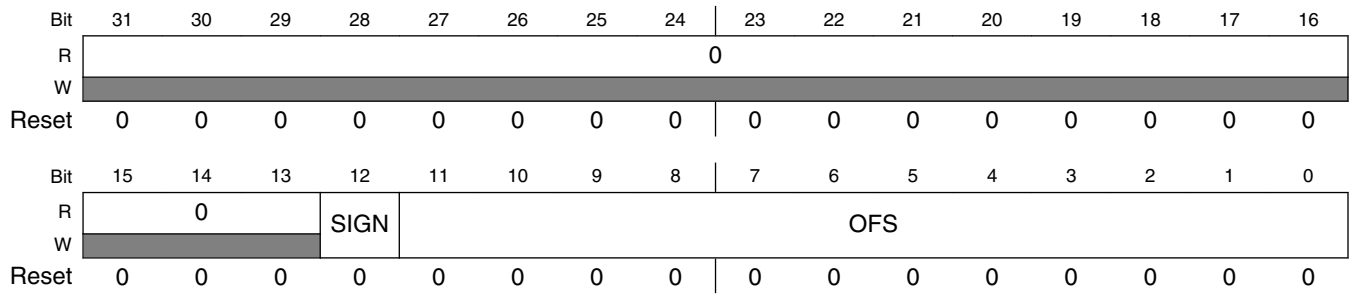
ADCx_CV field descriptions

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).

53.8.10 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: 400C_4000h base + 54h offset = 400C_4054h



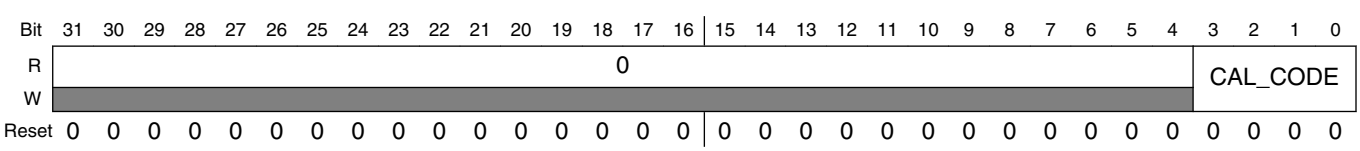
ADCx_OFS field descriptions

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.

53.8.11 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: 400C_4000h base + 58h offset = 400C_4058h



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 54
ADC External Trigger Control (ADC_ETC)

54.1 Chip-specific ADC_ETC 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
XBAR resource assignments	XBAR	XBAR Resource Assignments

NOTE

No TSC module and trigger_in[7:4] input signals in this device.

NOTE

Only 1x ADC in this device, so SyncMode or AsyncMode is not applicable.

54.2 About this module

54.2.1 Introduction

The ADC_ETC module enables multiple users share the ADC modules in a TIME-Division-Multiplexing (TDM) way. The external triggers can be from Cross BAR(XBAR) and TSC in SOC. The ADC_ETC has two channels, each channel can support one TSC and four external triggers from XBAR to one ADC module. The ADC_ETC can support interrupt mode and DMA mode.

54.2.2 Features

The ADC_ETC includes the following features:

- External trigger interface with dual ADCs, support up to 8 Hardware External Trigger (HWT) control for each ADC
- Capable of triggering dual ADC in SyncMode or AsyncMode:
  - In SyncMode ADC1 is controlled by the same trigger source.
  - In AsyncMode ADC1 is controlled by separate trigger source.
- Support up to four external trigger inputs for each ADC:
  - Four single to multiple (up to 8) trigger sources. One external trigger results in multiple sequential triggers to ADC.
  - Flexible ADC trigger interval and initial delay control.
  - Each trigger sources can be configured as HW or SW trigger mode.
- ADC result holding and status reporting
- External trigger auto hold and arbitration
  - Each external trigger can be configured with a fixed priority. External trigger with the highest priority is severed first.
  - Hold one trigger event upon arbitration lose or ADC busy.
- Support ADC trigger interface cascading
- Support interrupt mode and DMA mode

54.3 Block diagram

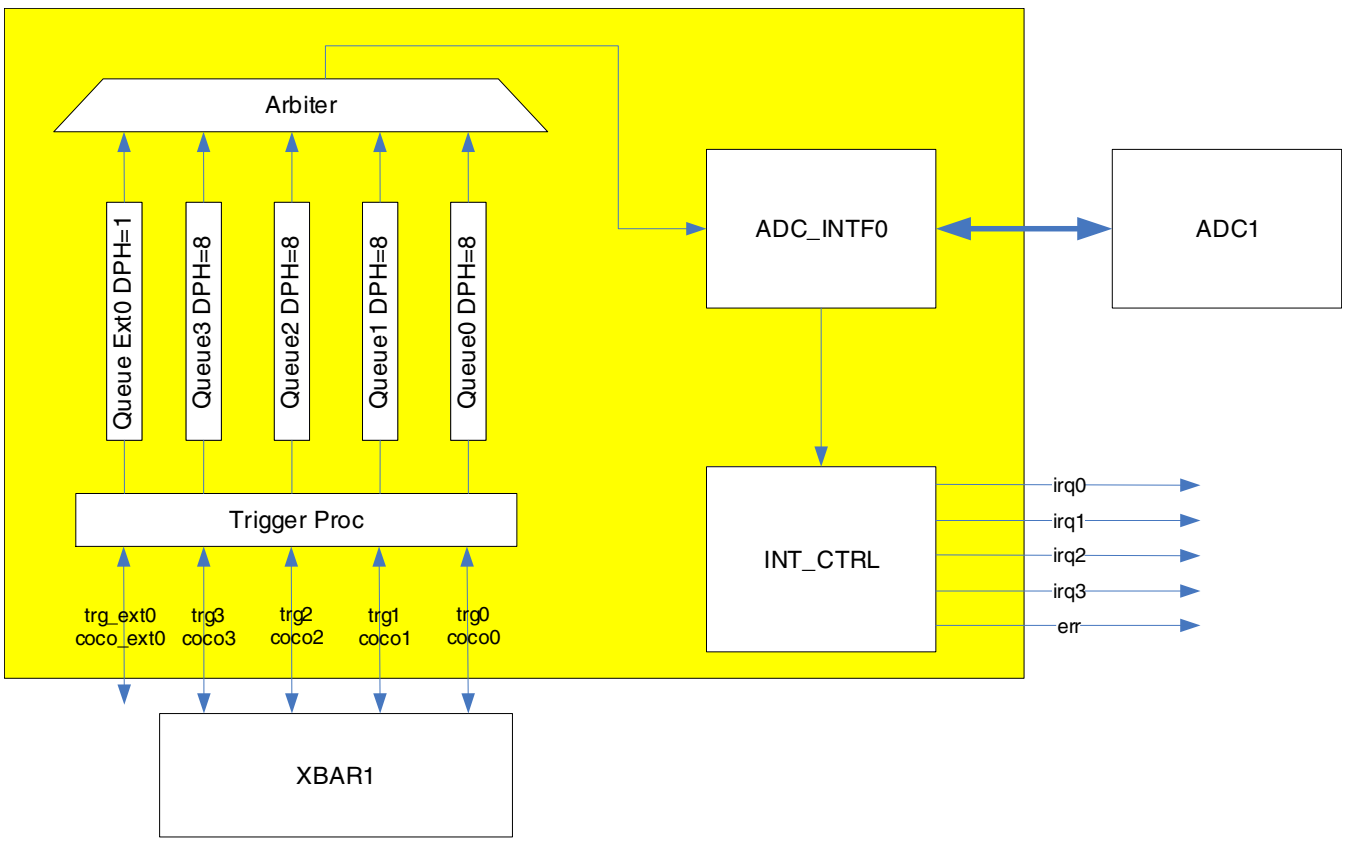


Figure 54-1. ADC_ETC block diagram

54.4 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.

54.4.1 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.

As XBAR and TSC input can be SYNC or ASYNC with the ADC_ETC clock, the ipg_clk need to be set that can capture the trigger input edge after the double synchronous circuit.

54.4.2 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. While the sw_reset_n is a software reset, which can reset all configuration registers by software.

54.4.3 Operations

This section describes the ADC_ETC module's operations.

The trigger process is used to process the multiple input source, trigger_enable can be used to mask the hardware trigger source. Software can select the hardware trigger or software trigger by configuration, and the selected source will be double synchronized and detected the posedge pulse before request to the arbiter. SYNC_MODE process only apply to the second ADC module inputs.

The trigger process diagram is indicated as below:

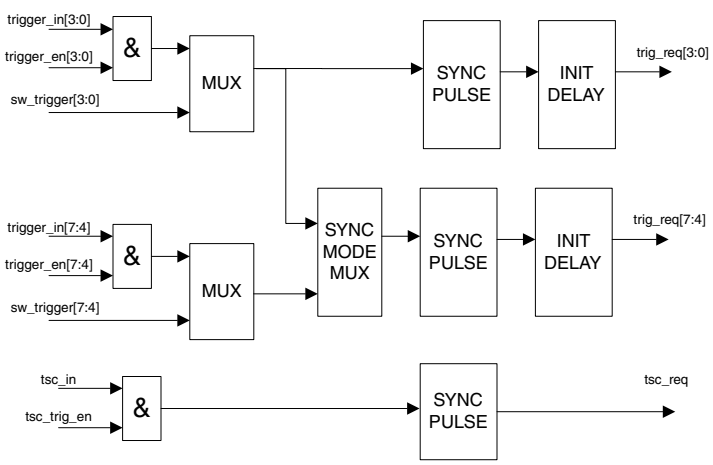


Figure 54-2. Trigger process diagram

The trigger requests from XBAR and TSC have fixed priority setting before go to the arbiter. There are total eight priority levels available for each trigger request, 7 is the highest priority level while 0 is the lowest priority level. If multiple trigger requests have

the same priority level, then the round-robin scheme will be applied for the requests to the ADC trigger source. The trigger request will be granted only when previous trigger had finished and the ADC module is IDLE. When in SYNC mode, the trigger request will be granted only when both ADC modules are IDLE state.

When there is trigger request that is granted, the ETC sends these signals to ADC according to corresponding configurations: Multiple ADC triggers, HWTS[7:0] as corresponding hardware trigger select signal, and with CSEL[3:0]. When ADC finished conversion, the ADC would send back a complete signal adc_coco to ETC, then ETC will store correct ADC converted results into the internal registers, and clear the ADC COCO signals at correct order. All parts of this steps are automatically done by hardware. DMA request or interrupts will assert according to the software configurations.

Each ADC trigger of the eight external trigger sources has separate configurations by software, including Interrupt Enable (IE), Back to Back ADC trigger (B2B), HWTS and CSEL to ADC. The ADC_ETC have four interrupts output of Done0, Done1, Done2 and Error Interrupt. The Done0/1/2 interrupts are controlled by software for each ADC trigger (i.e. 2'b00: No interrupt when finished; 2'b01: Finished interrupt on Done0; 2'b10: Finished interrupt on Done1; 2'b11: Finished interrupt on Done2). Error interrupt occurrs 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:

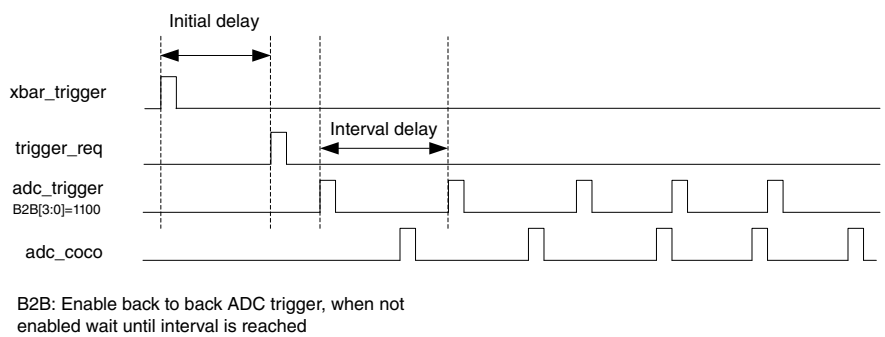


Figure 54-3. ETC ADC interface timing

54.5 Memory Map and register definition

This section includes the ADC_ETC module memory map and detailed descriptions of all registers.

54.5.1 ADC_ETC register descriptions

ADC_ETC memory map

54.5.1.1 ADC_ETC Memory map

ADC_ETC base address: 4008_8000h

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	RW	0000_0000h
8h	ETC DONE_2 and DONE_ERR IRQ State Register (DONE2_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
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

Table continues on the next page...

Offset	Register	Width (In bits)	Access	Reset value
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

54.5.1.2 ADC_ETC Global Control Register (CTRL)

54.5.1.2.1 Offset

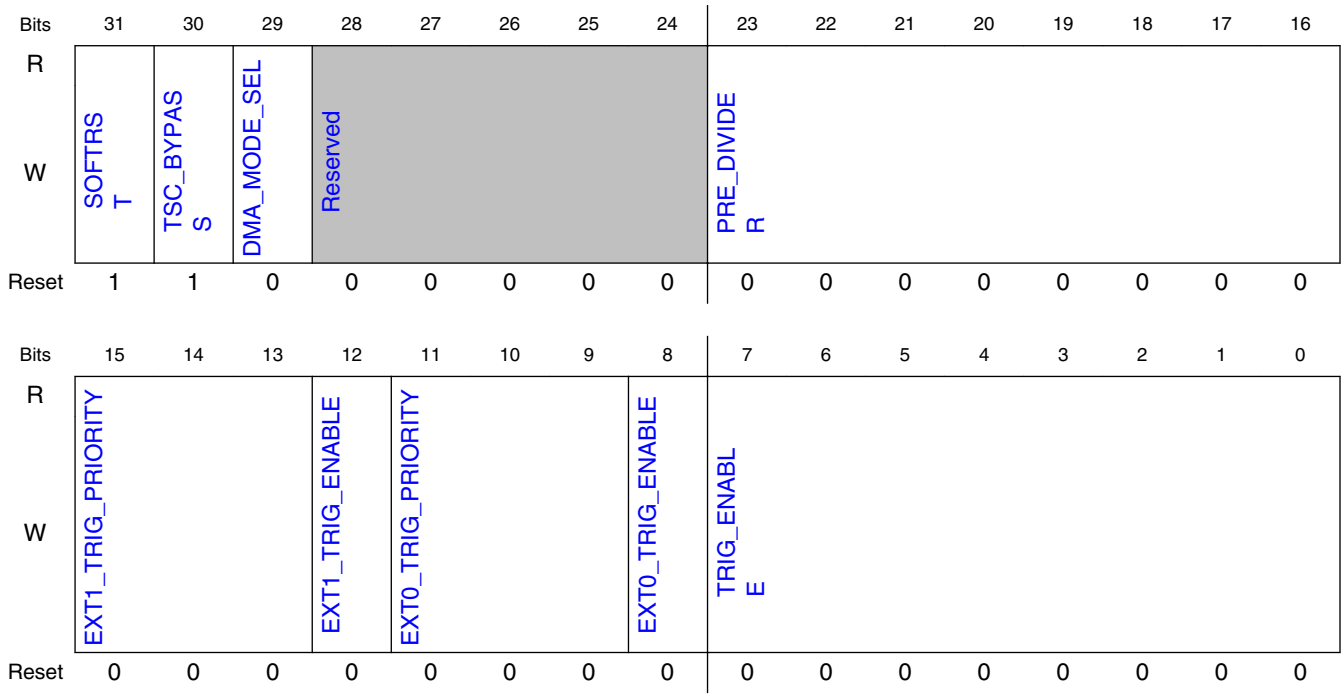
Register	Offset
CTRL	0h

54.5.1.2.2 Function

ADC_ETC global control register

This register controls various high-level functions of the ADC_ETC

54.5.1.2.3 Diagram



54.5.1.2.4 Fields

Field	Function
31 SOFTRST	Software reset, high active. When write 1 ,all logical will be reset.
30 TSC_BYPASS	1'b1: TSC is bypassed to ADC2. 1'b0: TSC not bypassed. **NOTE:** To use ADC2, this bit should be cleared.
29 DMA_MODE_SEL	1'b0: Trig DMA_REQ with latched signal, REQ will be cleared when ACK and source request cleared. 1'b1: Trig DMA_REQ with pulsed signal, REQ will be cleared by ACK only.
28-24 —	Reserved
23-16 PRE_DIVIDER	Pre-divider for trig delay and interval .
15-13 EXT1_TRIG_PRIORITY	External TSC1 trigger priority, 7 is Highest, 0 is lowest .
12 EXT1_TRIG_ENABLE	TSC1 TRIG enable register. 1'b1: enable external TSC1 trigger. 1'b0: disable external TSC1 trigger.

Table continues on the next page...

Field	Function
11-9 EXT0_TRIG_PRIORITY	External TSC0 trigger priority, 7 is Highest, 0 is lowest .
8 EXT0_TRIG_ENABLE	TSC0 TRIG enable register. 1'b1: enable external TSC0 trigger. 1'b0: disable external TSC0 trigger.
7-0 TRIG_ENABLE	TRIG enable register. 1'b1: enable correspond external XBAR trigger [7 ... 0] . 1'b0: disable correspond external XBAR trigger [7 ... 0] .

54.5.1.3 ETC DONE0 and DONE1 IRQ State Register (DONE0_1_IRQ)

54.5.1.3.1 Offset

Register	Offset
DONE0_1_IRQ	4h

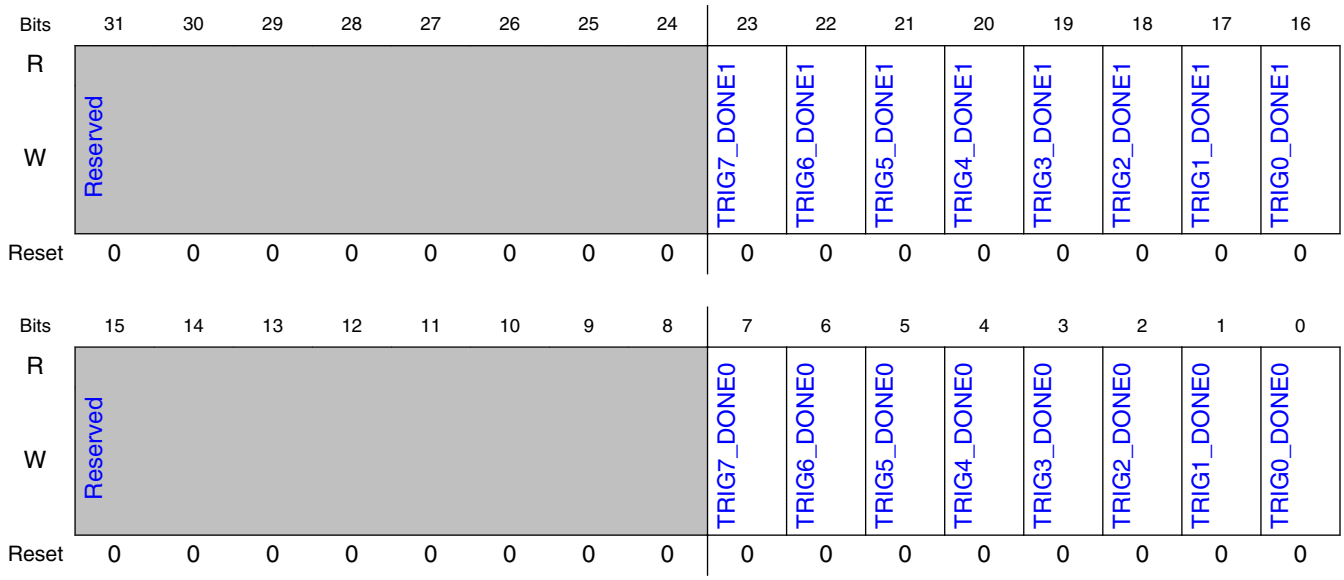
54.5.1.3.2 Function

This register has the state for ETC DONE0 and DONE1 interrupts

NOTE

The IRQs are cleared by writing a logic 1 to the bits.

54.5.1.3.3 Diagram



54.5.1.3.4 Fields

Field	Function
31-24 —	Reserved
23 TRIG7_DONE1	TRIG7 done1 interrupt detection
22 TRIG6_DONE1	TRIG6 done1 interrupt detection
21 TRIG5_DONE1	TRIG5 done1 interrupt detection
20 TRIG4_DONE1	TRIG4 done1 interrupt detection
19 TRIG3_DONE1	TRIG3 done1 interrupt detection
18 TRIG2_DONE1	TRIG2 done1 interrupt detection
17 TRIG1_DONE1	TRIG1 done1 interrupt detection
16 TRIG0_DONE1	TRIG0 done1 interrupt detection
15-8 —	Reserved

Table continues on the next page...

Field	Function
7 TRIG7_DONE0	TRIG7 done0 interrupt detection
6 TRIG6_DONE0	TRIG6 done0 interrupt detection
5 TRIG5_DONE0	TRIG5 done0 interrupt detection
4 TRIG4_DONE0	TRIG4 done0 interrupt detection
3 TRIG3_DONE0	TRIG3 done0 interrupt detection
2 TRIG2_DONE0	TRIG2 done0 interrupt detection
1 TRIG1_DONE0	TRIG1 done0 interrupt detection
0 TRIG0_DONE0	TRIG0 done0 interrupt detection

54.5.1.4 ETC DONE_2 and DONE_ERR IRQ State Register (DONE2_ERR_IRQ)

54.5.1.4.1 Offset

Register	Offset
DONE2_ERR_IRQ	8h

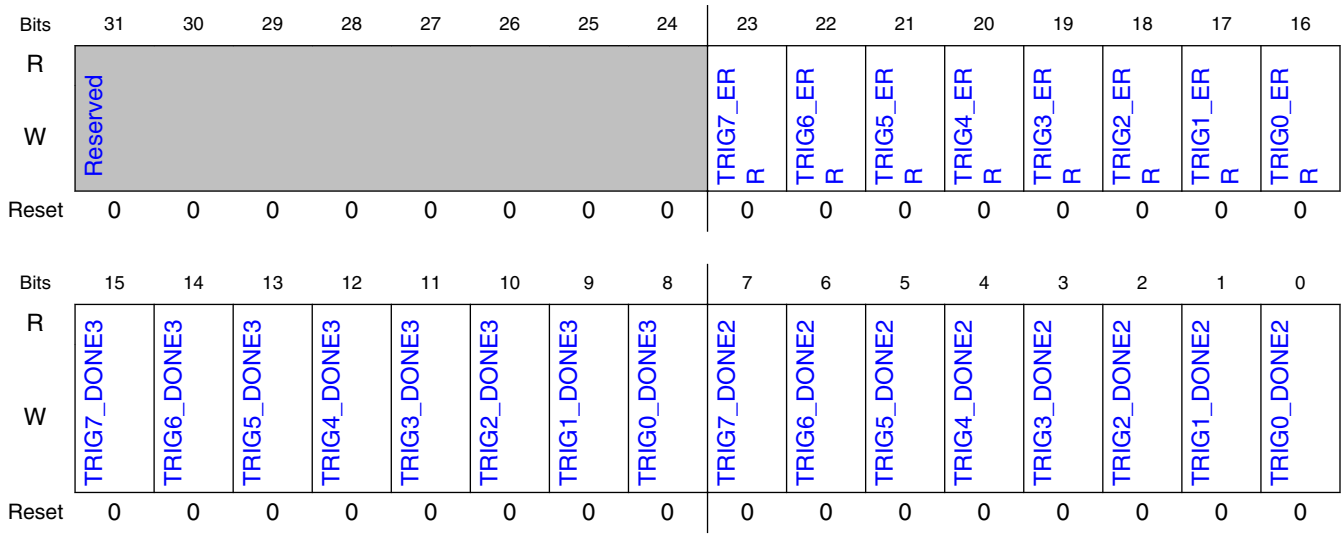
54.5.1.4.2 Function

This register has the state for ETC DONE2 and DONE_ERR interrupts

NOTE

The IRQs are cleared by writing a logic 1 to the bits.

54.5.1.4.3 Diagram



54.5.1.4.4 Fields

Field	Function
31-24 —	Reserved
23 TRIG7_ERR	TRIG7 error interrupt detection
22 TRIG6_ERR	TRIG6 error interrupt detection
21 TRIG5_ERR	TRIG5 error interrupt detection
20 TRIG4_ERR	TRIG4 error interrupt detection
19 TRIG3_ERR	TRIG3 error interrupt detection
18 TRIG2_ERR	TRIG2 error interrupt detection
17 TRIG1_ERR	TRIG1 error interrupt detection
16 TRIG0_ERR	TRIG0 error interrupt detection
15 TRIG7_DONE3	TRIG7 done3 interrupt detection
14	TRIG6 done3 interrupt detection

Table continues on the next page...

Field	Function
TRIG6_DONE3	
13 TRIG5_DONE3	TRIG5 done3 interrupt detection
12 TRIG4_DONE3	TRIG4 done3 interrupt detection
11 TRIG3_DONE3	TRIG3 done3 interrupt detection
10 TRIG2_DONE3	TRIG2 done3 interrupt detection
9 TRIG1_DONE3	TRIG1 done3 interrupt detection
8 TRIG0_DONE3	TRIG0 done3 interrupt detection
7 TRIG7_DONE2	TRIG7 done2 interrupt detection
6 TRIG6_DONE2	TRIG6 done2 interrupt detection
5 TRIG5_DONE2	TRIG5 done2 interrupt detection
4 TRIG4_DONE2	TRIG4 done2 interrupt detection
3 TRIG3_DONE2	TRIG3 done2 interrupt detection
2 TRIG2_DONE2	TRIG2 done2 interrupt detection
1 TRIG1_DONE2	TRIG1 done2 interrupt detection
0 TRIG0_DONE2	TRIG0 done2 interrupt detection

54.5.1.5 ETC DMA control Register (DMA_CTRL)

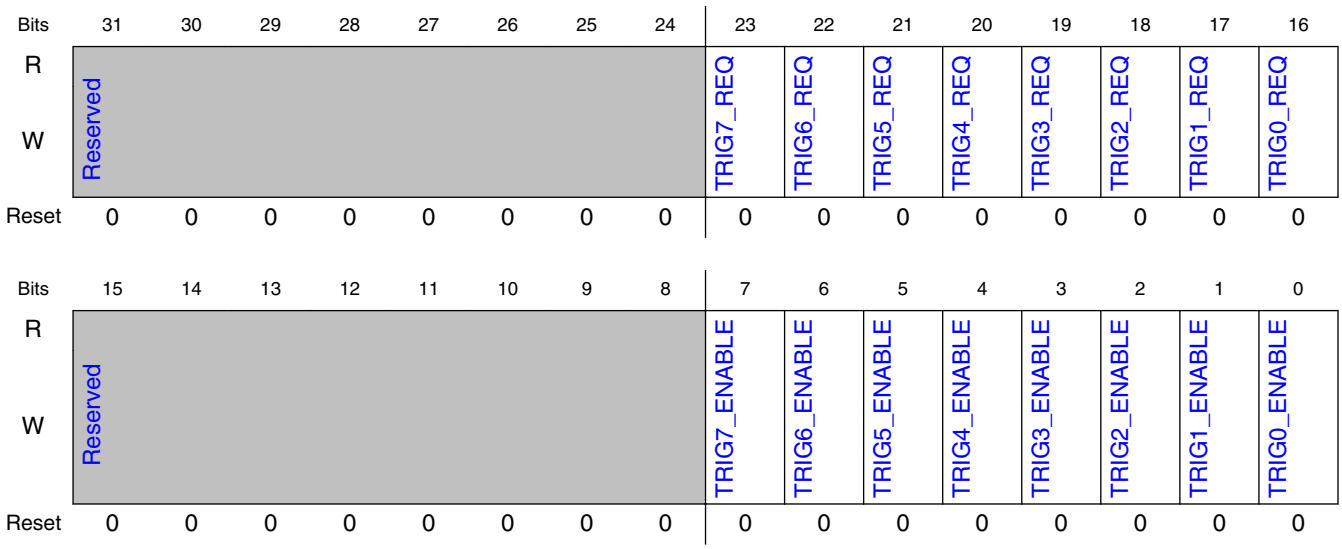
54.5.1.5.1 Offset

Register	Offset
DMA_CTRL	Ch

54.5.1.5.2 Function

This register has the DMA control for ETC

54.5.1.5.3 Diagram



54.5.1.5.4 Fields

Field	Function
31-24 —	Reserved
23 TRIG7_REQ	When TRIG7 done DMA request detection
22 TRIG6_REQ	When TRIG6 done DMA request detection
21 TRIG5_REQ	When TRIG5 done DMA request detection
20 TRIG4_REQ	When TRIG4 done DMA request detection
19 TRIG3_REQ	When TRIG3 done DMA request detection
18 TRIG2_REQ	When TRIG2 done DMA request detection
17 TRIG1_REQ	When TRIG1 done DMA request detection

Table continues on the next page...

Field	Function
16 TRIG0_REQ	When TRIG0 done DMA request detection
15-8 —	Reserved
7 TRIG7_ENABLE	When TRIG7 done enable DMA request
6 TRIG6_ENABLE	When TRIG6 done enable DMA request
5 TRIG5_ENABLE	When TRIG5 done enable DMA request
4 TRIG4_ENABLE	When TRIG4 done enable DMA request
3 TRIG3_ENABLE	When TRIG3 done enable DMA request
2 TRIG2_ENABLE	When TRIG2 done enable DMA request
1 TRIG1_ENABLE	When TRIG1 done enable DMA request
0 TRIG0_ENABLE	When TRIG0 done enable DMA request

54.5.1.6 ETC_TRIG Control Register (TRIG0_CTRL - TRIG3_CTRL)

54.5.1.6.1 Offset

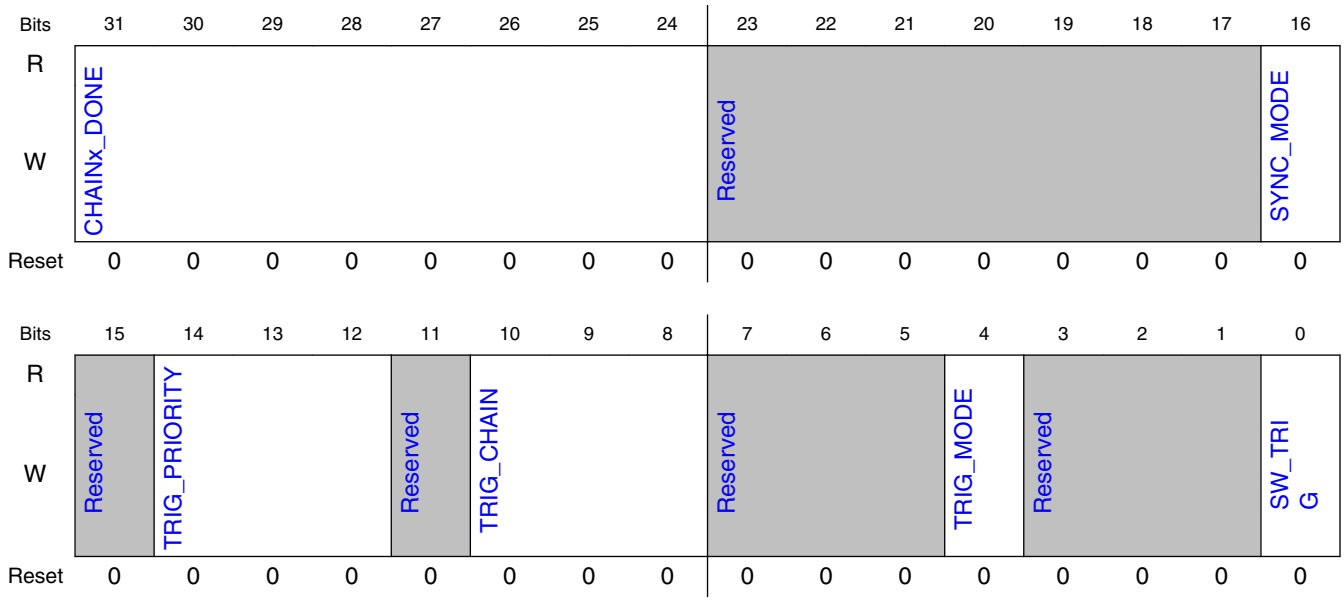
Register	Offset
TRIG0_CTRL	10h
TRIG1_CTRL	38h
TRIG2_CTRL	60h
TRIG3_CTRL	88h

54.5.1.6.2 Function

ETC_TRIG control register

This register controls various functions of the ETC_TRIG

54.5.1.6.3 Diagram



54.5.1.6.4 Fields

Field	Function
31-24 CHAINx_DONE	CHAINx done interrupt detection • bit 0: CHAIN0 done interrupt • bit 1: CHAIN1 done interrupt • bit 2: CHAIN2 done interrupt • bit 3: CHAIN3 done interrupt • bit 4: CHAIN4 done interrupt • bit 5: CHAIN5 done interrupt • bit 6: CHAIN6 done interrupt • bit 7: CHAIN7 done interrupt **NOTE:** The done interrupts are cleared by writing a logic 1 to the bits.
23-17 —	Reserved
16 SYNC_MODE	TRIG mode control . 1'b0: Disable sync mode; 1'b1: Enable sync mode
15 —	Reserved
14-12 TRIG_PRIORITY	External trigger priority, 7 is highest, 0 is lowest .
11 —	Reserved
10-8	TRIG chain length to the ADC. 0: Trig length is 1; ... 7: Trig length is 8;

Table continues on the next page...

Field	Function
TRIG_CHAIN	
7-5 —	Reserved
4 TRIG_MODE	TRIG mode register. 1'b0: hardware trigger. 1'b1: software trigger.
3-1 —	Reserved
0 SW_TRIG	Software write 1 as the TRIGGER. This register is self-clearing.

54.5.1.7 ETC_TRIG Counter Register (TRIG0_COUNTER - TRIG3_COUNTER)

54.5.1.7.1 Offset

Register	Offset
TRIG0_COUNTER	14h
TRIG1_COUNTER	3Ch
TRIG2_COUNTER	64h
TRIG3_COUNTER	8Ch

54.5.1.7.2 Function

This register controls ETC_TRIG counter

54.5.1.7.3 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

54.5.1.7.4 Fields

Field	Function
31-16 SAMPLE_INTERVAL	TRIGGER sampling interval counter
15-0 INIT_DELAY	TRIGGER initial delay counter

54.5.1.8 ETC_TRIG Chain 0/1 Register (TRIG0_CHAIN_1_0 - TRIG3_CHAIN_1_0)

54.5.1.8.1 Offset

Register	Offset
TRIG0_CHAIN_1_0	18h
TRIG1_CHAIN_1_0	40h
TRIG2_CHAIN_1_0	68h
TRIG3_CHAIN_1_0	90h

54.5.1.8.2 Function

This register controls ETC_TRIG Chain 0/1 configuration

54.5.1.8.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	IE1_EN	IE1		B2B1	HWTS1								CSEL_1			
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	IE0_EN	IE0		B2B0	HWTS0								CSEL_0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

54.5.1.8.4 Fields

Field	Function
31 IE1_EN	IRQ enable
30-29 IE1	CHAIN1 IE • 2'b00: Finished Interrupt on Done0 • 2'b01: Finished Interrupt on Done1 • 2'b10: Finished Interrupt on Done2 • 2'b11: Finished Interrupt on Done3
28 B2B1	CHAIN1 B2B • 1'b0: Disable B2B, wait until interval is reached • 1'b1: Enable B2B, back to back ADC trigger
27-20 HWTS1	CHAIN1 HWTS ADC hardware trigger selection. For more information, see the ADC chapter.
19-16 CSEL1	CHAIN1 CSEL ADC channel selection
15 IE0_EN	IRQ enable
14-13 IE0	CHAIN0 IE • 2'b00: Finished Interrupt on Done0 • 2'b01: Finished Interrupt on Done1 • 2'b10: Finished Interrupt on Done2 • 2'b11: Finished Interrupt on Done3
12 B2B0	CHAIN0 B2B • 1'b0: Disable B2B, wait until interval is reached • 1'b1: Enable B2B, back to back ADC trigger
11-4 HWTS0	CHAIN0 HWTS ADC hardware trigger selection. For more information, see the ADC chapter.
3-0 CSEL0	CHAIN0 CSEL ADC channel selection

54.5.1.9 ETC_TRIG Chain 2/3 Register (TRIG0_CHAIN_3_2 - TRIG3_CHAIN_3_2)

54.5.1.9.1 Offset

Register	Offset
TRIG0_CHAIN_3_2	1Ch
TRIG1_CHAIN_3_2	44h
TRIG2_CHAIN_3_2	6Ch
TRIG3_CHAIN_3_2	94h

54.5.1.9.2 Function

This register controls ETC_TRIG Chain 2/3 configuration

54.5.1.9.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R W	IE3_EN	IE3		B2B3	HWTS3								CSEL3			
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	IE2_EN	IE2		B2B2	HWTS2								CSEL2			
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

54.5.1.9.4 Fields

Field	Function
31 IE3_EN	IRQ enable
30-29 IE3	CHAIN3 IE
28 B2B3	CHAIN3 B2B
27-20 HWTS3	CHAIN3 HWTS
19-16 CSEL3	CHAIN3 CSEL
15 IE2_EN	IRQ enable
14-13 IE2	CHAIN2 IE
12 B2B2	CHAIN2 B2B
11-4 HWTS2	CHAIN2 HWTS
3-0	CHAIN2 CSEL

Field	Function
CSEL2	

54.5.1.10 ETC_TRIG Chain 4/5 Register (TRIG0_CHAIN_5_4 - TRIG3_CHAIN_5_4)

54.5.1.10.1 Offset

Register	Offset
TRIG0_CHAIN_5_4	20h
TRIG1_CHAIN_5_4	48h
TRIG2_CHAIN_5_4	70h
TRIG3_CHAIN_5_4	98h

54.5.1.10.2 Function

This register controls ETC_TRIG Chain 4/5 configuration

54.5.1.10.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	IE5_EN	IE5		B2B5	HWTS5								CSEL5			
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	IE4_EN	IE4		B2B4	HWTS4								CSEL4			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

54.5.1.10.4 Fields

Field	Function
31 IE5_EN	IRQ enable

Table continues on the next page...

Field	Function
30-29 IE5	CHAIN5 IE
28 B2B5	CHAIN5 B2B
27-20 HWTS5	CHAIN5 HWTS
19-16 CSEL5	CHAIN5 CSEL
15 IE4_EN	IRQ enable
14-13 IE4	CHAIN4 IE
12 B2B4	CHAIN4 B2B
11-4 HWTS4	CHAIN4 HWTS
3-0 CSEL4	CHAIN4 CSEL

54.5.1.11 ETC_TRIG Chain 6/7 Register (TRIG0_CHAIN_7_6 - TRIG3_CHAIN_7_6)

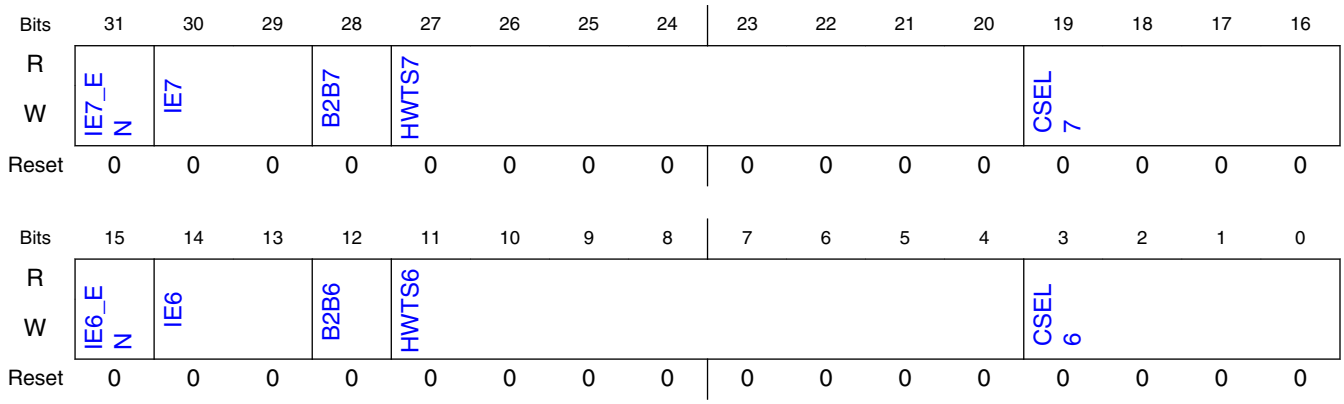
54.5.1.11.1 Offset

Register	Offset
TRIG0_CHAIN_7_6	24h
TRIG1_CHAIN_7_6	4Ch
TRIG2_CHAIN_7_6	74h
TRIG3_CHAIN_7_6	9Ch

54.5.1.11.2 Function

This register controls ETC_TRIG Chain 6/7 configuration

54.5.1.11.3 Diagram



54.5.1.11.4 Fields

Field	Function
31 IE7_EN	IRQ enable
30-29 IE7	CHAIN7 IE
28 B2B7	CHAIN7 B2B
27-20 HWTS7	CHAIN7 HWTS
19-16 CSEL7	CHAIN7 CSEL
15 IE6_EN	IRQ enable
14-13 IE6	CHAIN6 IE
12 B2B6	CHAIN6 B2B
11-4 HWTS6	CHAIN6 HWTS
3-0 CSEL6	CHAIN6 CSEL

54.5.1.12 ETC_TRIG Result Data 1/0 Register (TRIG0_RESULT_1_0 - TRIG3_RESULT_1_0)

54.5.1.12.1 Offset

Register	Offset
TRIG0_RESULT_1_0	28h
TRIG1_RESULT_1_0	50h
TRIG2_RESULT_1_0	78h
TRIG3_RESULT_1_0	A0h

54.5.1.12.2 Function

This register contains the result data of ETC_TRIG 1/0

54.5.1.12.3 Diagram

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	Reserved				DATA1											
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				DATA0											
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

54.5.1.12.4 Fields

Field	Function
31-28 —	Reserved
27-16 DATA1	Result DATA1
15-12 —	Reserved
11-0 DATA0	Result DATA0

54.5.1.13 ETC_TRIG Result Data 3/2 Register (TRIG0_RESULT_3_2 - TRIG3_RESULT_3_2)

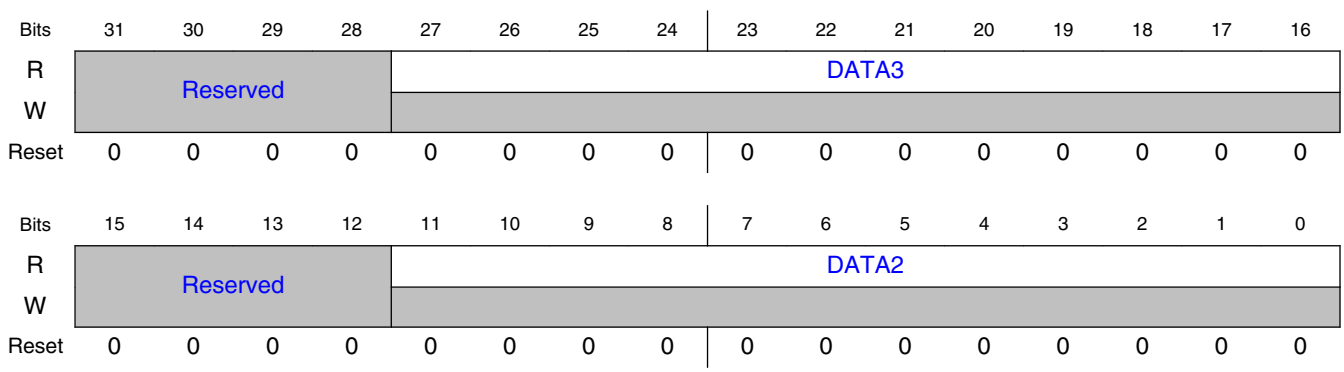
54.5.1.13.1 Offset

Register	Offset
TRIG0_RESULT_3_2	2Ch
TRIG1_RESULT_3_2	54h
TRIG2_RESULT_3_2	7Ch
TRIG3_RESULT_3_2	A4h

54.5.1.13.2 Function

This register contains the result data of ETC_TRIG 3/2

54.5.1.13.3 Diagram



54.5.1.13.4 Fields

Field	Function
31-28 —	Reserved
27-16 DATA3	Result DATA3
15-12 —	Reserved
11-0	Result DATA2

Field	Function
DATA2	

54.5.1.14 ETC_TRIG Result Data 5/4 Register (TRIG0_RESULT_5_4 - TRIG3_RESULT_5_4)

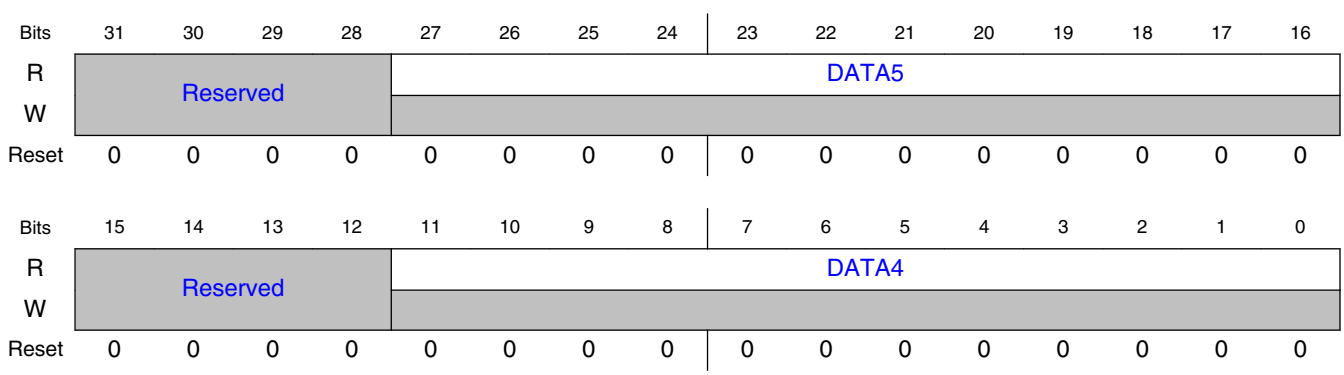
54.5.1.14.1 Offset

Register	Offset
TRIG0_RESULT_5_4	30h
TRIG1_RESULT_5_4	58h
TRIG2_RESULT_5_4	80h
TRIG3_RESULT_5_4	A8h

54.5.1.14.2 Function

This register contains the result data of ETC_TRIG 5/4

54.5.1.14.3 Diagram



54.5.1.14.4 Fields

Field	Function
31-28 —	Reserved
27-16	Result DATA5

Table continues on the next page...

Field	Function
DATA5	
15-12 —	Reserved
11-0 DATA4	Result DATA4

54.5.1.15 ETC_TRIG Result Data 7/6 Register (TRIG0_RESULT_7_6 - TRIG3_RESULT_7_6)

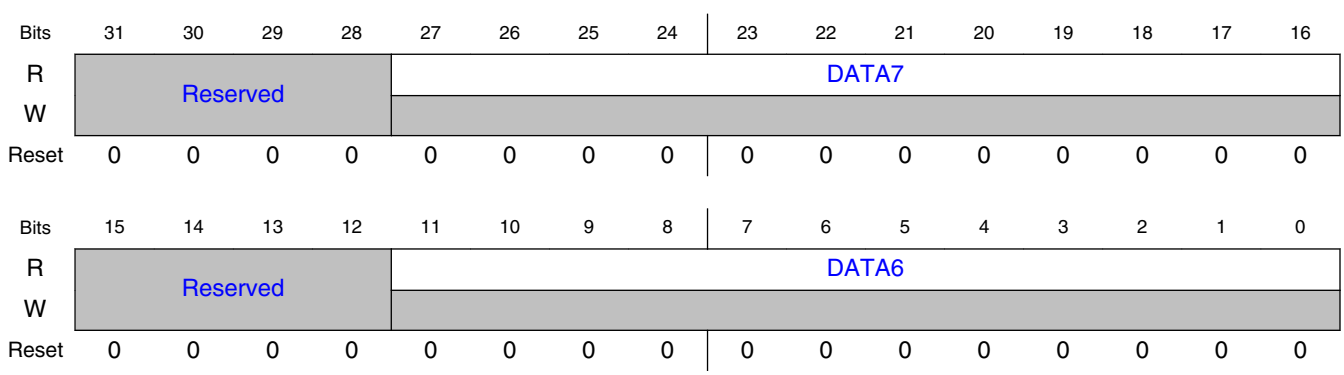
54.5.1.15.1 Offset

Register	Offset
TRIG0_RESULT_7_6	34h
TRIG1_RESULT_7_6	5Ch
TRIG2_RESULT_7_6	84h
TRIG3_RESULT_7_6	ACh

54.5.1.15.2 Function

This register contains the result data of ETC_TRIG 7/6

54.5.1.15.3 Diagram



54.5.1.15.4 Fields

Field	Function
31-28 —	Reserved
27-16 DATA7	Result DATA7
15-12 —	Reserved
11-0 DATA6	Result DATA6

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	09/2019	Initial public release.

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number IMXRT1010RM
Revision 0, 09/2019